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

  Code highlighter that highlights the source code on each step.
- class FileDialog

  File Dialog for opening and saving files.
- class MenuItem

  Items in the menu screen to navigate to other screens.
- class RandomTextInput

  Text input that supports random values.
- class SequenceController

  Controls the display of frames of the animation sequence.
- class SideBar

  "Side bar" for extra navigation
- class TextInput

  Input for entering text.

5.2 constants Namespace Reference

Variables

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

5.2.1 Variable Documentation

5.2.1.1 ani_speed

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

Definition at line 11 of file constants.hpp.

5.2.1.2 default_color_path

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

Definition at line 20 of file constants.hpp.

5.2.1.3 default_font_size

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

Definition at line 18 of file constants.hpp.

5.2.1.4 frames_per_second

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

Definition at line 8 of file constants.hpp.

5.2.1.5 max_val

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

Definition at line 16 of file constants.hpp.

5.2.1.6 min_val

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

Definition at line 15 of file constants.hpp.

5.2.1.7 scene_height

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

Definition at line 7 of file constants.hpp.

5.2.1.8 scene_width

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

Definition at line 6 of file constants.hpp.

5.2.1.9 sidebar_width

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

Definition at line 10 of file constants.hpp.

5.2.1.10 text_buffer_size

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

Definition at line 13 of file constants.hpp.

5.3 core Namespace Reference

Classes

- class BaseList

  The base container for implementing other data structures.
- class Deque

  The deque container.
- class DoublyLinkedList

  The doubly linked list container.
- class Queue

  The queue container.
- class Stack

  The stack container.

5.4 gui Namespace Reference

Namespaces

- namespace internal

Classes

- class GuiArray

  The GUI array container.
- class GuiCircularLinkedList

  The GUI circular linked list container.
- class GuiDoublyLinkedList

  The GUI doubly linked list container.
- class GuiDynamicArray

  The GUI dynamic array container.
- class GuiElement

  The GUI element (used in arrays)
- class GuiLinkedList

  The GUI linked list container.
- class GuiNode

  The GUI node (used in linked lists)
- class GuiQueue

  The GUI queue container.
- class GuiStack

  The GUI stack container.

5.5 gui::internal Namespace Reference

Classes

- class Base

  The base class for all GUI classes.

5.6 scene Namespace Reference

Namespaces

- namespace internal

Classes

- class ArrayScene
  The array scene.
- class BaseLinkedListScene
  The base linked list scene.
- class DynamicArrayScene
  The dynamic array scene.
- class MenuScene
  The menu scene.
- class QueueScene
  The queue scene.
- class SceneRegistry
  The scene registry.
- class SettingsScene
  The settings scene.
- class StackScene
  The stack scene.

Typedefs

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

Enumerations

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

5.6.1 Typedef Documentation

5.6.1.1 CircularLinkedListScene

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

Definition at line 175 of file base_linked_list_scene.hpp.

5.6.1.2 DoublyLinkedListScene

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

Definition at line 173 of file base_linked_list_scene.hpp.

5.6.1.3 LinkedListScene

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

Definition at line 172 of file base_linked_list_scene.hpp.

5.6.2 Enumeration Type Documentation

5.6.2.1 SceneId

```
enum scene::SceneId
```

The scene ID.

Enumerator

Array	
DynamicArray	
LinkedList	
DoublyLinkedList	
CircularLinkedList	
Stack	
Queue	
Menu	
Settings	

Definition at line 10 of file scene_id.hpp.

5.7 scene::internal Namespace Reference

Classes

- class BaseScene

  The base scene class.
- struct SceneOptions

  The scene options.

5.8 utils Namespace Reference

The utility functions.

Functions

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

  Draws text with custom font size and spacing.
- Vector2 MeasureText (const char *text, float font_size, float spacing)

  Measures the text with custom font size and spacing.
- core::Deque< int > str_extract_data (char str[constants::text_buffer_size])

  Extracts integers from a string separated by commas.
- bool val_in_range (int num)

  Checks if a value is in range [min_val, max_val].
- void unreachable ()

  Tells the compiler that this branch is unreachable.
- char * strtok (char *str, const char *delim, char **save_ptr)

  Splits a string into tokens.
- Color color_from_hex (const std::string &hex)

  Converts a hex string to a color.
- Color adaptive_text_color (Color bg_color)

  Returns the color of the text based on the background color.
- template<typename T >
  T get_random (T low, T high)

  Get a random number in the range [low, high].

5.8.1 Detailed Description

The utility functions.

5.8.2 Function Documentation

5.8.2.1 adaptive_text_color()

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

Returns the color of the text based on the background color.

Parameters

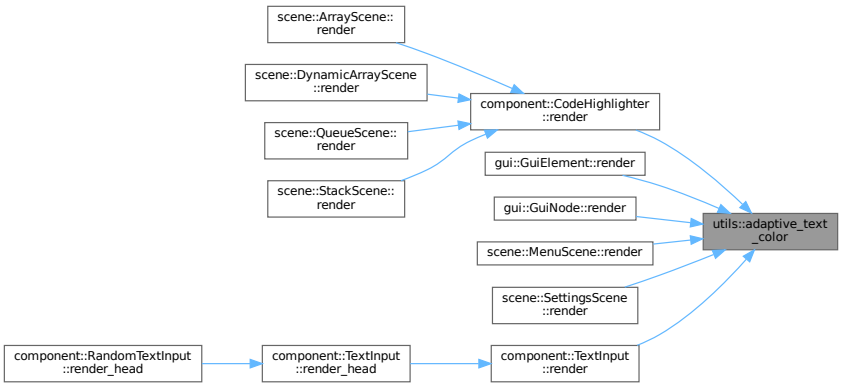
color	The background color

Return values

BLACK	The text color is black
WHITE	The text color is white

Definition at line 90 of file utils.cpp.

Here is the caller graph for this function:



5.8.2.2 color_from_hex()

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

Converts a hex string to a color.

Parameters

str	The string

Definition at line 82 of file utils.cpp.

5.8.2.3 DrawText()

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

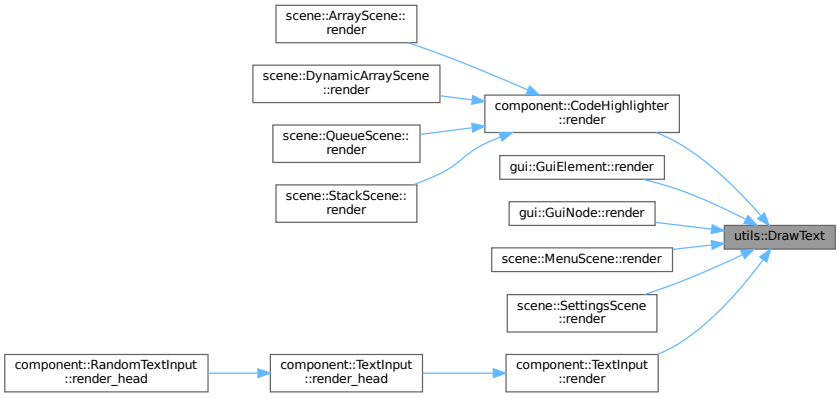
Draws text with custom font size and spacing.

Parameters

text	The drawn text
pos	The position of the text
color	The color of the text
font_size	The font size of the text
spacing	The spacing of the text

Definition at line 14 of file utils.cpp.

Here is the caller graph for this function:



5.8.2.4 get_random()

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

Get a random number in the range [low, high].

Template Parameters

T	Integral type

Parameters

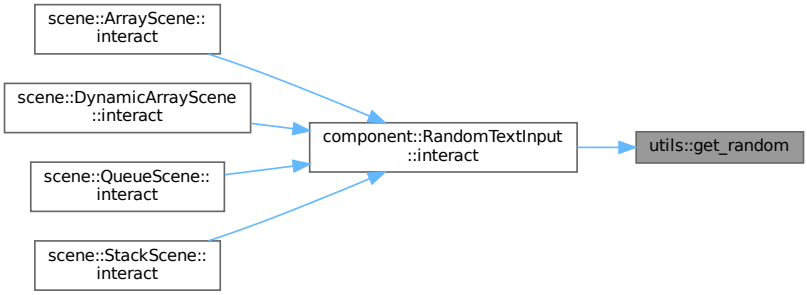
low	The lower bound
high	The upper bound

Returns

T The random number

Definition at line 48 of file utils.hpp.

Here is the caller graph for this function:



5.8.2.5 MeasureText()

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

Measures the text with custom font size and spacing.

Parameters

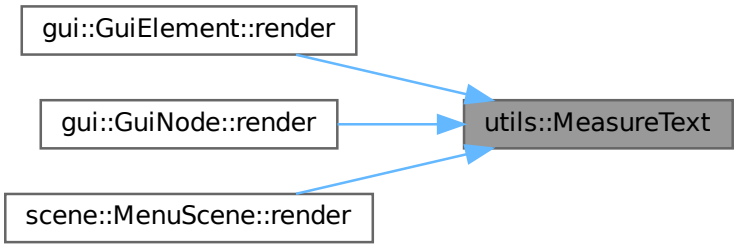
text	The measured text
font_size	The font size of the text
spacing	The spacing of the text

Returns

Vector2 The size of the text

Definition at line 23 of file utils.cpp.

Here is the caller graph for this function:



5.8.2.6 str_extract_data()

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

Extracts integers from a string separated by commas.

Parameters

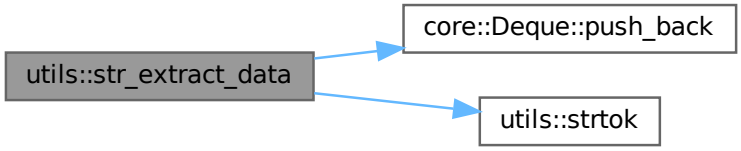
str	The string

Returns

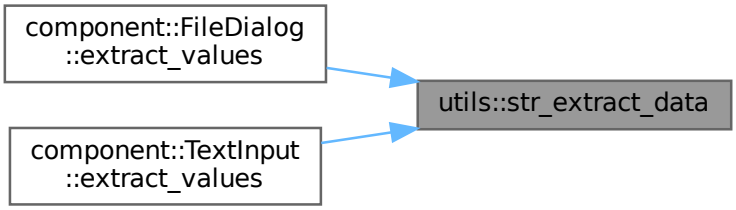
core::Deque<int> The extracted data

Definition at line 30 of file utils.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.8.2.7 strtok()

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

Splits a string into tokens.

Parameters

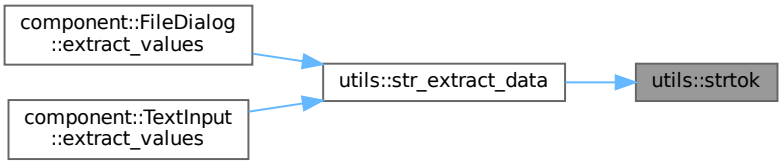
str	The string
delim	The delimiter
save_ptr	The save pointer

Returns

char* The token

Definition at line 73 of file utils.cpp.

Here is the caller graph for this function:



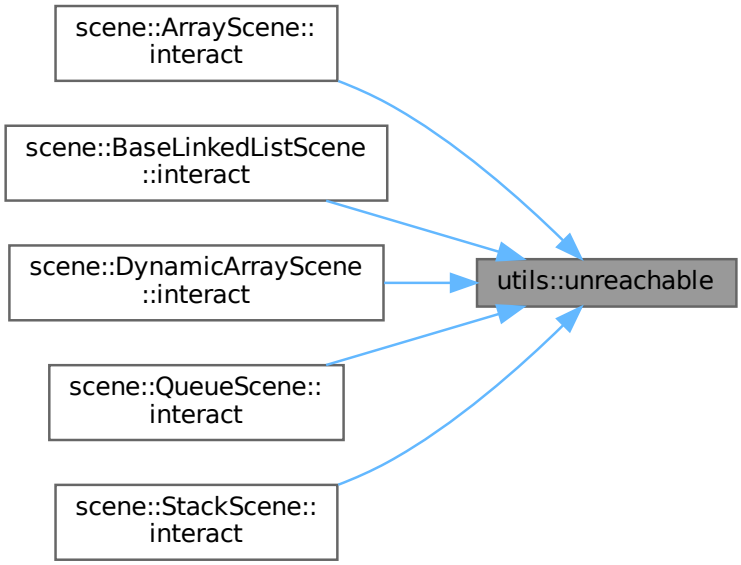
5.8.2.8 unreachable()

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

Tells the compiler that this branch is unreachable.

Definition at line 65 of file utils.cpp.

Here is the caller graph for this function:



5.8.2.9 val_in_range()

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

Checks if a value is in range [min_val, max_val].

Parameters

num	The value

Return values

true	The value is in range
false	The value is not in range

Definition at line 61 of file utils.cpp.

Chapter 6

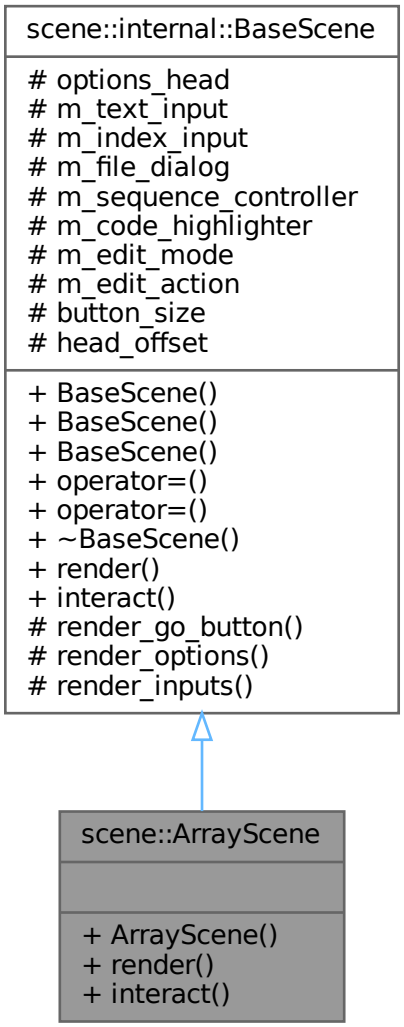
Class Documentation

6.1 scene::ArrayScene Class Reference

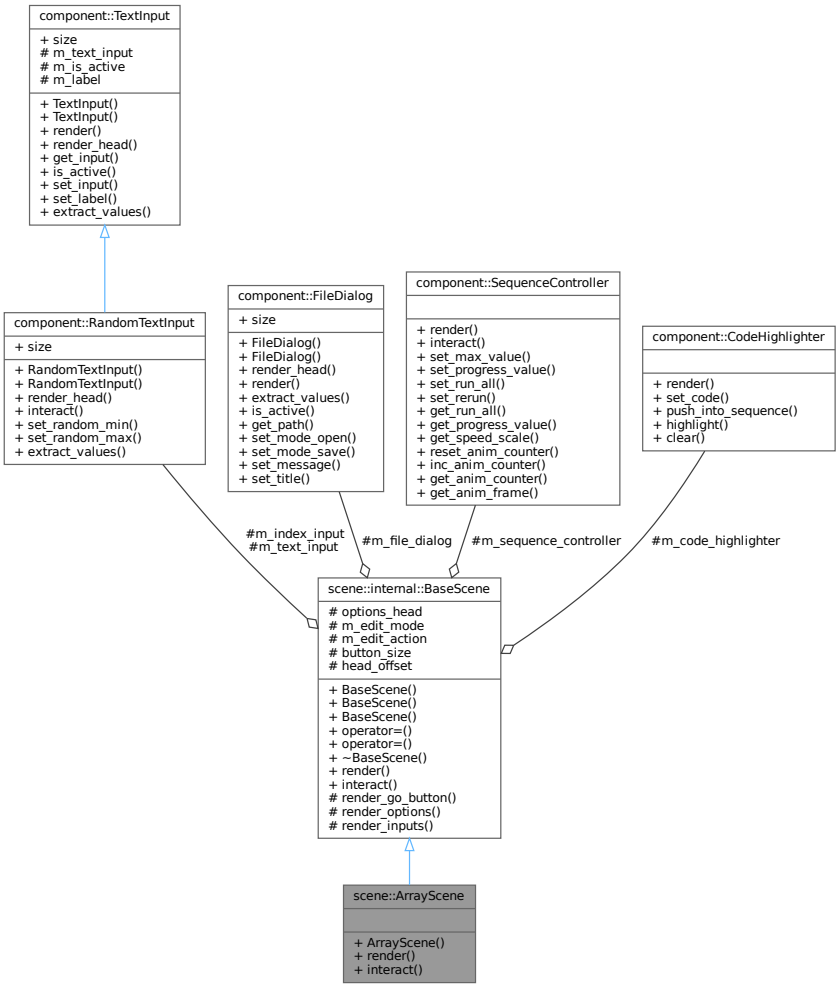
The array scene.

`#include <array_scene.hpp>`

Inheritance diagram for scene::ArrayScene:



Collaboration diagram for scene::ArrayScene:



Public Member Functions

- ArrayScene ()

  Construct a new ArrayScene object.
- void render () override

  Renders the scene.
- void interact () override

  Interacts with the scene.

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

- BaseScene ()=default

  Construct a new BaseScene object.
- BaseScene (const BaseScene &)=delete

  Copy constructor (deleted)
- BaseScene (BaseScene &&)=delete

Move constructor (deleted)

- BaseScene & operator= (const BaseScene &)=delete

  Copy assignment (deleted)
- BaseScene & operator= (BaseScene &&)=delete

  Move assignment (deleted)
- virtual ~BaseScene ()=default

  Destroy the BaseScene object.
- virtual void render ()

  Renders the scene.
- virtual void interact ()

  Interacts with the scene.

Additional Inherited Members

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

- virtual bool render_go_button () const

  Renders the go button.
- virtual void render_options (SceneOptions &scene_config)

  Renders the options.
- virtual void render_inputs ()

  Renders the inputs.

Protected Attributes inherited from scene::internal::BaseScene

- float options_head {}

  The head of the options.
- component::RandomTextInput m_text_input {"value"}

  The text input for the value.
- component::RandomTextInput m_index_input {"index"}

  The text input for the index.
- component::FileDialog m_file_dialog

  The file dialog.
- component::SequenceController m_sequence_controller

  The sequence controller.
- component::CodeHighlighter m_code_highlighter

  The code highlighter.
- bool m_edit_mode {}

  Whether the edit mode is enabled.
- bool m_edit_action {}

  Whether the edit action is enabled.

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

- static constexpr Vector2 button_size {200, 50}

  The size of the buttons.
- static constexpr int head_offset = 20

  The offset of the widgets.

6.1.1 Detailed Description

The array scene.

Definition at line 21 of file array_scene.hpp.

6.1.2 Constructor & Destructor Documentation

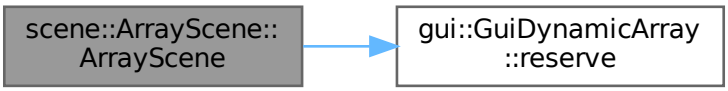
6.1.2.1 ArrayScene()

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

Construct a new ArrayScene object.

Definition at line 17 of file array_scene.cpp.

Here is the call graph for this function:



6.1.3 Member Function Documentation

6.1.3.1 interact()

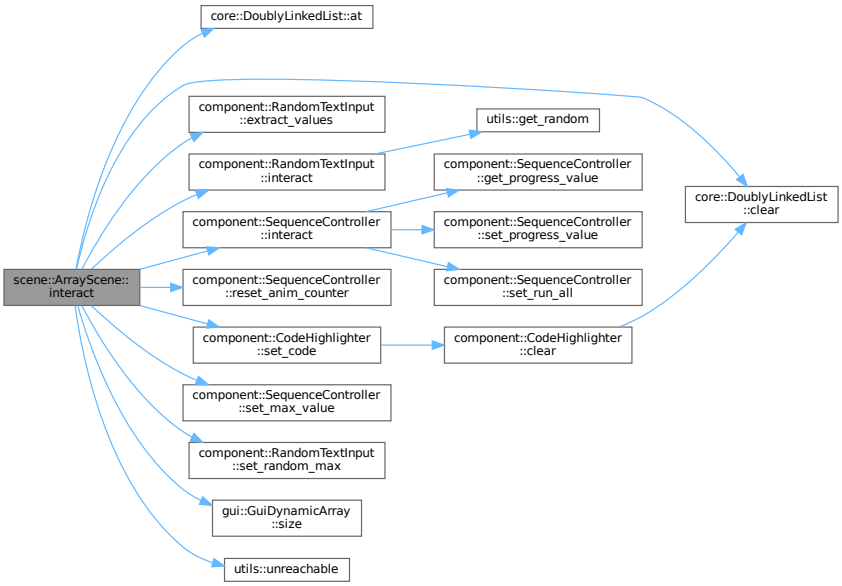
```
void scene::ArrayScene::interact ( )  [override], [virtual]
```

Interacts with the scene.

Reimplemented from scene::internal::BaseScene.

Definition at line 89 of file array_scene.cpp.

Here is the call graph for this function:



6.1.3.2 render()

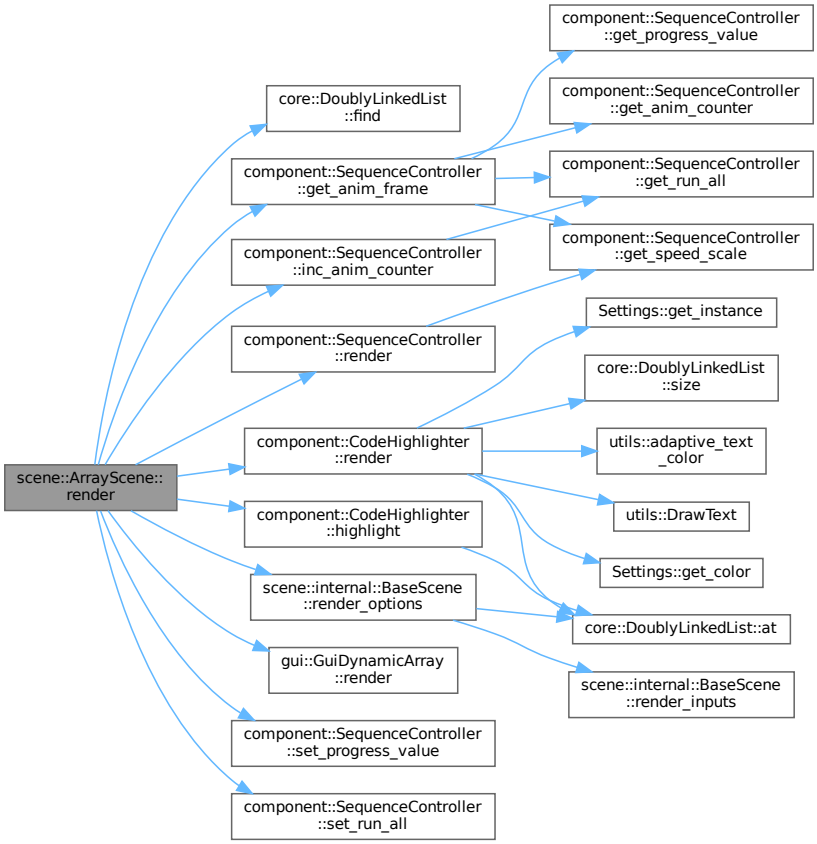
```
void scene::ArrayScene::render ( )  [override], [virtual]
```

Renders the scene.

Reimplemented from scene::internal::BaseScene.

Definition at line 69 of file array_scene.cpp.

Here is the call graph for this function:



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

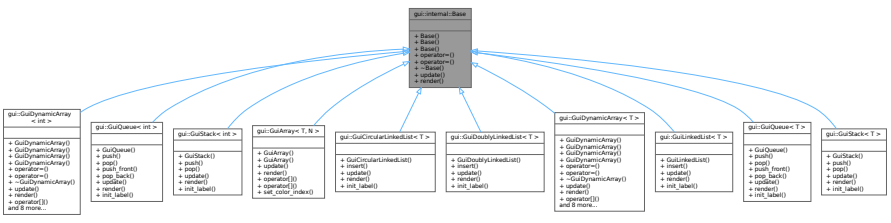
- src/scene/array_scene.hpp
- src/scene/array_scene.cpp

6.2 gui::internal::Base Class Reference

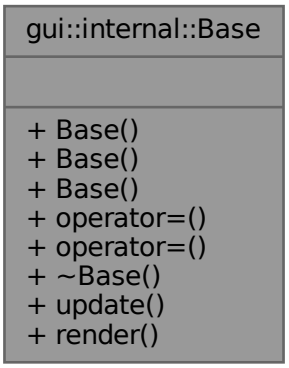
The base class for all GUI classes.

`#include <base_gui.hpp>`

Inheritance diagram for gui::internal::Base:



Collaboration diagram for gui::internal::Base:



Public Member Functions

- Base ()=default

  Constructs a Base object.
- Base (const Base &)=default

  Copy constructor.
- Base (Base &&)=default

  Move constructor.
- Base & operator= (const Base &)=default

  Copy assignment operator.
- Base & operator= (Base &&)=default

  Move assignment operator.
- virtual ~Base ()=default

  Destructor.
- virtual void update ()=0

  Updates the GUI.
- virtual void render ()=0

  Renders the GUI.

6.2.1 Detailed Description

The base class for all GUI classes.

Definition at line 12 of file base_gui.hpp.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 Base() [1/3]

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

Constructs a Base object.

6.2.2.2 Base() [2/3]

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

Copy constructor.

6.2.2.3 Base() [3/3]

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

Move constructor.

6.2.2.4 ~Base()

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

Destructor.

6.2.3 Member Function Documentation

6.2.3.1 operator=() [1/2]

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

Move assignment operator.

6.2.3.2 operator=() [2/2]

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

Copy assignment operator.

6.2.3.3 render()

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

Renders the GUI.

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

6.2.3.4 update()

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

Updates the GUI.

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

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

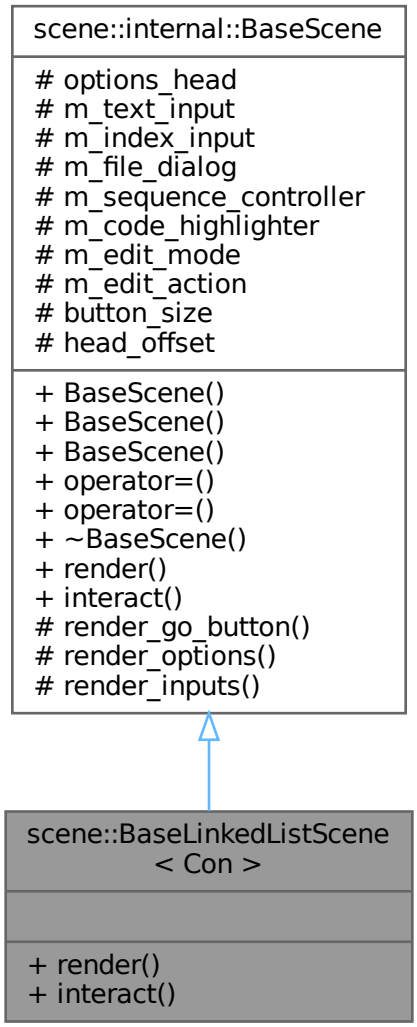
- src/gui/base_gui.hpp

6.3 scene::BaseLinkedListScene< Con > Class Template Reference

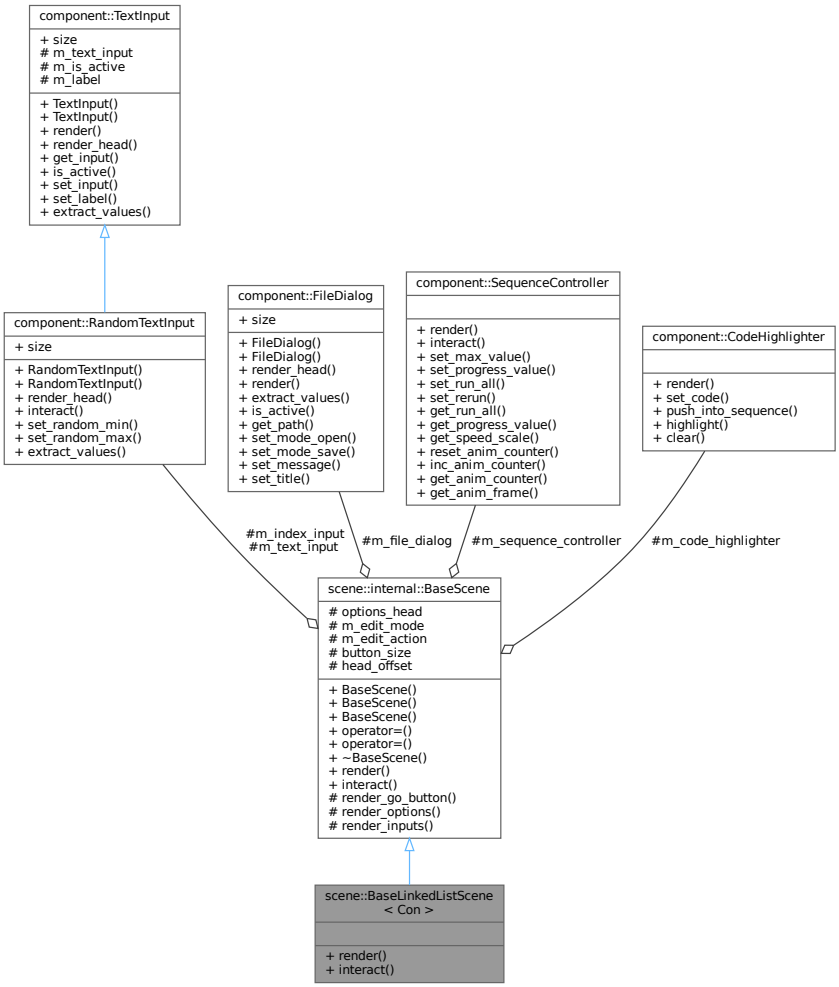
The base linked list scene.

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

Inheritance diagram for scene::BaseLinkedListScene< Con >:



Collaboration diagram for scene::BaseLinkedListScene< Con >:



Public Member Functions

- void render () override

  Renders the scene.
- void interact () override

  Interacts with the scene.

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

- BaseScene ()=default

  Construct a new BaseScene object.
- BaseScene (const BaseScene &)=delete

  Copy constructor (deleted)
- BaseScene (BaseScene &&)=delete

  Move constructor (deleted)
- BaseScene & operator= (const BaseScene &)=delete

Copy assignment (deleted)

- BaseScene & operator= (BaseScene &&)=delete

  Move assignment (deleted)
- virtual ~BaseScene ()=default

  Destroy the BaseScene object.
- virtual void render ()

  Renders the scene.
- virtual void interact ()

  Interacts with the scene.

Additional Inherited Members

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

- virtual bool render_go_button () const

  Renders the go button.
- virtual void render_options (SceneOptions &scene_config)

  Renders the options.
- virtual void render_inputs ()

  Renders the inputs.

Protected Attributes inherited from scene::internal::BaseScene

- float options_head {}

  The head of the options.
- component::RandomTextInput m_text_input {"value"}

  The text input for the value.
- component::RandomTextInput m_index_input {"index"}

  The text input for the index.
- component::FileDialog m_file_dialog

  The file dialog.
- component::SequenceController m_sequence_controller

  The sequence controller.
- component::CodeHighlighter m_code_highlighter

  The code highlighter.
- bool m_edit_mode {}

  Whether the edit mode is enabled.
- bool m_edit_action {}

  Whether the edit action is enabled.

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

- static constexpr Vector2 button_size {200, 50}

  The size of the buttons.
- static constexpr int head_offset = 20

  The offset of the widgets.

6.3.1 Detailed Description

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

The base linked list scene.

Template Parameters

Con	the container type

Definition at line 21 of file base_linked_list_scene.hpp.

6.3.2 Member Function Documentation

6.3.2.1 interact()

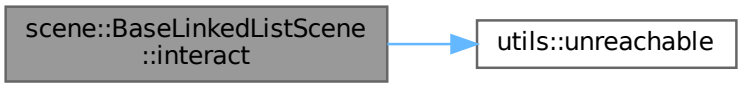
```
template<typename Con >
void scene::BaseLinkedListScene< Con >::interact  [override], [virtual]
```

Interacts with the scene.

Reimplemented from scene::internal::BaseScene.

Definition at line 247 of file base_linked_list_scene.hpp.

Here is the call graph for this function:



6.3.2.2 render()

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

Renders the scene.

Reimplemented from scene::internal::BaseScene.

Definition at line 226 of file base_linked_list_scene.hpp.

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

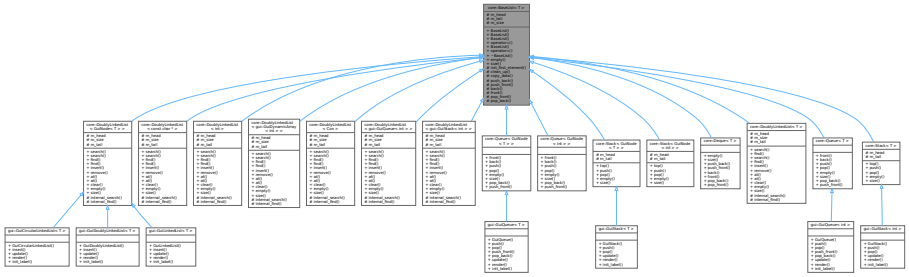
- src/scene/base_linked_list_scene.hpp

6.4 core::BaseList< T > Class Template Reference

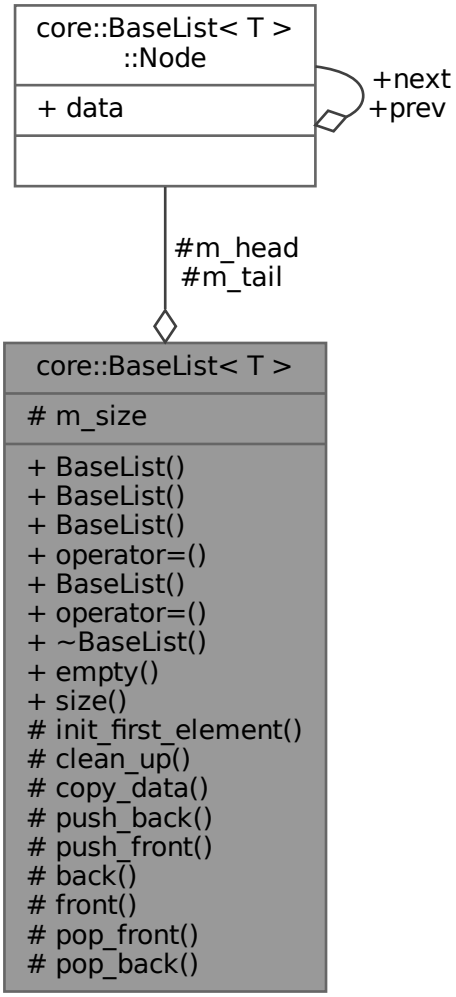
The base container for implementing other data structures.

`#include <base_list.hpp>`

Inheritance diagram for core::BaseList< T >:



Collaboration diagram for core::BaseList< T >:



Classes

- struct Node

  The node of the list.

Public Member Functions

- BaseList ()=default

  Default constructor.
- BaseList (std::initializer_list< T > init_list)

  Constructs the container with the contents of the initializer list.
- BaseList (const BaseList &rhs)

  Copy constructor.

- BaseList & operator= (const BaseList &rhs)
  Copy assignment operator.
- BaseList (BaseList &&rhs) noexcept
  Move constructor.
- BaseList & operator= (BaseList &&rhs) noexcept
  Move assignment operator.
- ~BaseList ()
  Destructor.
- bool empty () const
  Check whether the container is empty.
- std::size_t size () const
  Returns the size of the container.

Protected Types

- using Node_ptr = Node *

Protected Member Functions

- void init_first_element (const T &elem)
  Initializes the first element of the container.
- void clean_up ()
  Frees all elements in the container.
- void copy_data (const BaseList &rhs)
  Copies data from another container.
- void push_back (const T &elem)
  Pushes the element to the back of the container.
- void push_front (const T &elem)
  Pushes the element to the front of the container.
- T & back () const
  Returns the reference to the element at the back of the container.
- T & front () const
  Returns the reference to the element at the front of the container.
- void pop_front ()
  Removes the element at the back of the container.
- void pop_back ()
  Removes the element at the front of the container.

Protected Attributes

- Node_ptr m_head {nullptr}
  The head of the list.
- Node_ptr m_tail {nullptr}
  The tail of the list.
- std::size_t m_size {}
  The size of the list.

6.4.1 Detailed Description

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

The base container for implementing other data structures.

Template Parameters

T	the type of the elements

Definition at line 16 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 73 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]
```

Default constructor.

6.4.3.2 BaseList() [2/4]

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

Constructs the container with the contents of the initializer list.

Parameters

init_list	The initializer list

Definition at line 159 of file base_list.hpp.

6.4.3.3 BaseList() [3/4]

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

Copy constructor.

Parameters

rhs	The other container

Definition at line 154 of file base_list.hpp.

6.4.3.4 BaseList() [4/4]

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

Move constructor.

Parameters

rhs	The other container

Definition at line 175 of file base_list.hpp.

6.4.3.5 ~BaseList()

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

Destructor.

Definition at line 200 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]
```

Returns the reference to the element at the back of the container.

Returns

T& The reference to the element at the back of the container

Definition at line 267 of file base_list.hpp.

6.4.4.2 clean_up()

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

Frees all elements in the container.

Definition at line 222 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]
```

Copies data from another container.

Parameters

rhs	Tnother container

Definition at line 236 of file base_list.hpp.

6.4.4.4 empty()

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

Check whether the container is empty.

Return values

true	The container is empty
false	The container is not empty

Definition at line 205 of file base_list.hpp.

6.4.4.5 front()

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

Returns the reference to the element at the front of the container.

Returns

T& The reference to the element at the front of the container

Definition at line 272 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]
```

Initializes the first element of the container.

Parameters

elem	The first element of the container

Definition at line 215 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]
```

Move assignment operator.

Parameters

rhs	The other container

Definition at line 183 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 )
```

Copy assignment operator.

Parameters

rhs	The other container

Definition at line 166 of file base_list.hpp.

6.4.4.9 pop_back()

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

Removes the element at the front of the container.

Definition at line 277 of file base_list.hpp.

6.4.4.10 pop_front()

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

Removes the element at the back of the container.

Definition at line 290 of file base_list.hpp.

6.4.4.11 push_back()

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

Pushes the element to the back of the container.

Parameters

elem	The element to be pushed into the back

Definition at line 243 of file base_list.hpp.

6.4.4.12 push_front()

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

Pushes the element to the front of the container.

Parameters

elem	The element to be pushed into the front

Definition at line 255 of file base_list.hpp.

6.4.4.13 size()

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

Returns the size of the container.

Returns

The size of the container

Definition at line 210 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]
```

The head of the list.

Definition at line 87 of file base_list.hpp.

6.4.5.2 m_size

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

The size of the list.

Definition at line 97 of file base_list.hpp.

6.4.5.3 m_tail

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

The tail of the list.

Definition at line 92 of file base_list.hpp.

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

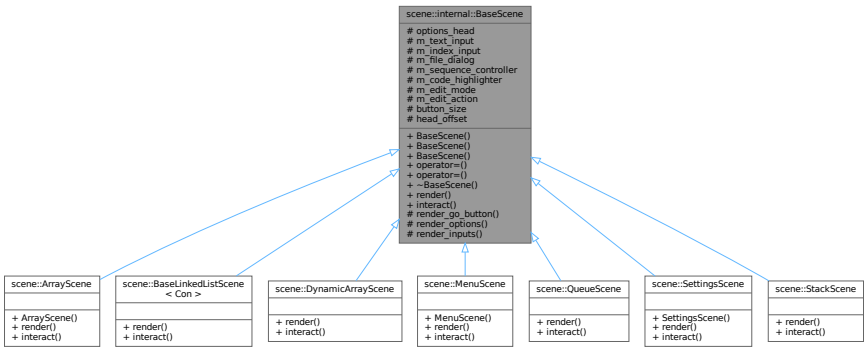
- src/core/base_list.hpp

6.5 scene::internal::BaseScene Class Reference

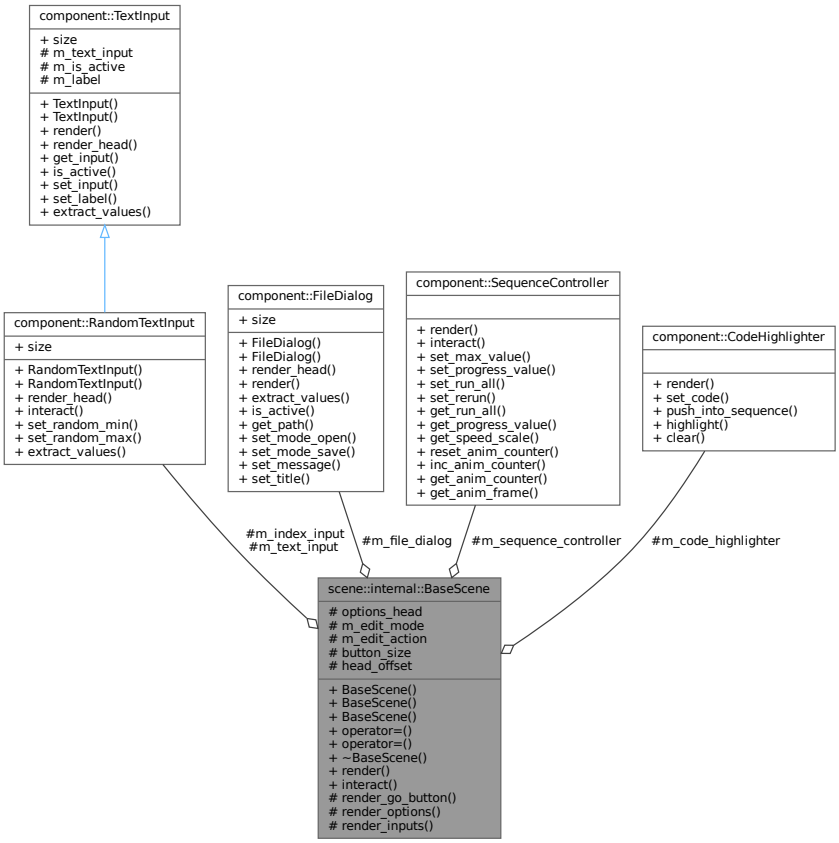
The base scene class.

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

Inheritance diagram for scene::internal::BaseScene:



Collaboration diagram for scene::internal::BaseScene:



Public Member Functions

- BaseScene ()=default

  Construct a new BaseScene object.
- BaseScene (const BaseScene &)=delete

  Copy constructor (deleted)
- BaseScene (BaseScene &&)=delete

  Move constructor (deleted)
- BaseScene & operator= (const BaseScene &)=delete

  Copy assignment (deleted)
- BaseScene & operator= (BaseScene &&)=delete

  Move assignment (deleted)
- virtual ~BaseScene ()=default

  Destroy the BaseScene object.
- virtual void render ()

  Renders the scene.
- virtual void interact ()

  Interacts with the scene.

Protected Member Functions

- virtual bool render_go_button () const

  Renders the go button.
- virtual void render_options (SceneOptions &scene_config)

  Renders the options.
- virtual void render_inputs ()

  Renders the inputs.

Protected Attributes

- float options_head {}

  The head of the options.
- component::RandomTextInput m_text_input {"value"}

  The text input for the value.
- component::RandomTextInput m_index_input {"index"}

  The text input for the index.
- component::FileDialog m_file_dialog

  The file dialog.
- component::SequenceController m_sequence_controller

  The sequence controller.
- component::CodeHighlighter m_code_highlighter

  The code highlighter.
- bool m_edit_mode {}

  Whether the edit mode is enabled.
- bool m_edit_action {}

  Whether the edit action is enabled.

Static Protected Attributes

- static constexpr Vector2 button_size {200, 50}

  The size of the buttons.
- static constexpr int head_offset = 20

  The offset of the widgets.

6.5.1 Detailed Description

The base scene class.

Definition at line 17 of file base_scene.hpp.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 BaseScene() [1/3]

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

Construct a new BaseScene object.

6.5.2.2 BaseScene() [2/3]

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

Copy constructor (deleted)

6.5.2.3 BaseScene() [3/3]

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

Move constructor (deleted)

6.5.2.4 ~BaseScene()

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

Destroy the BaseScene object.

6.5.3 Member Function Documentation

6.5.3.1 interact()

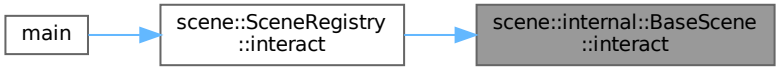
```
virtual void scene::internal::BaseScene::interact ( )  [inline], [virtual]
```

Interacts with the scene.

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

Definition at line 57 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]
```

Move assignment (deleted)

6.5.3.3 operator=() [2/2]

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

Copy assignment (deleted)

6.5.3.4 render()

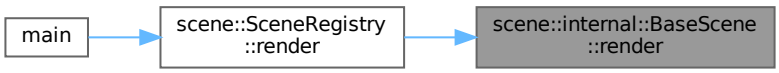
```
virtual void scene::internal::BaseScene::render ( )  [inline], [virtual]
```

Renders the scene.

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

Definition at line 52 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]
```

Renders the go button.

Return values

true	The go button was pressed

Returns

false The go button was not pressed

Definition at line 10 of file base_scene.cpp.

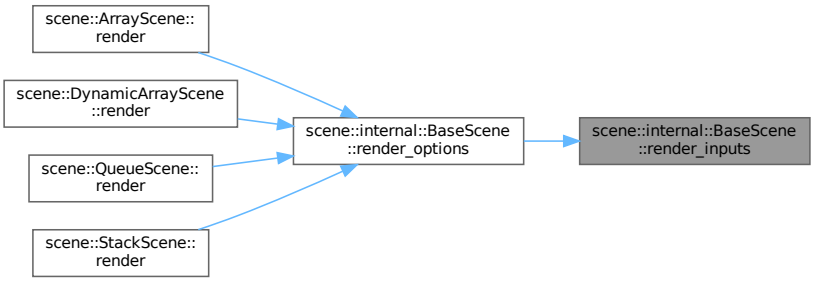
6.5.3.6 render_inputs()

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

Renders the inputs.

Definition at line 91 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]
```

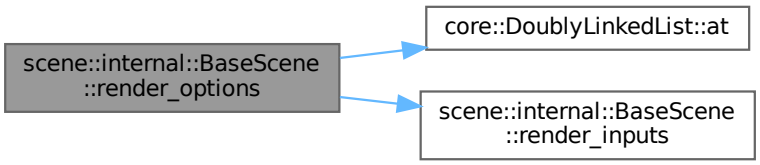
Renders the options.

Parameters

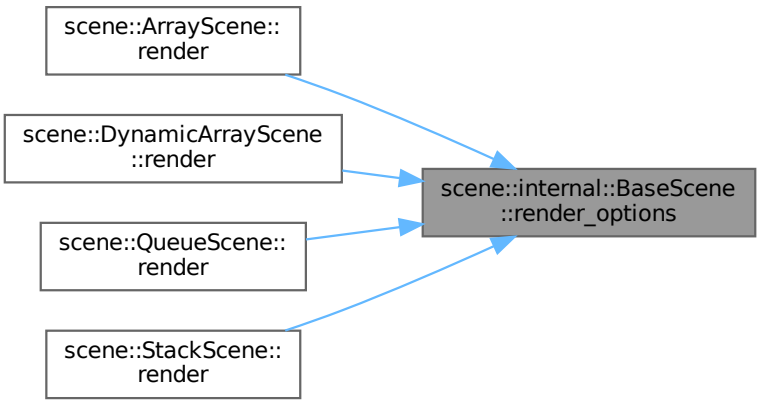
scene_config	The scene options

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]
```

The size of the buttons.

Definition at line 63 of file base_scene.hpp.

6.5.4.2 head_offset

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

The offset of the widgets.

Definition at line 68 of file base_scene.hpp.

6.5.4.3 m_code_highlighter

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

The code highlighter.

Definition at line 116 of file base_scene.hpp.

6.5.4.4 m_edit_action

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

Whether the edit action is enabled.

Definition at line 126 of file base_scene.hpp.

6.5.4.5 m_edit_mode

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

Whether the edit mode is enabled.

Definition at line 121 of file base_scene.hpp.

6.5.4.6 m_file_dialog

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

The file dialog.

Definition at line 106 of file base_scene.hpp.

6.5.4.7 m_index_input

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

The text input for the index.

Definition at line 101 of file base_scene.hpp.

6.5.4.8 m_sequence_controller

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

The sequence controller.

Definition at line 111 of file base_scene.hpp.

6.5.4.9 m_text_input

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

The text input for the value.

Definition at line 96 of file base_scene.hpp.

6.5.4.10 options_head

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

The head of the options.

Definition at line 73 of file base_scene.hpp.

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

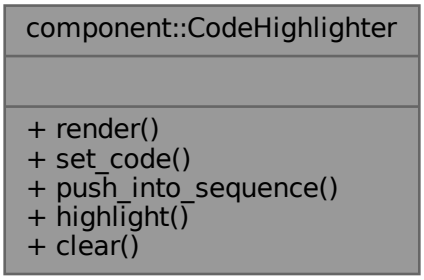
- src/scene/base_scene.hpp
- src/scene/base_scene.cpp

6.6 component::CodeHighlighter Class Reference

Code highlighter that highlights the source code on each step.

`#include <code_highlighter.hpp>`

Collaboration diagram for component::CodeHighlighter:



Public Member Functions

- void render ()

  Renders the code highlighter.
- void set_code (core::DoublyLinkedList< const char * > &&src_code)

  Set the source code to be highlighted.
- void push_into_sequence (int line_number)

  Pushes the line number to be highlighted into the sequence.
- void highlight (int frame_idx)

  Highlights the line number at the given frame index.
- void clear ()

  Clears the code highlighter.

6.6.1 Detailed Description

Code highlighter that highlights the source code on each step.

Definition at line 14 of file code_highlighter.hpp.

6.6.2 Member Function Documentation

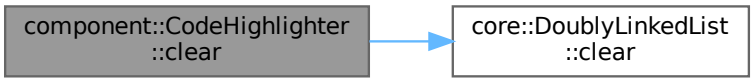
6.6.2.1 clear()

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

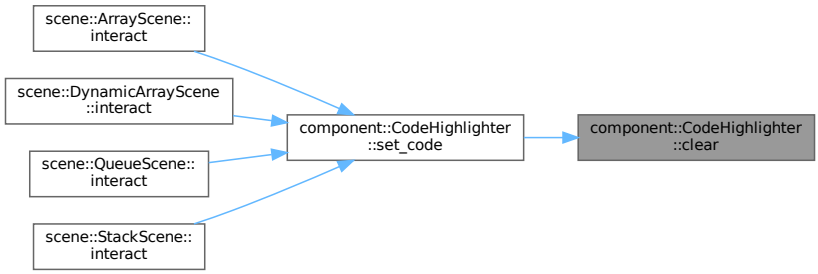
Clears the code highlighter.

Definition at line 38 of file code_highlighter.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



6.6.2.2 highlight()

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

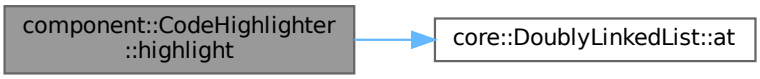
Highlights the line number at the given frame index.

Parameters

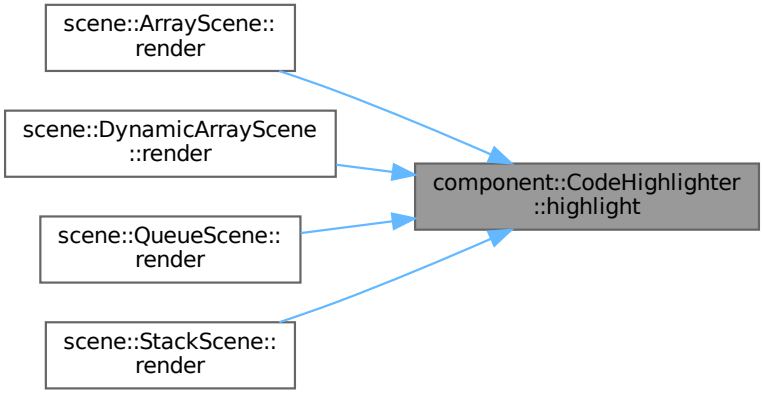
frame_idx	the frame index

Definition at line 34 of file code_highlighter.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



6.6.2.3 push_into_sequence()

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

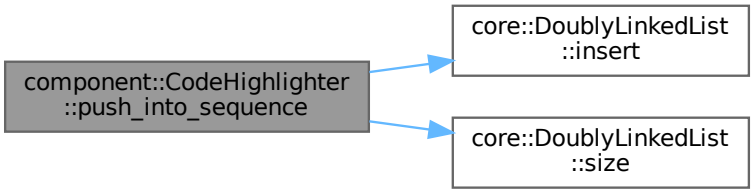
Pushes the line number to be highlighted into the sequence.

Parameters

line_number	the line number to be pushed into the sequence

Definition at line 30 of file code_highlighter.cpp.

Here is the call graph for this function:



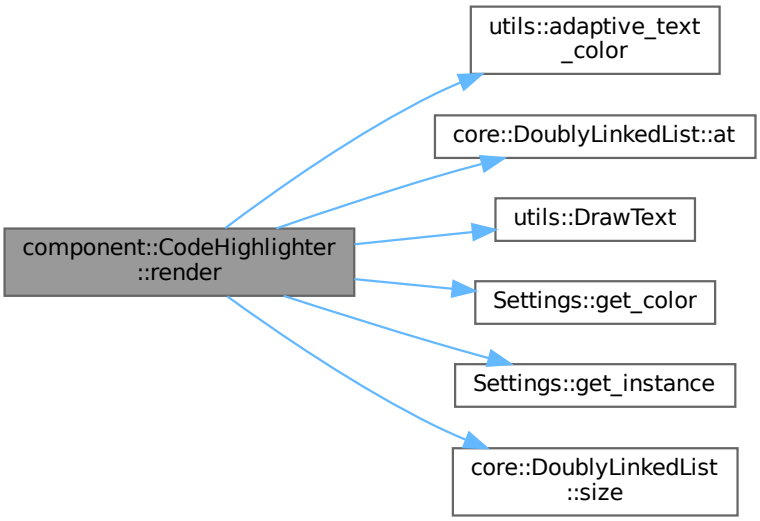
6.6.2.4 render()

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

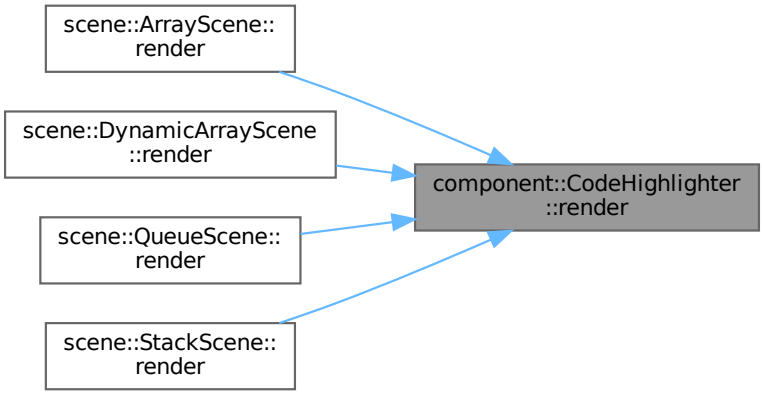
Renders the code highlighter.

Definition at line 9 of file code_highlighter.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



6.6.2.5 set_code()

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

Set the source code to be highlighted.

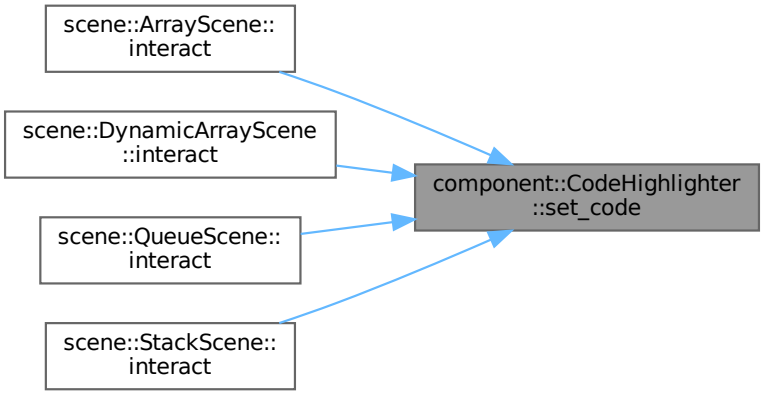
Parameters

src_code	a collection of lines of source code

Definition at line 25 of file code_highlighter.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:

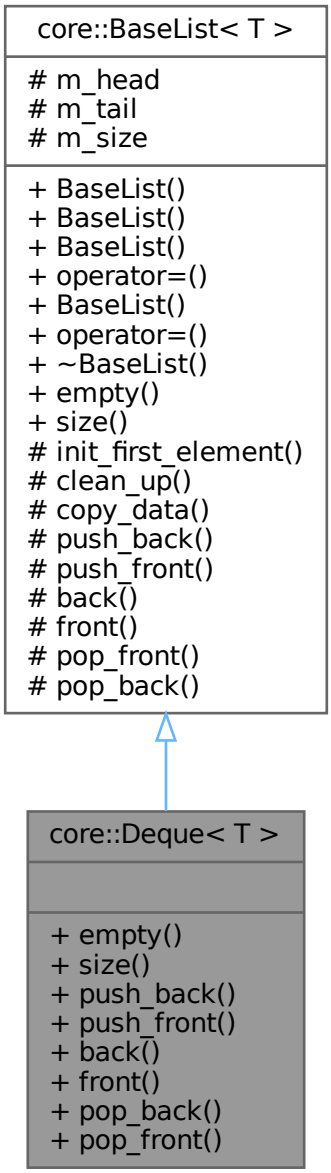
- src/component/code_highlighter.hpp
- src/component/code_highlighter.cpp

6.7 core::Deque< T > Class Template Reference

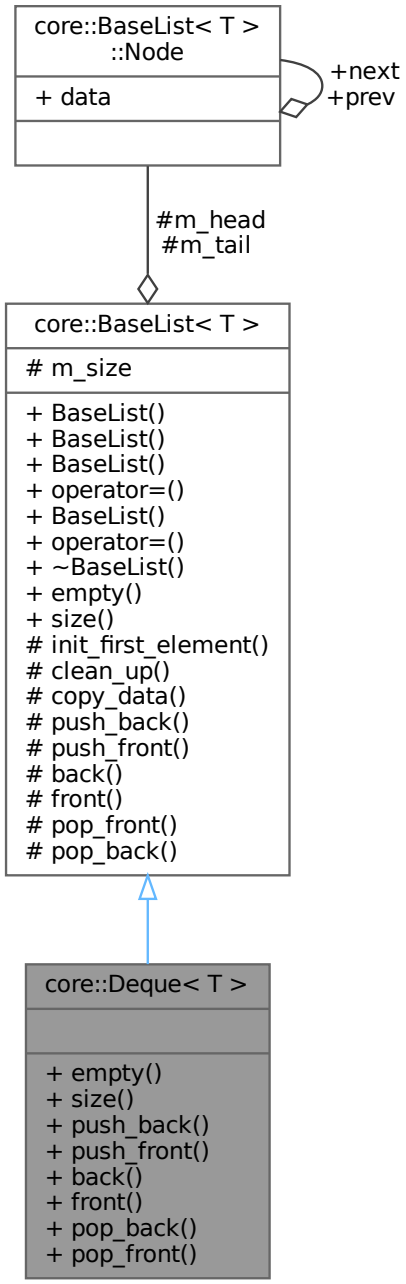
The deque container.

`#include <deque.hpp>`

Inheritance diagram for core::Deque< T >:



Collaboration diagram for core::Deque< T >:



Public Member Functions

- bool empty () const

  Check whether the container is empty.
- std::size_t size () const

  Returns the size of the container.
- void push_back (const T &elem)

Pushes the element to the back of the container.

- void push_front (const T &elem)

  Pushes the element to the front of the container.
- T & back () const

  Returns the reference to the element at the back of the container.
- T & front () const

  Returns the reference to the element at the front of the container.
- void pop_back ()

  Removes the element at the front of the container.
- void pop_front ()

  Removes the element at the back of the container.

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

- BaseList ()=default

  Default constructor.
- BaseList (std::initializer_list< T > init_list)

  Constructs the container with the contents of the initializer list.
- BaseList (const BaseList &rhs)

  Copy constructor.
- BaseList & operator= (const BaseList &rhs)

  Copy assignment operator.
- BaseList (BaseList &&rhs) noexcept

  Move constructor.
- BaseList & operator= (BaseList &&rhs) noexcept

  Move assignment operator.
- ~BaseList ()

  Destructor.
- bool empty () const

  Check whether the container is empty.
- std::size_t size () const

  Returns the size of the container.

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)

  Initializes the first element of the container.
- void clean_up ()

  Frees all elements in the container.
- void copy_data (const BaseList &rhs)

  Copies data from another container.
- void push_back (const T &elem)

  Pushes the element to the back of the container.

- void push_front (const T &elem)

  Pushes the element to the front of the container.
- T & back () const

  Returns the reference to the element at the back of the container.
- T & front () const

  Returns the reference to the element at the front of the container.
- void pop_front ()

  Removes the element at the back of the container.
- void pop_back ()

  Removes the element at the front of the container.

Protected Attributes inherited from core::BaseList< T >

- Node_ptr m_head {nullptr}

  The head of the list.
- Node_ptr m_tail {nullptr}

  The tail of the list.
- std::size_t m_size {}

  The size of the list.

6.7.1 Detailed Description

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

The deque container.

Template Parameters

T	the type of the elements

Definition at line 14 of file deque.hpp.

6.7.2 Member Function Documentation

6.7.2.1 back()

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

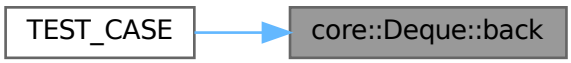
Returns the reference to the element at the back of the container.

Returns

T& The reference to the element at the back of the container

Definition at line 134 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
```

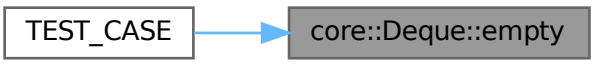
Check whether the container is empty.

Return values

true	The container is empty
false	The container is not empty

Definition at line 63 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
```

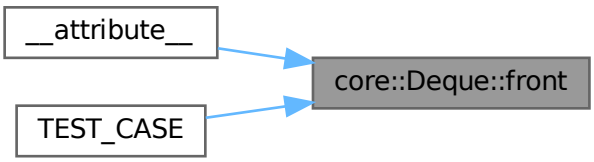
Returns the reference to the element at the front of the container.

Returns

T& The reference to the element at the front of the container

Definition at line 140 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
```

Removes the element at the front of the container.

Definition at line 150 of file base_list.hpp.

Here is the caller graph for this function:

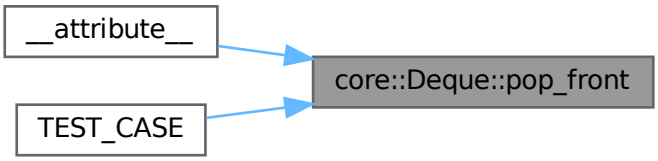
6.7.2.5 pop_front()

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

Removes the element at the back of the container.

Definition at line 145 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 )
```

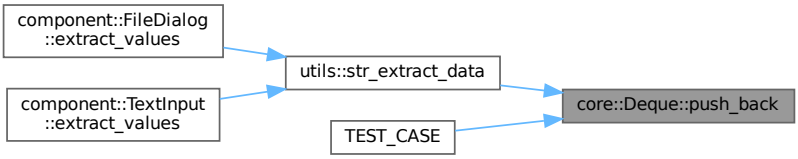
Pushes the element to the back of the container.

Parameters

elem	The element to be pushed into the back

Definition at line 122 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 )
```

Pushes the element to the front of the container.

Parameters

elem	The element to be pushed into the front

Definition at line 128 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
```

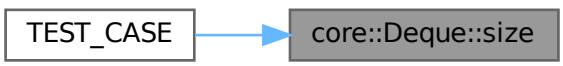
Returns the size of the container.

Returns

The size of the container

Definition at line 69 of file base_list.hpp.

Here is the caller graph for this function:



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

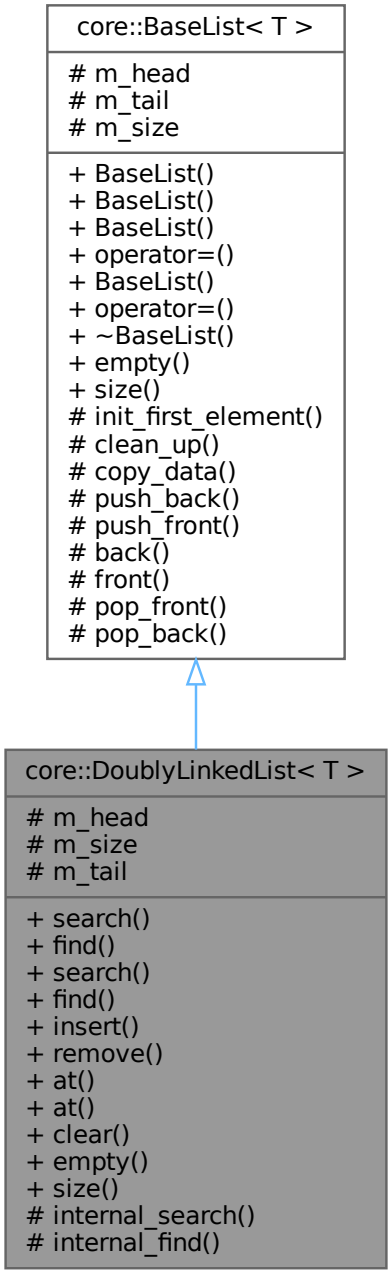
- src/core/deque.hpp

6.8 core::DoublyLinkedList< T > Class Template Reference

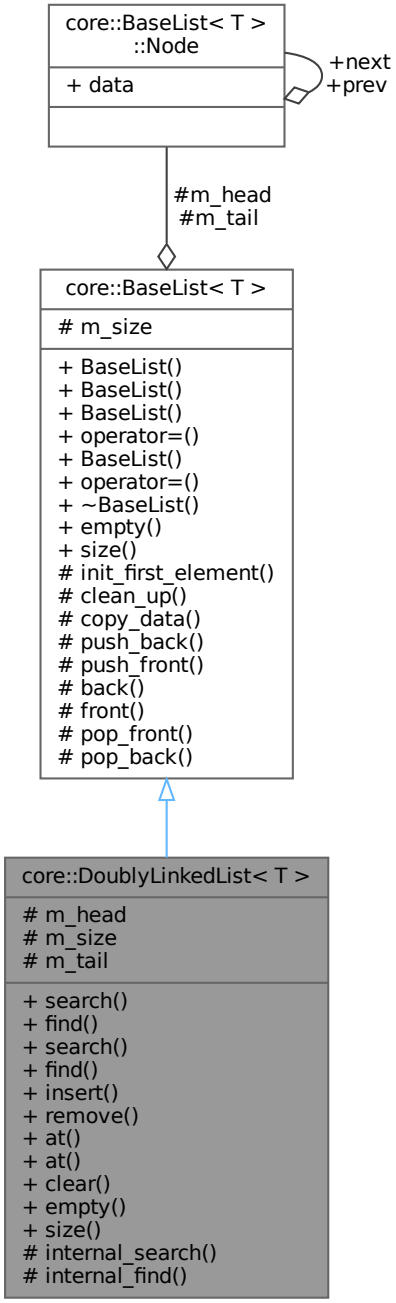
The doubly linked list container.

`#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)

  Searches for the element in the container.
- Node_ptr find (std::size_t index)

  Finds the element at the specified index.
- cNode_ptr search (const T &elem) const

Searches for the element in the container.

- cNode_ptr find (std::size_t index) const

  Finds the element at the specified index.
- Node_ptr insert (std::size_t index, const T &elem)

  Inserts the element at the specified index.
- Node_ptr remove (std::size_t index)

  Removes the element at the specified index.
- T & at (std::size_t index)

  Gets the element at the specified index.
- T at (std::size_t index) const

  Gets the element at the specified index.
- void clear ()

  Clears the container.
- bool empty () const

  Check whether the container is empty.
- std::size_t size () const

  Returns the size of the container.

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

- BaseList ()=default

  Default constructor.
- BaseList (std::initializer_list< T > init_list)

  Constructs the container with the contents of the initializer list.
- BaseList (const BaseList &rhs)

  Copy constructor.
- BaseList & operator= (const BaseList &rhs)

  Copy assignment operator.
- BaseList (BaseList &&rhs) noexcept

  Move constructor.
- BaseList & operator= (BaseList &&rhs) noexcept

  Move assignment operator.
- ~BaseList ()

  Destructor.
- bool empty () const

  Check whether the container is empty.
- std::size_t size () const

  Returns the size of the container.

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)

  Internal method to search for the element in the container.
- Node_ptr internal_find (std::size_t index)

  Internal method to find the element at the specified index.

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

- void init_first_element (const T &elem)

  Initializes the first element of the container.
- void clean_up ()

  Frees all elements in the container.
- void copy_data (const BaseList &rhs)

  Copies data from another container.
- void push_back (const T &elem)

  Pushes the element to the back of the container.
- void push_front (const T &elem)

  Pushes the element to the front of the container.
- T & back () const

  Returns the reference to the element at the back of the container.
- T & front () const

  Returns the reference to the element at the front of the container.
- void pop_front ()

  Removes the element at the back of the container.
- void pop_back ()

  Removes the element at the front of the container.

Protected Attributes

- Node_ptr m_head

  The head of the list.
- std::size_t m_size

  The size of the list.
- Node_ptr m_tail

  The tail of the list.

Protected Attributes inherited from core::BaseList< T >

- Node_ptr m_head {nullptr}

  The head of the list.
- Node_ptr m_tail {nullptr}

  The tail of the list.
- std::size_t m_size {}

  The size of the list.

6.8.1 Detailed Description

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

The doubly linked list container.

Template Parameters

T	the type of the elements

Definition at line 16 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 18 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 21 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 19 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 20 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 )
```

Gets the element at the specified index.

Parameters

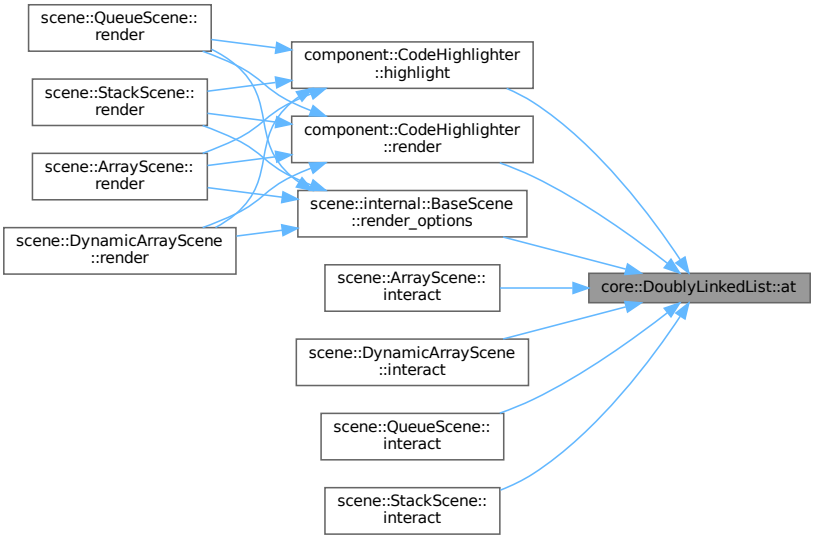
index	The index of the element

Returns

T& The reference to the element

Definition at line 218 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
```

Gets the element at the specified index.

Parameters

index	The index of the element

Returns

T The copy of the element

Definition at line 223 of file doubly_linked_list.hpp.

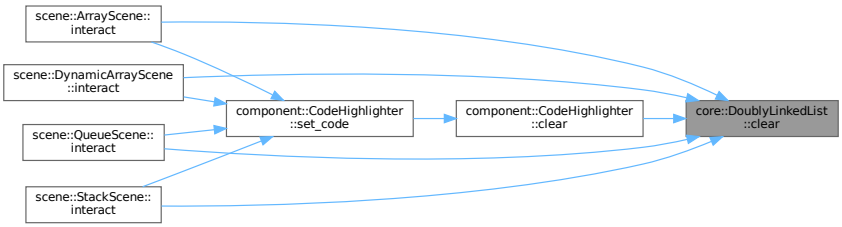
6.8.3.3 clear()

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

Clears the container.

Definition at line 228 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
```

Check whether the container is empty.

Return values

true	The container is empty
false	The container is not empty

Definition at line 63 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 )
```

Finds the element at the specified index.

Parameters

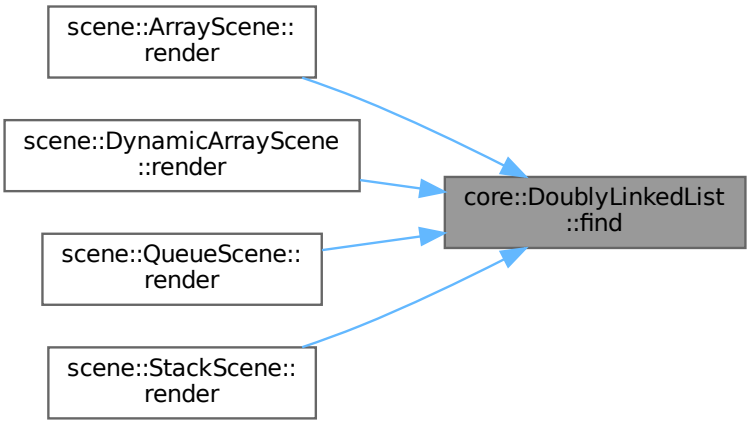
index	The index of the element

Returns

The pointer to the element if found, nullptr otherwise

Definition at line 148 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
```

Finds the element at the specified index.

Parameters

index	The index of the element

Returns

The const pointer to the element if found, nullptr otherwise

Definition at line 160 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 )
```

Inserts the element at the specified index.

Parameters

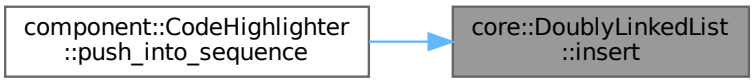
index	The index of the element
elem	The element to insert

Returns

The pointer to the inserted element

Definition at line 166 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]
```

Internal method to find the element at the specified index.

Parameters

index	The index of the element

Returns

The pointer to the element if found, nullptr otherwise

Definition at line 128 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]
```

Internal method to search for the element in the container.

Parameters

elem	The element to search for

Returns

The pointer to the element if found, nullptr otherwise

Definition at line 112 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 )
```

Removes the element at the specified index.

Parameters

index	The index of the element

Returns

The pointer to the removed element

Definition at line 189 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 )
```

Searches for the element in the container.

Parameters

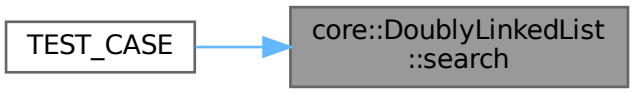
elem	The element to search for

Returns

The pointer to the element if found, nullptr otherwise

Definition at line 142 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
```

Searches for the element in the container.

Parameters

elem	The element to search for

Returns

The const pointer to the element if found, nullptr otherwise

Definition at line 154 of file doubly_linked_list.hpp.

6.8.3.13 size()

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

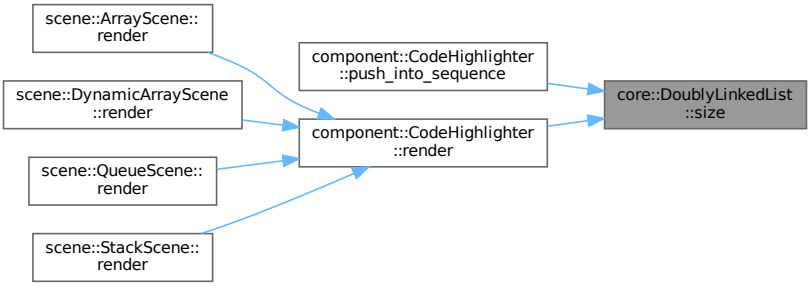
Returns the size of the container.

Returns

The size of the container

Definition at line 69 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]
```

The head of the list.

Definition at line 87 of file base_list.hpp.

6.8.4.2 m_size

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

The size of the list.

Definition at line 97 of file base_list.hpp.

6.8.4.3 m_tail

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

The tail of the list.

Definition at line 92 of file base_list.hpp.

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

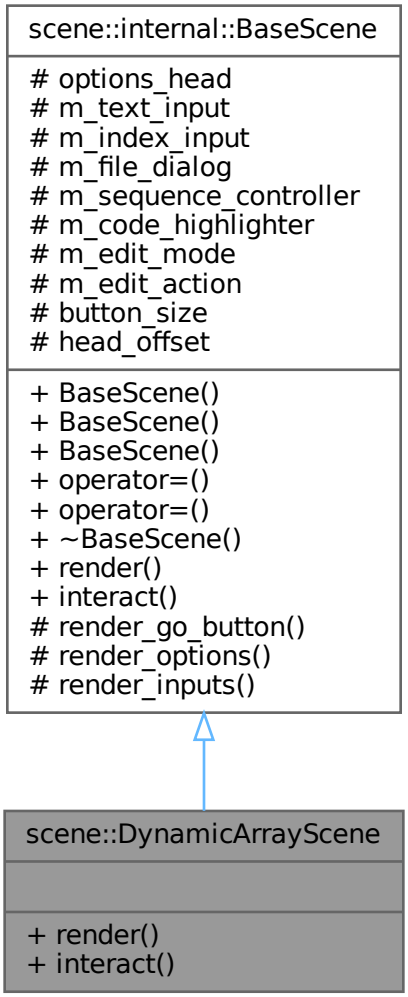
- src/core/doubly_linked_list.hpp

6.9 scene::DynamicArrayScene Class Reference

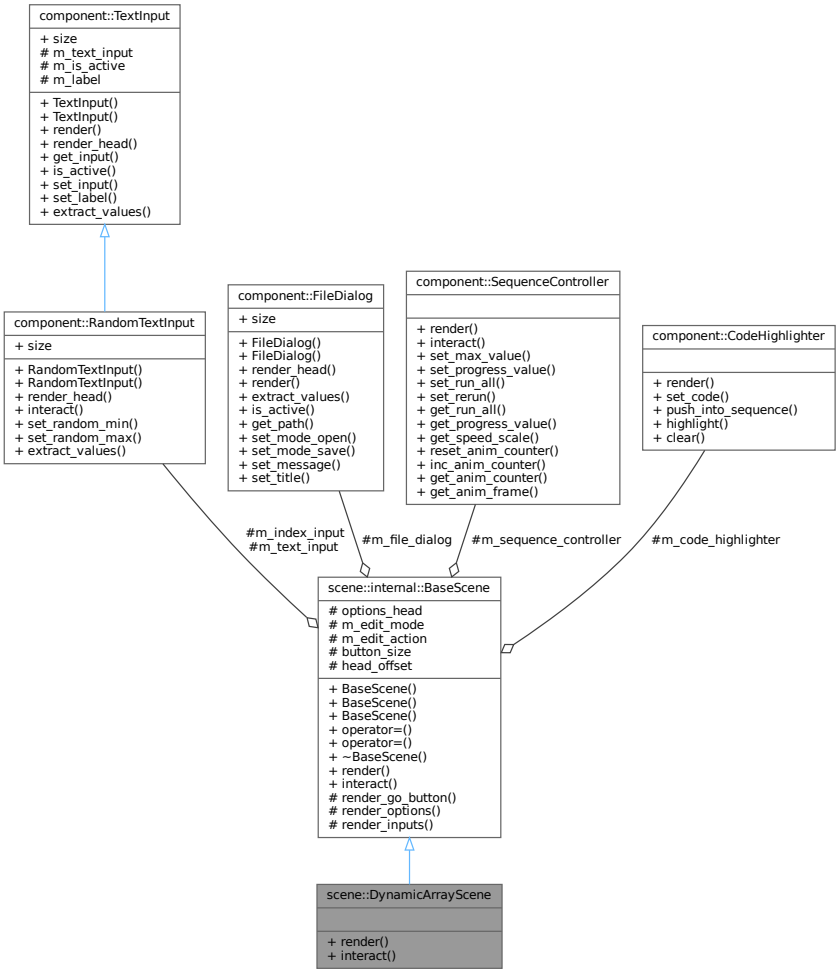
The dynamic array scene.

`#include <dynamic_array_scene.hpp>`

Inheritance diagram for scene::DynamicArrayScene:



Collaboration diagram for scene::DynamicArrayScene:



Public Member Functions

- void render () override

  Renders the scene.
- void interact () override

  Interacts with the scene.

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

- BaseScene ()=default

  Construct a new BaseScene object.
- BaseScene (const BaseScene &)=delete

  Copy constructor (deleted)
- BaseScene (BaseScene &&)=delete

  Move constructor (deleted)
- BaseScene & operator= (const BaseScene &)=delete

  Copy assignment (deleted)

- BaseScene & operator= (BaseScene &&)=delete

  Move assignment (deleted)
- virtual ~BaseScene ()=default

  Destroy the BaseScene object.
- virtual void render ()

  Renders the scene.
- virtual void interact ()

  Interacts with the scene.

Additional Inherited Members

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

- virtual bool render_go_button () const

  Renders the go button.
- virtual void render_options (SceneOptions &scene_config)

  Renders the options.
- virtual void render_inputs ()

  Renders the inputs.

Protected Attributes inherited from scene::internal::BaseScene

- float options_head {}

  The head of the options.
- component::RandomTextInput m_text_input {"value"}

  The text input for the value.
- component::RandomTextInput m_index_input {"index"}

  The text input for the index.
- component::FileDialog m_file_dialog

  The file dialog.
- component::SequenceController m_sequence_controller

  The sequence controller.
- component::CodeHighlighter m_code_highlighter

  The code highlighter.
- bool m_edit_mode {}

  Whether the edit mode is enabled.
- bool m_edit_action {}

  Whether the edit action is enabled.

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

- static constexpr Vector2 button_size {200, 50}

  The size of the buttons.
- static constexpr int head_offset = 20

  The offset of the widgets.

6.9.1 Detailed Description

The dynamic array scene.

Definition at line 21 of file dynamic_array_scene.hpp.

6.9.2 Member Function Documentation

6.9.2.1 interact()

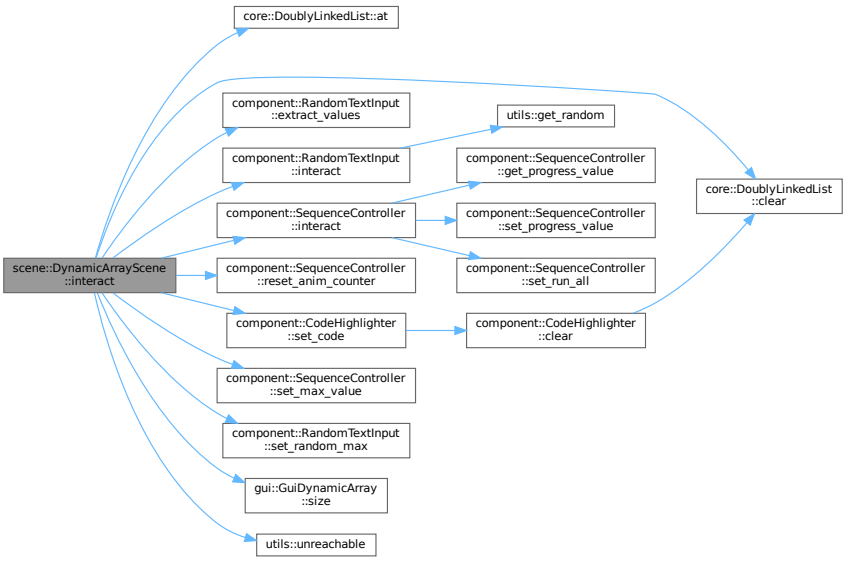
```
void scene::DynamicArrayScene::interact ( )  [override], [virtual]
```

Interacts with the scene.

Reimplemented from scene::internal::BaseScene.

Definition at line 99 of file dynamic_array_scene.cpp.

Here is the call graph for this function:



6.9.2.2 render()

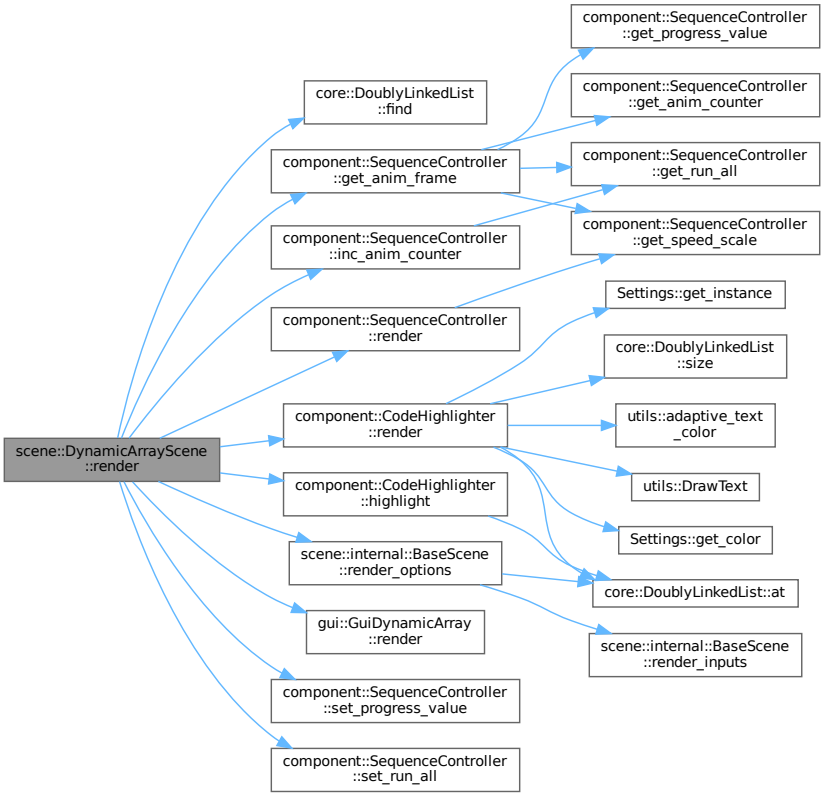
```
void scene::DynamicArrayScene::render ( )  [override], [virtual]
```

Renders the scene.

Reimplemented from scene::internal::BaseScene.

Definition at line 79 of file dynamic_array_scene.cpp.

Here is the call graph for this function:



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

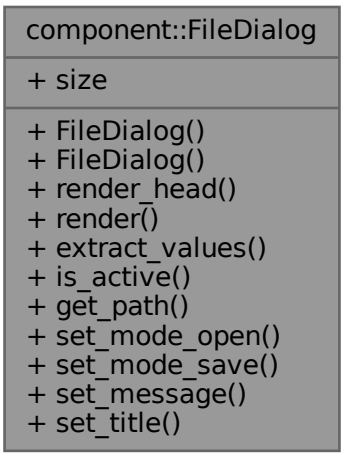
- src/scene/dynamic_array_scene.hpp
- src/scene/dynamic_array_scene.cpp

6.10 component::FileDialog Class Reference

File Dialog for opening and saving files.

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

Collaboration diagram for component::FileDialog:



Public Member Functions

- FileDialog ()

  Constructs a FileDialog object.
- FileDialog (int mode, const char ∗title, const char ∗message)

  Constructs a FileDialog object.
- int render_head (float &options_head, float head_offset)

  Renders the file dialog, updates the head position with offset.
- int render (float x, float y)

  Renders the file dialog.
- core::Deque< int > extract_values ()

  Extracts the values from the file.
- bool is_active () const

  Checks if the file dialog is active.
- std::string get_path ()

  Gets the path of the file.
- void set_mode_open ()

  Sets the mode of the file dialog to open.
- void set_mode_save ()

  Sets the mode of the file dialog to save.
- void set_message (const char ∗message)

  Sets the message of the file dialog.
- void set_title (const char ∗title)

  Sets the title of the file dialog.

Static Public Attributes

- static constexpr Vector2 size {200, 50}

  The size of the file dialog.

6.10.1 Detailed Description

File Dialog for opening and saving files.

Definition at line 17 of file file_dialog.hpp.

6.10.2 Constructor & Destructor Documentation

6.10.2.1 FileDialog() [1/2]

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

Constructs a FileDialog object.

Definition at line 16 of file file_dialog.cpp.

6.10.2.2 FileDialog() [2/2]

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

Constructs a FileDialog object.

Parameters

mode	the mode of the file dialog (open or save)
title	the title of the file dialog
message	the message of the file dialog

Definition at line 13 of file file_dialog.cpp.

6.10.3 Member Function Documentation

6.10.3.1 extract_values()

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

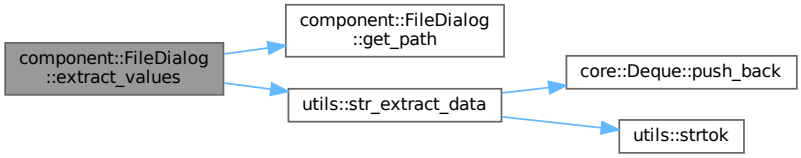
Extracts the values from the file.

Returns

the values from the file

Definition at line 49 of file file_dialog.cpp.

Here is the call graph for this function:



6.10.3.2 get_path()

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

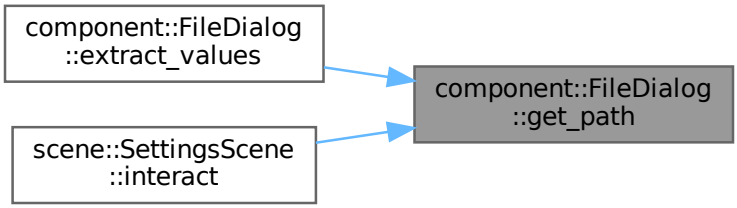
Gets the path of the file.

Returns

the path of the file

Definition at line 66 of file file_dialog.cpp.

Here is the caller graph for this function:



6.10.3.3 is_active()

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

Checks if the file dialog is active.

Return values

true	The file dialog is active
false	The file dialog is not active

Definition at line 57 of file file_dialog.cpp.

6.10.3.4 render()

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

Renders the file dialog.

Parameters

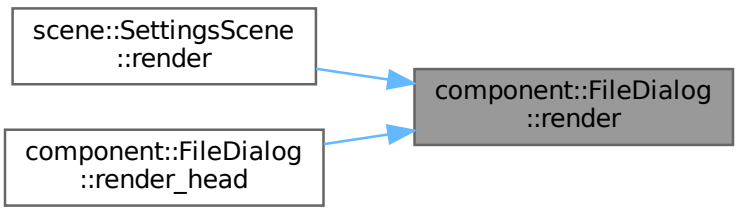
x	the x position of the file dialog
y	the y position of the file dialog

Return values

-1	The file dialog is not active
0	A file was not selected
1	A file was selected

Definition at line 18 of file file_dialog.cpp.

Here is the caller graph for this function:



6.10.3.5 render_head()

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

Renders the file dialog, updates the head position with offset.

Parameters

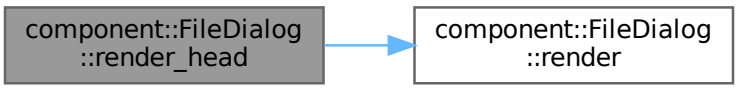
options_head	the head position of the options
head_offset	the offset of the head position

Return values

-1	The file dialog is not active
0	A file was not selected
1	A file was selected

Definition at line 43 of file file_dialog.cpp.

Here is the call graph for this function:



6.10.3.6 set_message()

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

Sets the message of the file dialog.

Parameters

message	the message of the file dialog

Definition at line 63 of file file_dialog.cpp.

6.10.3.7 set_mode_open()

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

Sets the mode of the file dialog to open.

Definition at line 59 of file file_dialog.cpp.

6.10.3.8 set_mode_save()

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

Sets the mode of the file dialog to save.

Definition at line 61 of file file_dialog.cpp.

6.10.3.9 set_title()

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

Sets the title of the file dialog.

Parameters

title	the title of the file dialog

Definition at line 65 of file file_dialog.cpp.

6.10.4 Member Data Documentation

6.10.4.1 size

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

The size of the file dialog.

Definition at line 22 of file file_dialog.hpp.

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

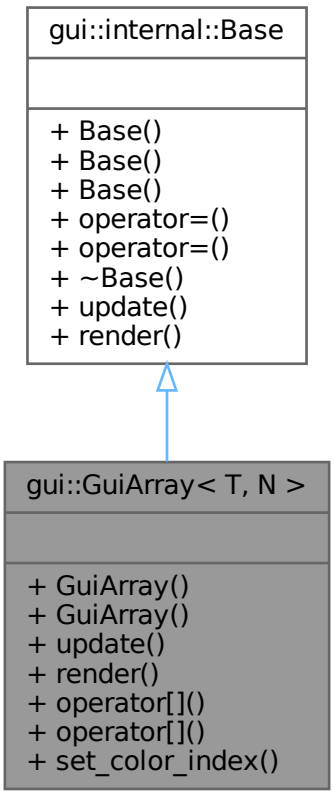
- src/component/file_dialog.hpp
- src/component/file_dialog.cpp

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

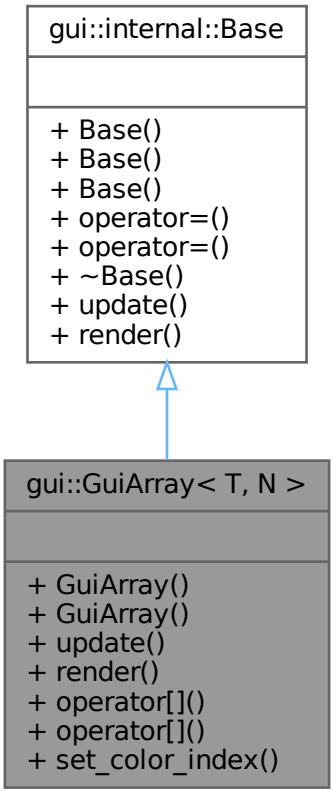
The GUI array container.

`#include <array_gui.hpp>`

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



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



Public Member Functions

- GuiArray ()

  Constructs a GUI array with the specified number of elements.
- GuiArray (std::array< GuiElement< T >, N > &&init_list)

  Constructs a GUI array with the specified initializer list.
- void update () override

  Updates the GUI array.
- void render () override

  Renders the GUI array.
- T & operator[ ] (std::size_t idx)

  Returns the reference to the element at the specified index.
- T operator[ ] (std::size_t idx) const

  Returns the value to the element at the specified index.
- void set_color_index (std::size_t idx, int color_index)

  Set the color index object.

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

- Base ()=default
  Constructs a Base object.
- Base (const Base &)=default
  Copy constructor.
- Base (Base &&)=default
  Move constructor.
- Base & operator= (const Base &)=default
  Copy assignment operator.
- Base & operator= (Base &&)=default
  Move assignment operator.
- virtual ~Base ()=default
  Destructor.
- virtual void update ()=0
  Updates the GUI.
- virtual void render ()=0
  Renders the GUI.

6.11.1 Detailed Description

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

The GUI array container.

Template Parameters

T	the type of the elements
N	the number of elements

Definition at line 22 of file array_gui.hpp.

6.11.2 Constructor & Destructor Documentation

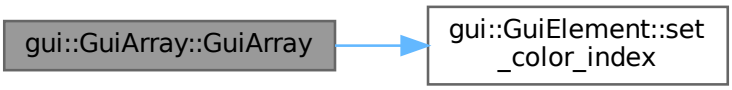
6.11.2.1 GuiArray() [1/2]

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

Constructs a GUI array with the specified number of elements.

Definition at line 89 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 )
```

Constructs a GUI array with the specified initializer list.

Parameters

init_list	The initializer list

Definition at line 97 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 )
```

Returns the reference to the element at the specified index.

Parameters

idx	The index of the element

Returns

The reference to the element at the specified index

Definition at line 123 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
```

Returns the value to the element at the specified index.

Parameters

idx	The index of the element

Returns

The value to the element at the specified index

Definition at line 128 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]
```

Renders the GUI array.

Implements gui::internal::Base.

Definition at line 104 of file array_gui.hpp.

6.11.3.4 set_color_index()

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

Set the color index object.

Parameters

idx	The index of the element to color
color_index	The index of the color in the settings

Definition at line 133 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]
```

Updates the GUI array.

Implements gui::internal::Base.

Definition at line 113 of file array_gui.hpp.

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

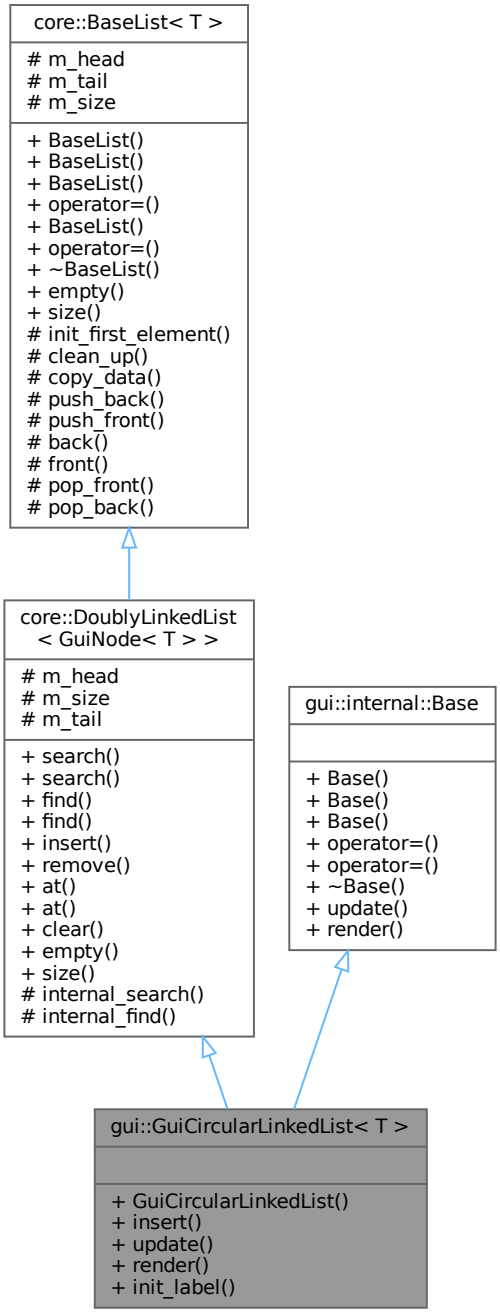
- src/gui/array_gui.hpp

6.12 gui::GuiCircularLinkedList< T > Class Template Reference

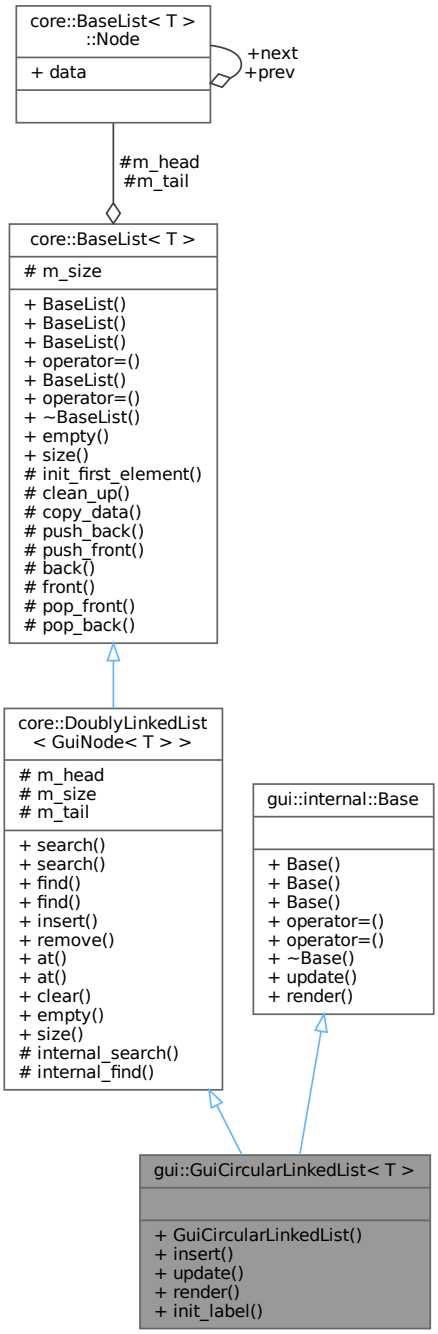
The GUI circular linked list container.

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

Inheritance diagram for gui::GuiCircularLinkedList< T >:



Collaboration diagram for gui::GuiCircularLinkedList< T >:



Public Member Functions

- GuiCircularLinkedList (std::initializer_list< GuiNode< T > > init_list)

  Construct a new Gui Circular Linked List object.
- void insert (std::size_t index, const T &elem)

  Inserts an element at the specified index.
- void update () override

Updates the GUI circular linked list.

- void render () override

  Renders the GUI circular linked list.

- void init_label ()

  Initializes the labels.

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

- Node_ptr search (const GuiNode< T > &elem)

  Searches for the element in the container.

- cNode_ptr search (const GuiNode< T > &elem) const

  Searches for the element in the container.

- Node_ptr find (std::size_t index)

  Finds the element at the specified index.

- cNode_ptr find (std::size_t index) const

  Finds the element at the specified index.

- Node_ptr insert (std::size_t index, const GuiNode< T > &elem)

  Inserts the element at the specified index.

- Node_ptr remove (std::size_t index)

  Removes the element at the specified index.

- GuiNode< T > & at (std::size_t index)

  Gets the element at the specified index.

- GuiNode< T > at (std::size_t index) const

  Gets the element at the specified index.

- void clear ()

  Clears the container.

- bool empty () const

  Check whether the container is empty.

- std::size_t size () const

  Returns the size of the container.

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

- BaseList ()=default

  Default constructor.

- BaseList (std::initializer_list< T > init_list)

  Constructs the container with the contents of the initializer list.

- BaseList (const BaseList &rhs)

  Copy constructor.

- BaseList & operator= (const BaseList &rhs)

  Copy assignment operator.

- BaseList (BaseList &&rhs) noexcept

  Move constructor.

- BaseList & operator= (BaseList &&rhs) noexcept

  Move assignment operator.

- ~BaseList ()

  Destructor.

- bool empty () const

  Check whether the container is empty.

- std::size_t size () const

  Returns the size of the container.

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

- Base ()=default

  Constructs a Base object.
- Base (const Base &)=default

  Copy constructor.
- Base (Base &&)=default

  Move constructor.
- Base & operator= (const Base &)=default

  Copy assignment operator.
- Base & operator= (Base &&)=default

  Move assignment operator.
- virtual ~Base ()=default

  Destructor.
- virtual void update ()=0

  Updates the GUI.
- virtual void render ()=0

  Renders the GUI.

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)

  Internal method to search for the element in the container.
- Node_ptr internal_find (std::size_t index)

  Internal method to find the element at the specified index.

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

- void init_first_element (const T &elem)

  Initializes the first element of the container.
- void clean_up ()

  Frees all elements in the container.
- void copy_data (const BaseList &rhs)

  Copies data from another container.
- void push_back (const T &elem)

  Pushes the element to the back of the container.
- void push_front (const T &elem)

  Pushes the element to the front of the container.
- T & back () const

  Returns the reference to the element at the back of the container.
- T & front () const

  Returns the reference to the element at the front of the container.
- void pop_front ()

  Removes the element at the back of the container.
- void pop_back ()

  Removes the element at the front of the container.

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

- Node_ptr m_head

  The head of the list.
- std::size_t m_size

  The size of the list.
- Node_ptr m_tail

  The tail of the list.

Protected Attributes inherited from core::BaseList< T >

- Node_ptr m_head {nullptr}

  The head of the list.
- Node_ptr m_tail {nullptr}

  The tail of the list.
- std::size_t m_size {}

  The size of the list.

6.12.1 Detailed Description

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

The GUI circular linked list container.

Template Parameters

T	the type of the elements

Definition at line 24 of file circular_linked_list_gui.hpp.

6.12.2 Constructor & Destructor Documentation

6.12.2.1 GuiCircularLinkedList()

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

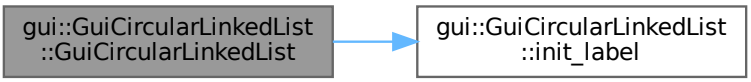
Construct a new Gui Circular Linked List object.

Parameters

init_list	The initializer list

Definition at line 103 of file circular_linked_list_gui.hpp.

Here is the call graph for this function:



6.12.3 Member Function Documentation

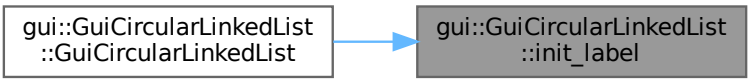
6.12.3.1 init_label()

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

Initializes the labels.

Definition at line 88 of file circular_linked_list_gui.hpp.

Here is the caller graph for this function:



6.12.3.2 insert()

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

Inserts an element at the specified index.

Parameters

index	The index
elem	The element

Definition at line 110 of file circular_linked_list_gui.hpp.

6.12.3.3 render()

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

Renders the GUI circular linked list.

Implements gui::internal::Base.

Definition at line 167 of file circular_linked_list_gui.hpp.

6.12.3.4 update()

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

Updates the GUI circular linked list.

Implements gui::internal::Base.

Definition at line 181 of file circular_linked_list_gui.hpp.

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

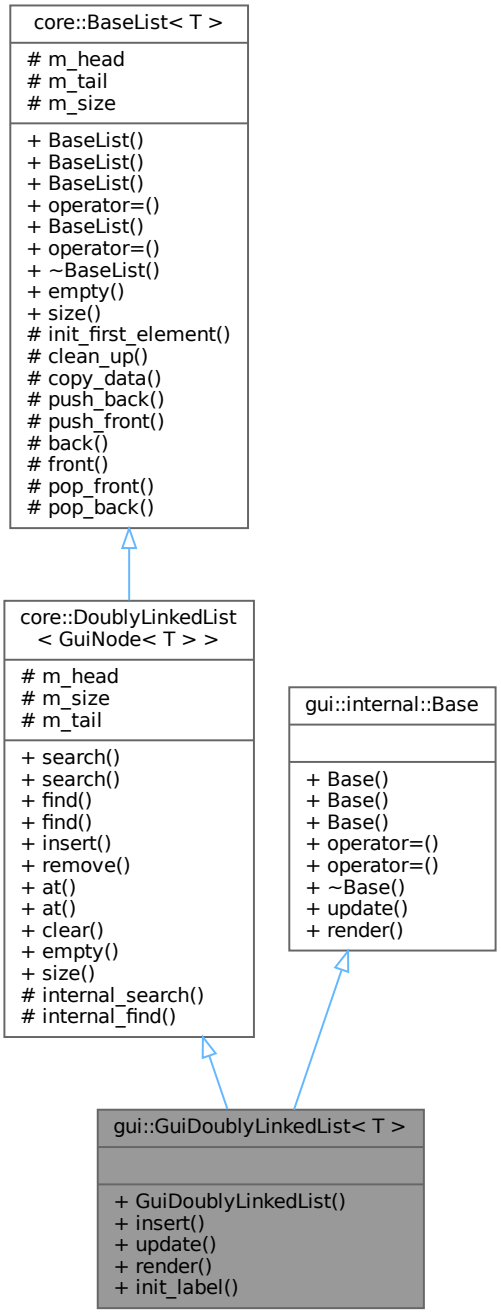
- src/gui/circular_linked_list_gui.hpp

6.13 gui::GuiDoublyLinkedList< T > Class Template Reference

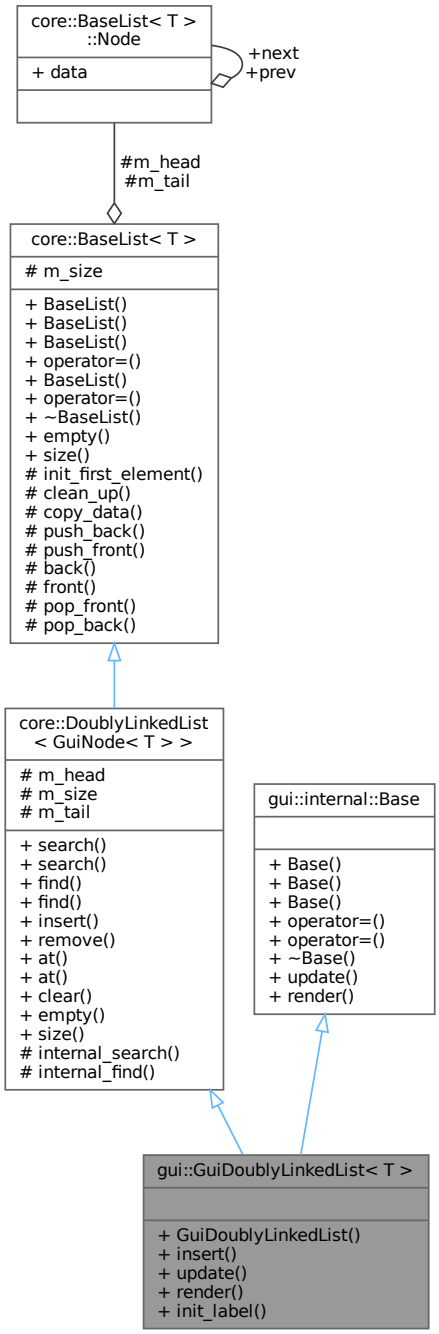
The GUI doubly linked list container.

`#include <doubly_linked_list_gui.hpp>`

Inheritance diagram for gui::GuiDoublyLinkedList< T >:



Collaboration diagram for gui::GuiDoublyLinkedList< T >:



Public Member Functions

- GuiDoublyLinkedList (std::initializer_list< GuiNode< T > > init_list)

  Construct a new Gui Doubly Linked List object.
- void insert (std::size_t index, const T &elem)

  Inserts an element at the specified index.
- void update () override

Updates the GUI doubly linked list.

- void render () override

  Renders the GUI doubly linked list.

- void init_label ()

  Initializes the labels.

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

- Node_ptr search (const GuiNode< T > &elem)

  Searches for the element in the container.

- cNode_ptr search (const GuiNode< T > &elem) const

  Searches for the element in the container.

- Node_ptr find (std::size_t index)

  Finds the element at the specified index.

- cNode_ptr find (std::size_t index) const

  Finds the element at the specified index.

- Node_ptr insert (std::size_t index, const GuiNode< T > &elem)

  Inserts the element at the specified index.

- Node_ptr remove (std::size_t index)

  Removes the element at the specified index.

- GuiNode< T > & at (std::size_t index)

  Gets the element at the specified index.

- GuiNode< T > at (std::size_t index) const

  Gets the element at the specified index.

- void clear ()

  Clears the container.

- bool empty () const

  Check whether the container is empty.

- std::size_t size () const

  Returns the size of the container.

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

- BaseList ()=default

  Default constructor.

- BaseList (std::initializer_list< T > init_list)

  Constructs the container with the contents of the initializer list.

- BaseList (const BaseList &rhs)

  Copy constructor.

- BaseList & operator= (const BaseList &rhs)

  Copy assignment operator.

- BaseList (BaseList &&rhs) noexcept

  Move constructor.

- BaseList & operator= (BaseList &&rhs) noexcept

  Move assignment operator.

- ~BaseList ()

  Destructor.

- bool empty () const

  Check whether the container is empty.

- std::size_t size () const

  Returns the size of the container.

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

- Base ()=default

  Constructs a Base object.
- Base (const Base &)=default

  Copy constructor.
- Base (Base &&)=default

  Move constructor.
- Base & operator= (const Base &)=default

  Copy assignment operator.
- Base & operator= (Base &&)=default

  Move assignment operator.
- virtual ~Base ()=default

  Destructor.
- virtual void update ()=0

  Updates the GUI.
- virtual void render ()=0

  Renders the GUI.

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)

  Internal method to search for the element in the container.
- Node_ptr internal_find (std::size_t index)

  Internal method to find the element at the specified index.

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

- void init_first_element (const T &elem)

  Initializes the first element of the container.
- void clean_up ()

  Frees all elements in the container.
- void copy_data (const BaseList &rhs)

  Copies data from another container.
- void push_back (const T &elem)

  Pushes the element to the back of the container.
- void push_front (const T &elem)

  Pushes the element to the front of the container.
- T & back () const

  Returns the reference to the element at the back of the container.
- T & front () const

  Returns the reference to the element at the front of the container.
- void pop_front ()

  Removes the element at the back of the container.
- void pop_back ()

  Removes the element at the front of the container.

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

- Node_ptr m_head

  The head of the list.
- std::size_t m_size

  The size of the list.
- Node_ptr m_tail

  The tail of the list.

Protected Attributes inherited from core::BaseList< T >

- Node_ptr m_head {nullptr}

  The head of the list.
- Node_ptr m_tail {nullptr}

  The tail of the list.
- std::size_t m_size {}

  The size of the list.

6.13.1 Detailed Description

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

The GUI doubly linked list container.

Template Parameters

T	the type of the elements

Definition at line 22 of file doubly_linked_list_gui.hpp.

6.13.2 Constructor & Destructor Documentation

6.13.2.1 GuiDoublyLinkedList()

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

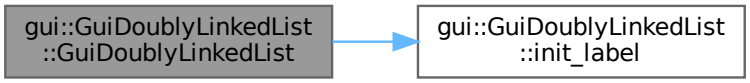
Construct a new Gui Doubly Linked List object.

Parameters

init_list	The initializer list

Definition at line 96 of file doubly_linked_list_gui.hpp.

Here is the call graph for this function:



6.13.3 Member Function Documentation

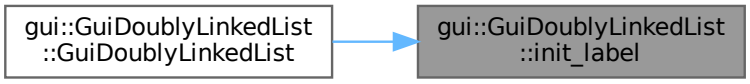
6.13.3.1 init_label()

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

Initializes the labels.

Definition at line 81 of file doubly_linked_list_gui.hpp.

Here is the caller graph for this function:



6.13.3.2 insert()

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

Inserts an element at the specified index.

Parameters

index	The index
elem	The element

Definition at line 103 of file doubly_linked_list_gui.hpp.

6.13.3.3 render()

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

Renders the GUI doubly linked list.

Implements gui::internal::Base.

Definition at line 139 of file doubly_linked_list_gui.hpp.

6.13.3.4 update()

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

Updates the GUI doubly linked list.

Implements gui::internal::Base.

Definition at line 152 of file doubly_linked_list_gui.hpp.

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

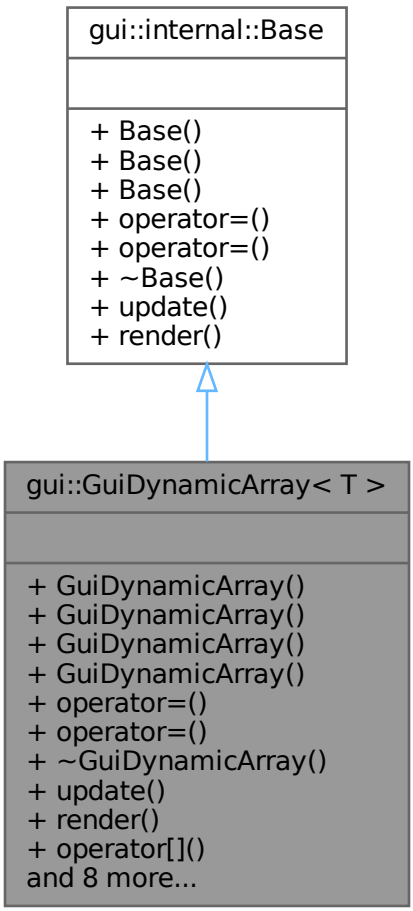
- src/gui/doubly_linked_list_gui.hpp

6.14 gui::GuiDynamicArray< T > Class Template Reference

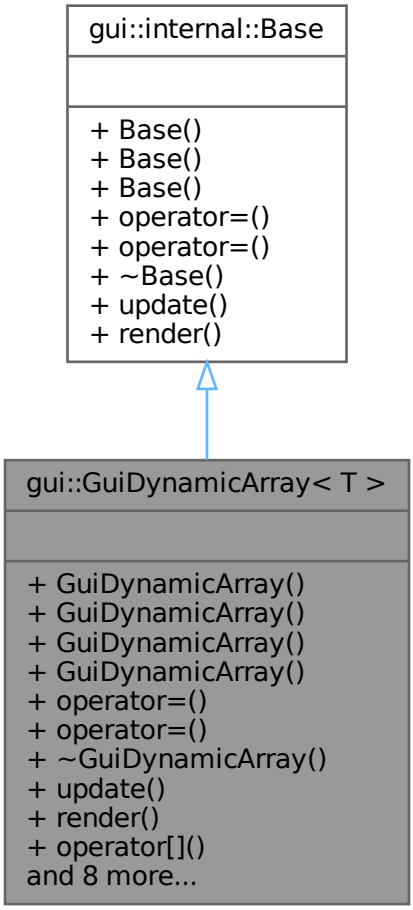
The GUI dynamic array container.

`#include <dynamic_array_gui.hpp>`

Inheritance diagram for gui::GuiDynamicArray< T >:



Collaboration diagram for gui::GuiDynamicArray< T >:



Public Member Functions

- GuiDynamicArray ()

  Constructs a GUI dynamic array.
- GuiDynamicArray (std::initializer_list< T > init_list)

  Constructs a GUI dynamic array with the specified initializer list.
- GuiDynamicArray (const GuiDynamicArray &other)

  Constructs a GUI dynamic array by coping another GUI dynamic array.
- GuiDynamicArray (GuiDynamicArray &&other) noexcept

  Constructs a GUI dynamic array by moving another GUI dynamic array.
- GuiDynamicArray & operator= (const GuiDynamicArray &other)

  Assigns another GUI dynamic array to this GUI dynamic array by copying.
- GuiDynamicArray & operator= (GuiDynamicArray &&other) noexcept

  Assigns another GUI dynamic array to this GUI dynamic array by moving.
- ~GuiDynamicArray () override

  Destructs the GUI dynamic array.

- void update () override

  Updates the GUI dynamic array.
- void render () override

  Renders the GUI dynamic array.
- T & operator[ ] (std::size_t idx)

  Returns the reference to the element at the specified index.
- T operator[ ] (std::size_t idx) const

  Returns the value to the element at the specified index.
- void set_color_index (std::size_t idx, int color_index)

  Set the color index object.
- void reserve (std::size_t capacity)

  Reserves the array with specified capacity.
- void shrink_to_fit ()

  Resizes the array capacity to the current size.
- std::size_t capacity () const

  Returns the capacity of the array.
- std::size_t size () const

  Returns the size of the array.
- void push (const T &value)

  Pushes the value to the end of the array.
- void pop ()

  Pops the value from the end of the array.

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

- Base ()=default

  Constructs a Base object.
- Base (const Base &)=default

  Copy constructor.
- Base (Base &&)=default

  Move constructor.
- Base & operator= (const Base &)=default

  Copy assignment operator.
- Base & operator= (Base &&)=default

  Move assignment operator.
- virtual ~Base ()=default

  Destructor.
- virtual void update ()=0

  Updates the GUI.
- virtual void render ()=0

  Renders the GUI.

6.14.1 Detailed Description

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

The GUI dynamic array container.

Template Parameters

T	the type of the elements

Definition at line 22 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
```

Constructs a GUI dynamic array.

Definition at line 208 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 )
```

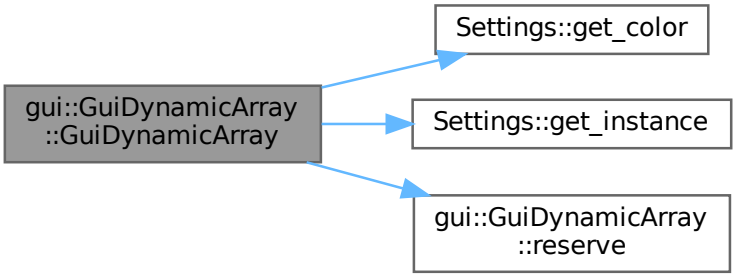
Constructs a GUI dynamic array with the specified initializer list.

Parameters

init_list	The initializer list

Definition at line 215 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 )
```

Constructs a GUI dynamic array by coping another GUI dynamic array.

Parameters

other	The other GUI dynamic array

Definition at line 226 of file dynamic_array_gui.hpp.

6.14.2.4 GuiDynamicArray() [4/4]

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

Constructs a GUI dynamic array by moving another GUI dynamic array.

Parameters

other	The other GUI dynamic array

Definition at line 236 of file dynamic_array_gui.hpp.

6.14.2.5 ~GuiDynamicArray()

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

Destructs the GUI dynamic array.

Definition at line 274 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
```

Returns the capacity of the array.

Returns

The capacity of the array

Definition at line 318 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 )
```

Assigns another GUI dynamic array to this GUI dynamic array by copying.

Parameters

other	The other GUI dynamic array

Returns

This GUI dynamic array

Definition at line 244 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]
```

Assigns another GUI dynamic array to this GUI dynamic array by moving.

Parameters

other	The other GUI dynamic array

Returns

This GUI dynamic array

Definition at line 260 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 )
```

Returns the reference to the element at the specified index.

Parameters

idx	The index of the element

Returns

The reference to the element at the specified index

Definition at line 303 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
```

Returns the value to the element at the specified index.

Parameters

idx	The index of the element

Returns

The value to the element at the specified index

Definition at line 308 of file dynamic_array_gui.hpp.

6.14.3.6 pop()

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

Pops the value from the end of the array.

Definition at line 339 of file dynamic_array_gui.hpp.

6.14.3.7 push()

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

Pushes the value to the end of the array.

Parameters

value	The value to push

Definition at line 328 of file dynamic_array_gui.hpp.

6.14.3.8 render()

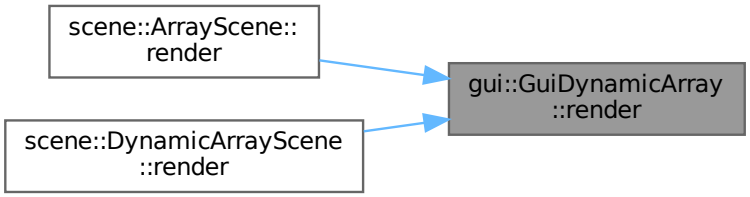
```
template<typename T >
void gui::GuiDynamicArray< T >::render  [override], [virtual]
```

Renders the GUI dynamic array.

Implements gui::internal::Base.

Definition at line 282 of file dynamic_array_gui.hpp.

Here is the caller graph for this function:



6.14.3.9 reserve()

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

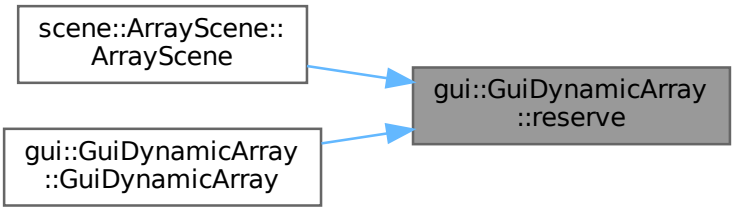
Reserves the array with specified capacity.

Parameters

capacity	The capacity

Definition at line 165 of file dynamic_array_gui.hpp.

Here is the caller graph for this function:



6.14.3.10 set_color_index()

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

Set the color index object.

Parameters

idx	The index of the element
color_index	The index of the color in the settings

Definition at line 313 of file dynamic_array_gui.hpp.

6.14.3.11 shrink_to_fit()

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

Resizes the array capacity to the current size.

Definition at line 188 of file dynamic_array_gui.hpp.

6.14.3.12 size()

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

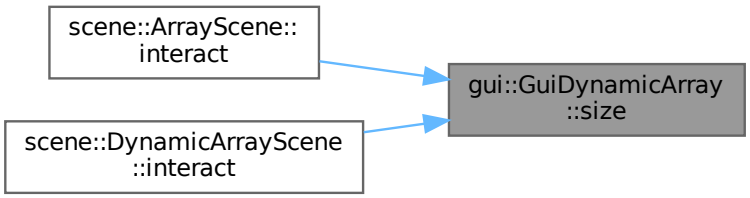
Returns the size of the array.

Returns

The size of the array

Definition at line 323 of file dynamic_array_gui.hpp.

Here is the caller graph for this function:



6.14.3.13 update()

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

Updates the GUI dynamic array.

Implements gui::internal::Base.

Definition at line 293 of file dynamic_array_gui.hpp.

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

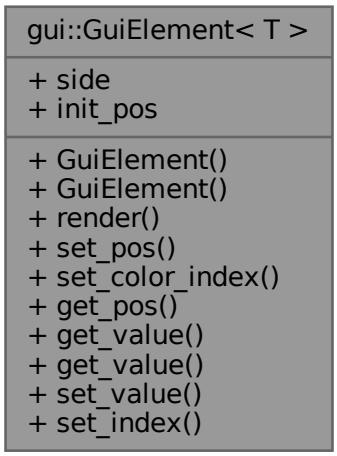
- src/gui/dynamic_array_gui.hpp

6.15 gui::GuiElement< T > Class Template Reference

The GUI element (used in arrays)

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

Collaboration diagram for gui::GuiElement< T >:



Public Member Functions

- GuiElement ()=default

  Construct a new GuiElement object.
- GuiElement (const T &value, std::size_t index)

  Construct a new GuiElement object.
- void render ()

  Renders the element.
- void set_pos (Vector2 pos)

  Sets the position of the element.
- void set_color_index (int color_index)

  Sets the color index of the element.
- Vector2 get_pos () const

  Returns the position of the element.
- T & get_value ()

  Returns the reference to the value of the element.
- T get_value () const

  Returns the value of the element.
- void set_value (const T &value)

  Set the value of the element.
- void set_index (std::size_t index)

  Set the index of the element.

Static Public Attributes

- static constexpr int side = 20

  The side length of the element.
- static constexpr Vector2 init_pos

  The initial position of the element.

6.15.1 Detailed Description

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

The GUI element (used in arrays)

Template Parameters

T	the type of the element

Definition at line 22 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]
```

Construct a new GuiElement object.

6.15.2.2 GuiElement() [2/2]

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

Construct a new GuiElement object.

Parameters

value	The value of the element
index	The index of the element

Definition at line 126 of file element_gui.hpp.

6.15.3 Member Function Documentation

6.15.3.1 get_pos()

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

Returns the position of the element.

Returns

The position of the element

6.15.3.2 get_value() [1/2]

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

Returns the reference to the value of the element.

Returns

The reference to the value of the element

Definition at line 176 of file element_gui.hpp.

6.15.3.3 get_value() [2/2]

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

Returns the value of the element.

Returns

The value of the element

Definition at line 181 of file element_gui.hpp.

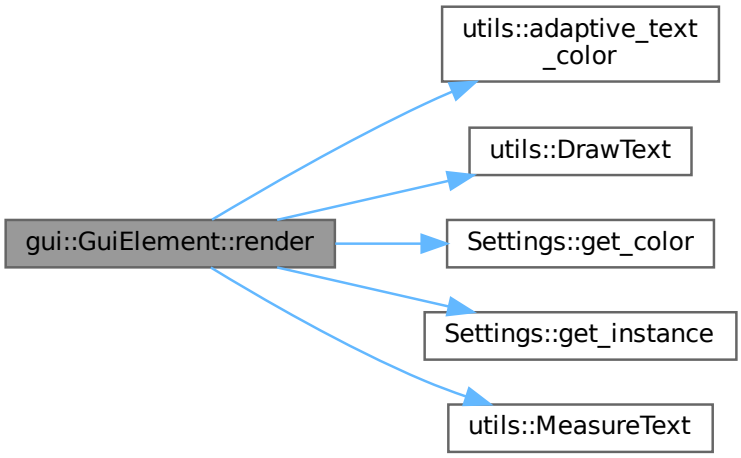
6.15.3.4 render()

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

Renders the element.

Definition at line 130 of file element_gui.hpp.

Here is the call graph for this function:



6.15.3.5 set_color_index()

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

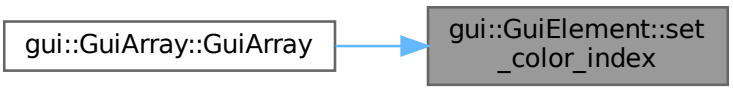
Sets the color index of the element.

Parameters

color_index	The index of the color in the settings

Definition at line 171 of file element_gui.hpp.

Here is the caller graph for this function:



6.15.3.6 set_index()

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

Set the index of the element.

Parameters

index	The index of the element

Definition at line 191 of file element_gui.hpp.

6.15.3.7 set_pos()

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

Sets the position of the element.

Parameters

pos	The position of the element

Definition at line 166 of file element_gui.hpp.

6.15.3.8 set_value()

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

Set the value of the element.

Parameters

value	The value of the element

Definition at line 186 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}
```

The initial position of the element.

Definition at line 32 of file element_gui.hpp.

6.15.4.2 side

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

The side length of the element.

Definition at line 27 of file element_gui.hpp.

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

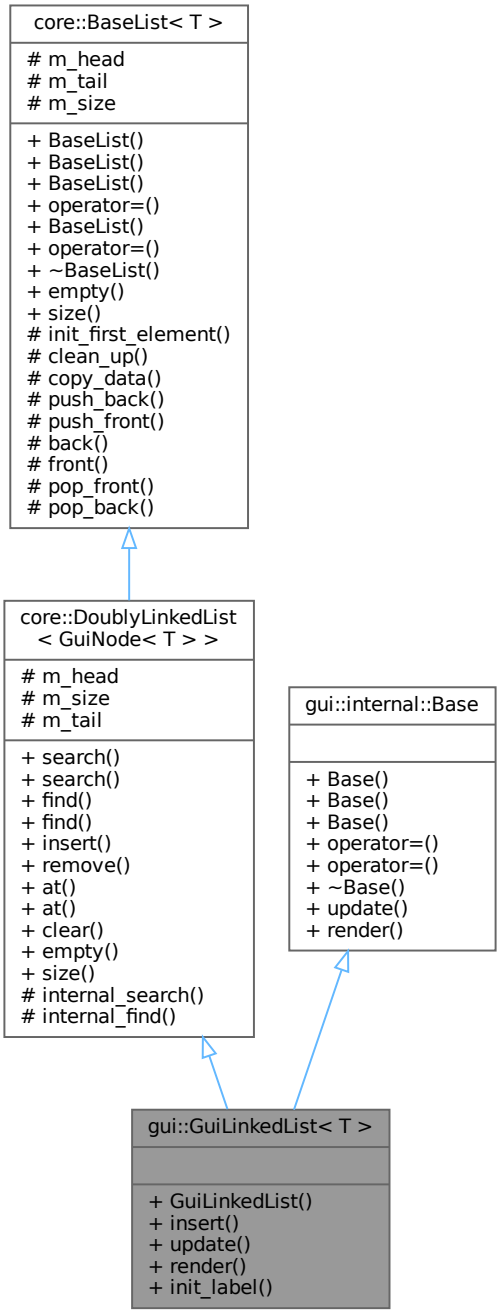
- src/gui/element_gui.hpp

6.16 gui::GuiLinkedList< T > Class Template Reference

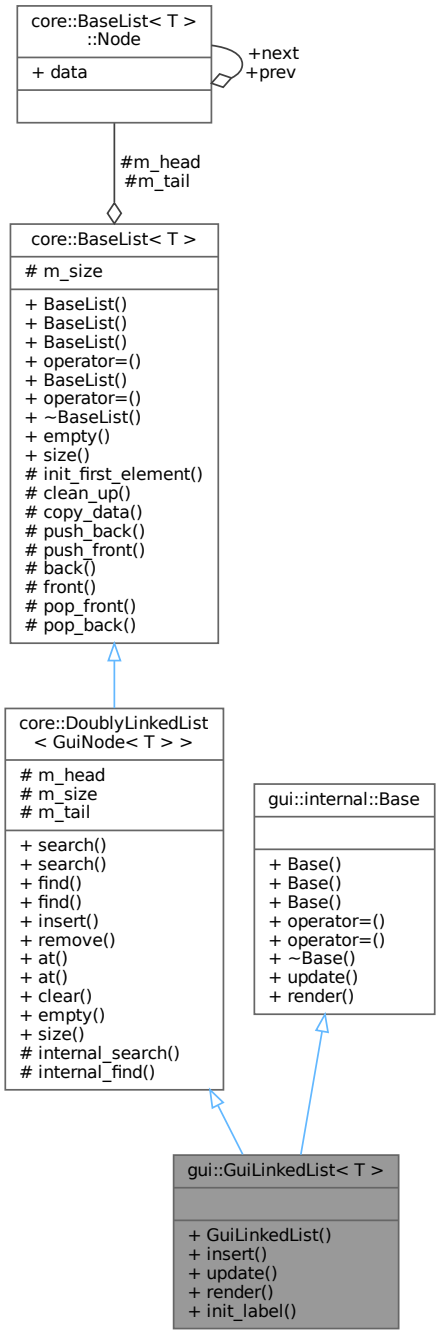
The GUI linked list container.

`#include <linked_list_gui.hpp>`

Inheritance diagram for gui::GuiLinkedList< T >:



Collaboration diagram for gui::GuiLinkedList< T >:



Public Member Functions

- GuiLinkedList (std::initializer_list< GuiNode< T > > init_list)

  Construct a new Gui Linked List object.
- void insert (std::size_t index, const T &elem)

  Inserts an element at the specified index.
- void update () override

Updates the GUI linked list.

- void render () override

  Renders the GUI linked list.

- void init_label ()

  Initializes the labels.

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

- Node_ptr search (const GuiNode< T > &elem)

  Searches for the element in the container.

- cNode_ptr search (const GuiNode< T > &elem) const

  Searches for the element in the container.

- Node_ptr find (std::size_t index)

  Finds the element at the specified index.

- cNode_ptr find (std::size_t index) const

  Finds the element at the specified index.

- Node_ptr insert (std::size_t index, const GuiNode< T > &elem)

  Inserts the element at the specified index.

- Node_ptr remove (std::size_t index)

  Removes the element at the specified index.

- GuiNode< T > & at (std::size_t index)

  Gets the element at the specified index.

- GuiNode< T > at (std::size_t index) const

  Gets the element at the specified index.

- void clear ()

  Clears the container.

- bool empty () const

  Check whether the container is empty.

- std::size_t size () const

  Returns the size of the container.

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

- BaseList ()=default

  Default constructor.

- BaseList (std::initializer_list< T > init_list)

  Constructs the container with the contents of the initializer list.

- BaseList (const BaseList &rhs)

  Copy constructor.

- BaseList & operator= (const BaseList &rhs)

  Copy assignment operator.

- BaseList (BaseList &&rhs) noexcept

  Move constructor.

- BaseList & operator= (BaseList &&rhs) noexcept

  Move assignment operator.

- ~BaseList ()

  Destructor.

- bool empty () const

  Check whether the container is empty.

- std::size_t size () const

  Returns the size of the container.

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

- Base ()=default

  Constructs a Base object.
- Base (const Base &)=default

  Copy constructor.
- Base (Base &&)=default

  Move constructor.
- Base & operator= (const Base &)=default

  Copy assignment operator.
- Base & operator= (Base &&)=default

  Move assignment operator.
- virtual ~Base ()=default

  Destructor.
- virtual void update ()=0

  Updates the GUI.
- virtual void render ()=0

  Renders the GUI.

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)

  Internal method to search for the element in the container.
- Node_ptr internal_find (std::size_t index)

  Internal method to find the element at the specified index.

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

- void init_first_element (const T &elem)

  Initializes the first element of the container.
- void clean_up ()

  Frees all elements in the container.
- void copy_data (const BaseList &rhs)

  Copies data from another container.
- void push_back (const T &elem)

  Pushes the element to the back of the container.
- void push_front (const T &elem)

  Pushes the element to the front of the container.
- T & back () const

  Returns the reference to the element at the back of the container.
- T & front () const

  Returns the reference to the element at the front of the container.
- void pop_front ()

  Removes the element at the back of the container.
- void pop_back ()

  Removes the element at the front of the container.

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

- Node_ptr m_head

  The head of the list.
- std::size_t m_size

  The size of the list.
- Node_ptr m_tail

  The tail of the list.

Protected Attributes inherited from core::BaseList< T >

- Node_ptr m_head {nullptr}

  The head of the list.
- Node_ptr m_tail {nullptr}

  The tail of the list.
- std::size_t m_size {}

  The size of the list.

6.16.1 Detailed Description

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

The GUI linked list container.

Template Parameters

T	the type of the elements

Definition at line 23 of file linked_list_gui.hpp.

6.16.2 Constructor & Destructor Documentation

6.16.2.1 GuiLinkedList()

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

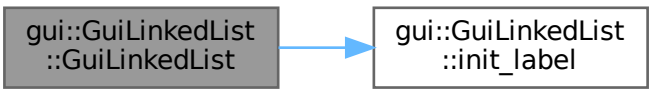
Construct a new Gui Linked List object.

Parameters

init_list	The initializer list

Definition at line 97 of file linked_list_gui.hpp.

Here is the call graph for this function:



6.16.3 Member Function Documentation

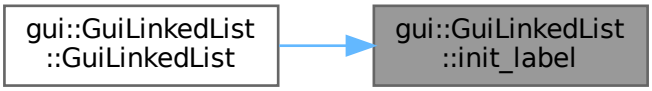
6.16.3.1 init_label()

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

Initializes the labels.

Definition at line 82 of file linked_list_gui.hpp.

Here is the caller graph for this function:



6.16.3.2 insert()

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

Inserts an element at the specified index.

Parameters

index	The index
elem	The element

Definition at line 103 of file linked_list_gui.hpp.

6.16.3.3 render()

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

Renders the GUI linked list.

Implements gui::internal::Base.

Definition at line 129 of file linked_list_gui.hpp.

6.16.3.4 update()

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

Updates the GUI linked list.

Implements gui::internal::Base.

Definition at line 142 of file linked_list_gui.hpp.

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

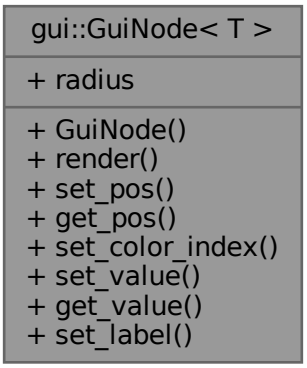
- src/gui/linked_list_gui.hpp

6.17 gui::GuiNode< T > Class Template Reference

The GUI node (used in linked lists)

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

Collaboration diagram for gui::GuiNode< T >:



Public Member Functions

- GuiNode (const T &value)

  Construct a new GuiNode object with the specified value.
- void render ()

  Renders the node.
- void set_pos (Vector2 pos)

  Sets the position of the node.
- Vector2 get_pos () const

Gets the position of the node.

- void set_color_index (int color_index)

  Sets the color index of the node.

- void set_value (const T &value)

  Sets the value of the node.

- T & get_value ()

  Returns the reference to the value of the node.

- void set_label (const char *label)

  Sets the label of the node.

Static Public Attributes

- static constexpr int radius = 20

  The radius of the node.

6.17.1 Detailed Description

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

The GUI node (used in linked lists)

Template Parameters

T	the type of the node

Definition at line 21 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]
```

Construct a new GuiNode object with the specified value.

Definition at line 106 of file node_gui.hpp.

6.17.3 Member Function Documentation

6.17.3.1 get_pos()

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

Gets the position of the node.

Returns

The position of the node

Definition at line 159 of file node_gui.hpp.

6.17.3.2 get_value()

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

Returns the reference to the value of the node.

Returns

T& The reference to the value of the node

Definition at line 149 of file node_gui.hpp.

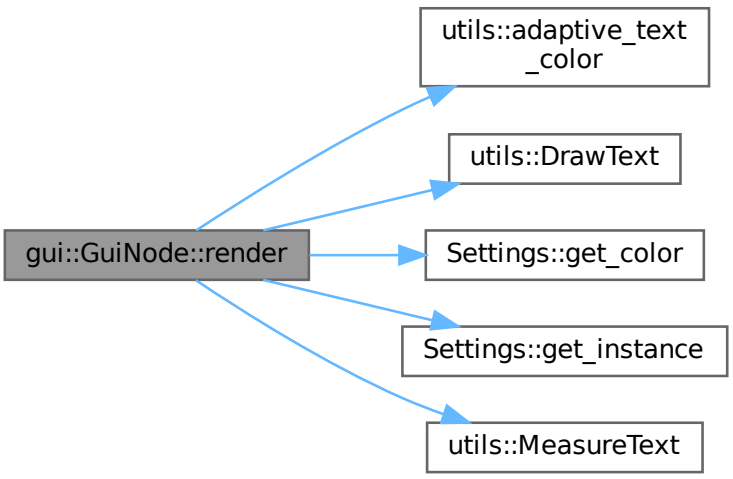
6.17.3.3 render()

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

Renders the node.

Definition at line 109 of file node_gui.hpp.

Here is the call graph for this function:



6.17.3.4 set_color_index()

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

Sets the color index of the node.

Parameters

color_index	The color index of the node in the settings

Definition at line 139 of file node_gui.hpp.

6.17.3.5 set_label()

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

Sets the label of the node.

Parameters

label	The label of the node

Definition at line 164 of file node_gui.hpp.

6.17.3.6 set_pos()

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

Sets the position of the node.

Parameters

pos	The position of the node

Definition at line 154 of file node_gui.hpp.

6.17.3.7 set_value()

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

Sets the value of the node.

Parameters

value	The value of the node

Definition at line 144 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]
```

The radius of the node.

Definition at line 26 of file node_gui.hpp.

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

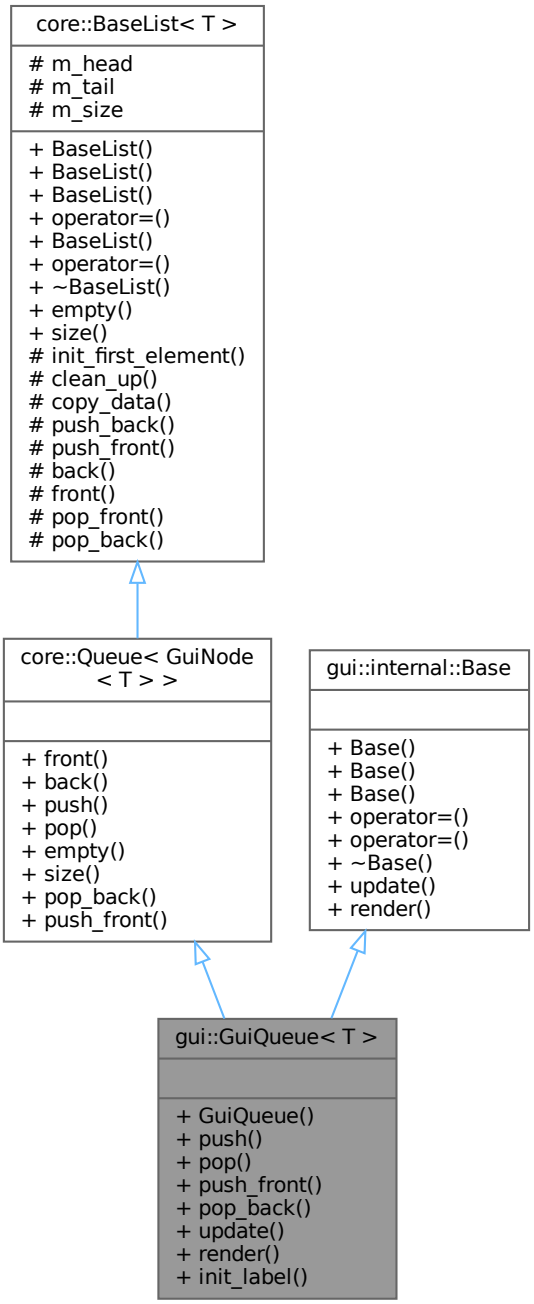
- src/gui/node_gui.hpp

6.18 gui::GuiQueue< T > Class Template Reference

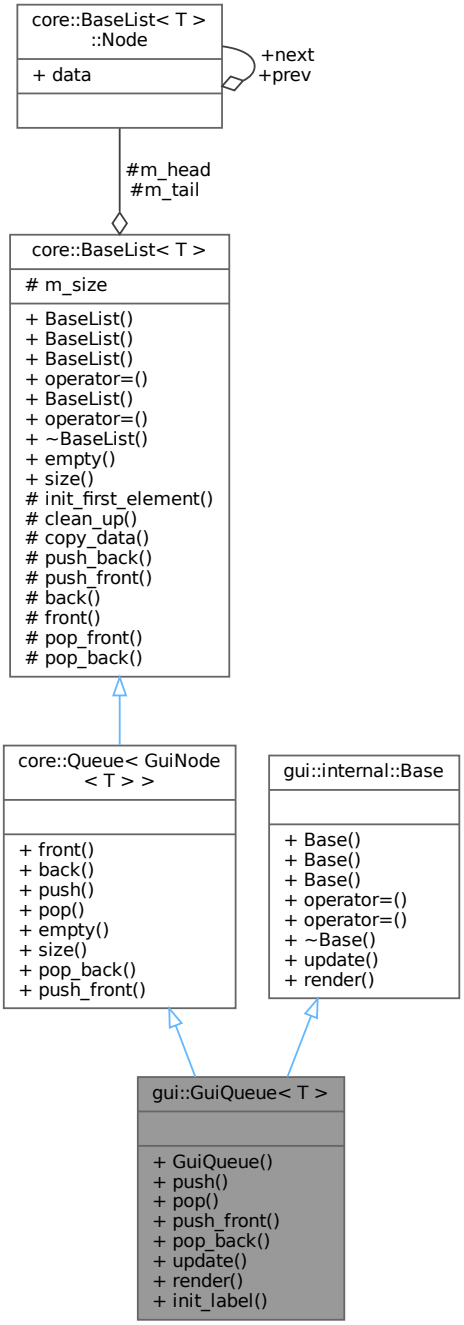
The GUI queue container.

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

Inheritance diagram for gui::GuiQueue< T >:



Collaboration diagram for gui::GuiQueue< T >:



Public Member Functions

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

  Construct a new Gui Queue object.
- void push (const T &elem)

  Pushes an element to the back of the queue.
- void pop ()

Pops an element from the front of the queue.

- void push_front (const T &elem)

  Pushes an element to the front of the queue.
- void pop_back ()

  Pops an element from the back of the queue.
- void update () override

  Updates the GUI queue.
- void render () override

  Renders the GUI queue.
- void init_label ()

  Initializes the labels.

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

- GuiNode< T > & front () const

  Returns the reference to the front element of the queue.
- GuiNode< T > & back () const

  Returns the reference to the back element of the queue.
- void push (const GuiNode< T > &elem)

  Inserts the element at the back of the queue.
- void pop ()

  Removes the front element of the queue.
- bool empty () const

  Check whether the container is empty.
- std::size_t size () const

  Returns the size of the container.
- void pop_back ()
- void push_front (const GuiNode< T > &elem)

  Pushes the element to the front of the container.

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

- BaseList ()=default

  Default constructor.
- BaseList (std::initializer_list< T > init_list)

  Constructs the container with the contents of the initializer list.
- BaseList (const BaseList &rhs)

  Copy constructor.
- BaseList & operator= (const BaseList &rhs)

  Copy assignment operator.
- BaseList (BaseList &&rhs) noexcept

  Move constructor.
- BaseList & operator= (BaseList &&rhs) noexcept

  Move assignment operator.
- ~BaseList ()

  Destructor.
- bool empty () const

  Check whether the container is empty.
- std::size_t size () const

  Returns the size of the container.

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

- Base ()=default

  Constructs a Base object.
- Base (const Base &)=default

  Copy constructor.
- Base (Base &&)=default

  Move constructor.
- Base & operator= (const Base &)=default

  Copy assignment operator.
- Base & operator= (Base &&)=default

  Move assignment operator.
- virtual ~Base ()=default

  Destructor.
- virtual void update ()=0

  Updates the GUI.
- virtual void render ()=0

  Renders the GUI.

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)

  Initializes the first element of the container.
- void clean_up ()

  Frees all elements in the container.
- void copy_data (const BaseList &rhs)

  Copies data from another container.
- void push_back (const T &elem)

  Pushes the element to the back of the container.
- void push_front (const T &elem)

  Pushes the element to the front of the container.
- T & back () const

  Returns the reference to the element at the back of the container.
- T & front () const

  Returns the reference to the element at the front of the container.
- void pop_front ()

  Removes the element at the back of the container.
- void pop_back ()

  Removes the element at the front of the container.

Protected Attributes inherited from core::BaseList< T >

- Node_ptr m_head {nullptr}

  The head of the list.
- Node_ptr m_tail {nullptr}

  The tail of the list.
- std::size_t m_size {}

  The size of the list.

6.18.1 Detailed Description

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

The GUI queue container.

Template Parameters

T	the type of the elements

Definition at line 22 of file queue_gui.hpp.

6.18.2 Constructor & Destructor Documentation

6.18.2.1 GuiQueue()

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

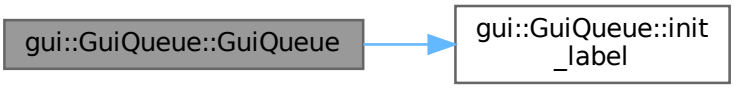
Construct a new Gui Queue object.

Parameters

init_list	The initializer list

Definition at line 111 of file queue_gui.hpp.

Here is the call graph for this function:



6.18.3 Member Function Documentation

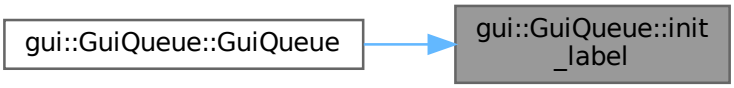
6.18.3.1 init_label()

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

Initializes the labels.

Definition at line 96 of file queue_gui.hpp.

Here is the caller graph for this function:



6.18.3.2 pop()

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

Pops an element from the front of the queue.

Definition at line 122 of file queue_gui.hpp.

6.18.3.3 pop_back()

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

Pops an element from the back of the queue.

Definition at line 132 of file queue_gui.hpp.

6.18.3.4 push()

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

Pushes an element to the back of the queue.

Parameters

elem	The element

Definition at line 117 of file queue_gui.hpp.

6.18.3.5 push_front()

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

Pushes an element to the front of the queue.

Parameters

elem	The element

Note

This is for demonstration purposes only

Definition at line 127 of file queue_gui.hpp.

6.18.3.6 render()

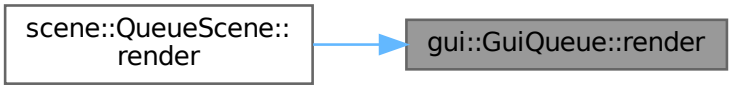
```
template<typename T >
void gui::GuiQueue< T >::render  [override], [virtual]
```

Renders the GUI queue.

Implements gui::internal::Base.

Definition at line 158 of file queue_gui.hpp.

Here is the caller graph for this function:



6.18.3.7 update()

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

Updates the GUI queue.

Implements gui::internal::Base.

Definition at line 171 of file queue_gui.hpp.

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

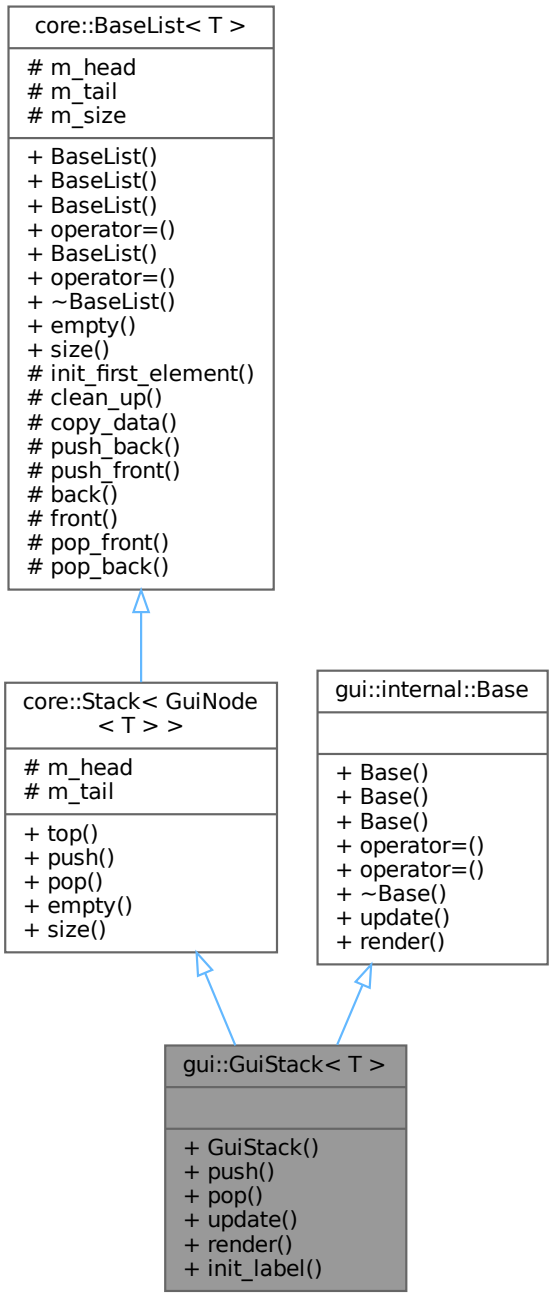
- src/gui/queue_gui.hpp

6.19 gui::GuiStack< T > Class Template Reference

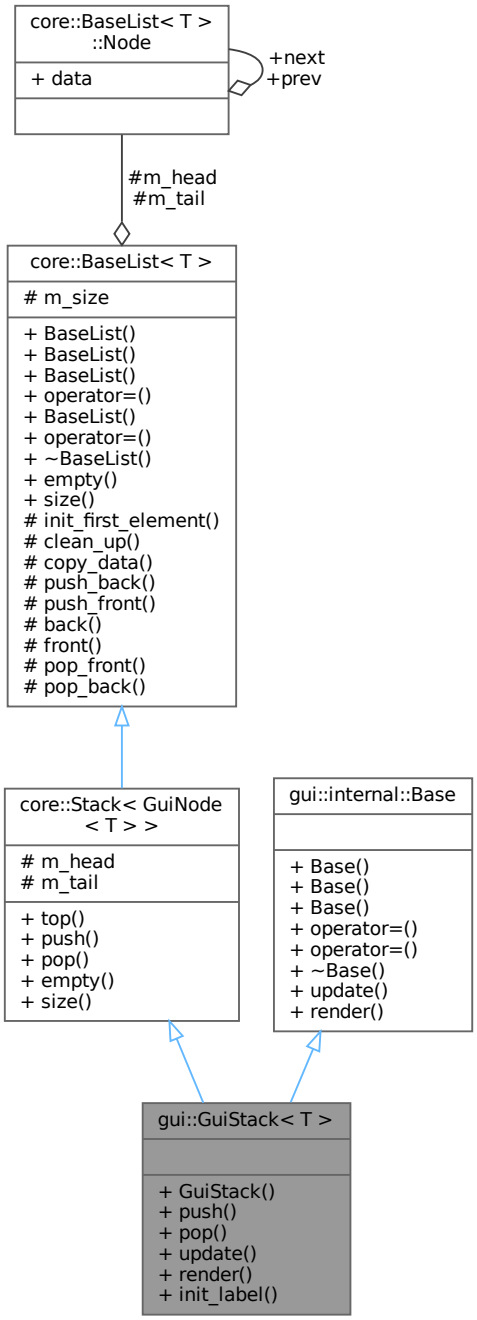
The GUI stack container.

`#include <stack_gui.hpp>`

Inheritance diagram for gui::GuiStack< T >:



Collaboration diagram for gui::GuiStack< T >:



Public Member Functions

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

  Construct a new Gui Stack object.
- void push (const T &elem)

  Pushes an element to the top of the stack.
- void pop ()

Pops an element from the top of the stack.

- void update () override

  Updates the GUI stack.
- void render () override

  Renders the GUI stack.
- void init_label ()

  Initializes the labels.

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

- GuiNode< T > & top () const

  Returns the reference to the top element of the stack.
- void push (const GuiNode< T > &elem)

  Inserts the element at the top of the stack.
- void pop ()

  Removes the top element of the stack.
- bool empty () const

  Check whether the container is empty.
- std::size_t size () const

  Returns the size of the container.

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

- BaseList ()=default

  Default constructor.
- BaseList (std::initializer_list< T > init_list)

  Constructs the container with the contents of the initializer list.
- BaseList (const BaseList &rhs)

  Copy constructor.
- BaseList & operator= (const BaseList &rhs)

  Copy assignment operator.
- BaseList (BaseList &&rhs) noexcept

  Move constructor.
- BaseList & operator= (BaseList &&rhs) noexcept

  Move assignment operator.
- ~BaseList ()

  Destructor.
- bool empty () const

  Check whether the container is empty.
- std::size_t size () const

  Returns the size of the container.

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

- Base ()=default

  Constructs a Base object.
- Base (const Base &)=default

  Copy constructor.
- Base (Base &&)=default

  Move constructor.
- Base & operator= (const Base &)=default

  Copy assignment operator.
- Base & operator= (Base &&)=default

  Move assignment operator.
- virtual ~Base ()=default

  Destructor.
- virtual void update ()=0

  Updates the GUI.
- virtual void render ()=0

  Renders the GUI.

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)

  Initializes the first element of the container.
- void clean_up ()

  Frees all elements in the container.
- void copy_data (const BaseList &rhs)

  Copies data from another container.
- void push_back (const T &elem)

  Pushes the element to the back of the container.
- void push_front (const T &elem)

  Pushes the element to the front of the container.
- T & back () const

  Returns the reference to the element at the back of the container.
- T & front () const

  Returns the reference to the element at the front of the container.
- void pop_front ()

  Removes the element at the back of the container.
- void pop_back ()

  Removes the element at the front of the container.

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

- Node_ptr m_head

  The head of the list.
- Node_ptr m_tail

  The tail of the list.

Protected Attributes inherited from core::BaseList< T >

- Node_ptr m_head {nullptr}

  The head of the list.
- Node_ptr m_tail {nullptr}

  The tail of the list.
- std::size_t m_size {}

  The size of the list.

6.19.1 Detailed Description

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

The GUI stack container.

Template Parameters

T	the type of the elements

Definition at line 22 of file stack_gui.hpp.

6.19.2 Constructor & Destructor Documentation

6.19.2.1 GuiStack()

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

Construct a new Gui Stack object.

Parameters

init_list	The initializer list

Definition at line 91 of file stack_gui.hpp.

Here is the call graph for this function:

6.19.3 Member Function Documentation

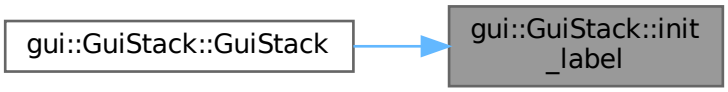
6.19.3.1 init_label()

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

Initializes the labels.

Definition at line 84 of file stack_gui.hpp.

Here is the caller graph for this function:



6.19.3.2 pop()

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

Pops an element from the top of the stack.

Definition at line 102 of file stack_gui.hpp.

6.19.3.3 push()

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

Pushes an element to the top of the stack.

Parameters

elem	The element

Definition at line 97 of file stack_gui.hpp.

6.19.3.4 render()

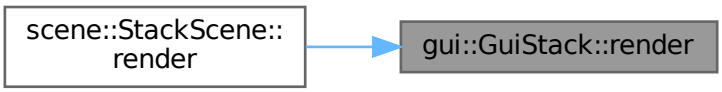
```
template<typename T >
void gui::GuiStack< T >::render  [override], [virtual]
```

Renders the GUI stack.

Implements gui::internal::Base.

Definition at line 128 of file stack_gui.hpp.

Here is the caller graph for this function:



6.19.3.5 update()

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

Updates the GUI stack.

Implements gui::internal::Base.

Definition at line 141 of file stack_gui.hpp.

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

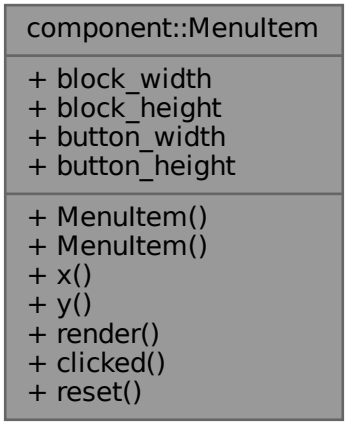
- src/gui/stack_gui.hpp

6.20 component::MenuItem Class Reference

Items in the menu screen to navigate to other screens.

`#include <menu_item.hpp>`

Collaboration diagram for component::MenuItem:



Public Member Functions

- MenuItem ()=default

  Constructs a MenuItem object.
- MenuItem (const char *text, int x, int y, const char *img_path)

  Constructs a MenuItem object.
- int x () const

  Returns the x position of the menu item.
- int y () const

  Returns the y position of the menu item.
- void render ()

  Renders the menu item.
- bool clicked () const

  Checks if the menu item was clicked.
- void reset ()

  Resets the menu item.

Static Public Attributes

- static constexpr int block_width = 300

  The width of the menu item frame.
- static constexpr int block_height = 200

  The height of the menu item frame.
- static constexpr int button_width = block_width

  The width of the button on the menu item.
- static constexpr int button_height = 50

  The height of the button on the menu item.

6.20.1 Detailed Description

Items in the menu screen to navigate to other screens.

Definition at line 12 of file menu_item.hpp.

6.20.2 Constructor & Destructor Documentation

6.20.2.1 MenuItem() [1/2]

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

Constructs a MenuItem object.

6.20.2.2 MenuItem() [2/2]

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

Constructs a MenuItem object.

Parameters

text	the text of the menu item
x	the x position of the menu item
y	the y position of the menu item
img_path	the path to the image of the menu item

Definition at line 8 of file menu_item.cpp.

6.20.3 Member Function Documentation

6.20.3.1 clicked()

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

Checks if the menu item was clicked.

Return values

true	The menu item was clicked
false	The menu item was not clicked

Definition at line 36 of file menu_item.cpp.

6.20.3.2 render()

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

Renders the menu item.

Definition at line 17 of file menu_item.cpp.

6.20.3.3 reset()

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

Resets the menu item.

Definition at line 38 of file menu_item.cpp.

6.20.3.4 x()

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

Returns the x position of the menu item.

Returns

the x position of the menu item

Definition at line 14 of file menu_item.cpp.

6.20.3.5 y()

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

Returns the y position of the menu item.

Returns

the y position of the menu item

Definition at line 15 of file menu_item.cpp.

6.20.4 Member Data Documentation

6.20.4.1 block_height

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

The height of the menu item frame.

Definition at line 22 of file menu_item.hpp.

6.20.4.2 block_width

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

The width of the menu item frame.

Definition at line 17 of file menu_item.hpp.

6.20.4.3 button_height

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

The height of the button on the menu item.

Definition at line 32 of file menu_item.hpp.

6.20.4.4 button_width

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

The width of the button on the menu item.

Definition at line 27 of file menu_item.hpp.

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

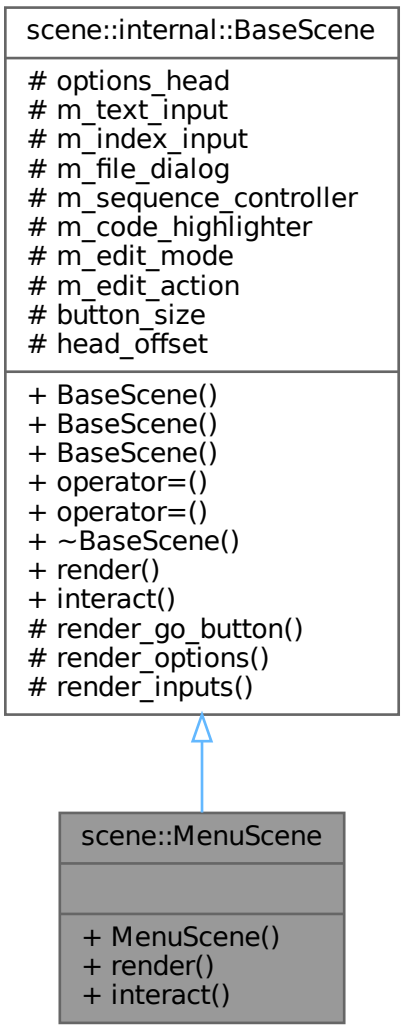
- src/component/menu_item.hpp
- src/component/menu_item.cpp

6.21 scene::MenuScene Class Reference

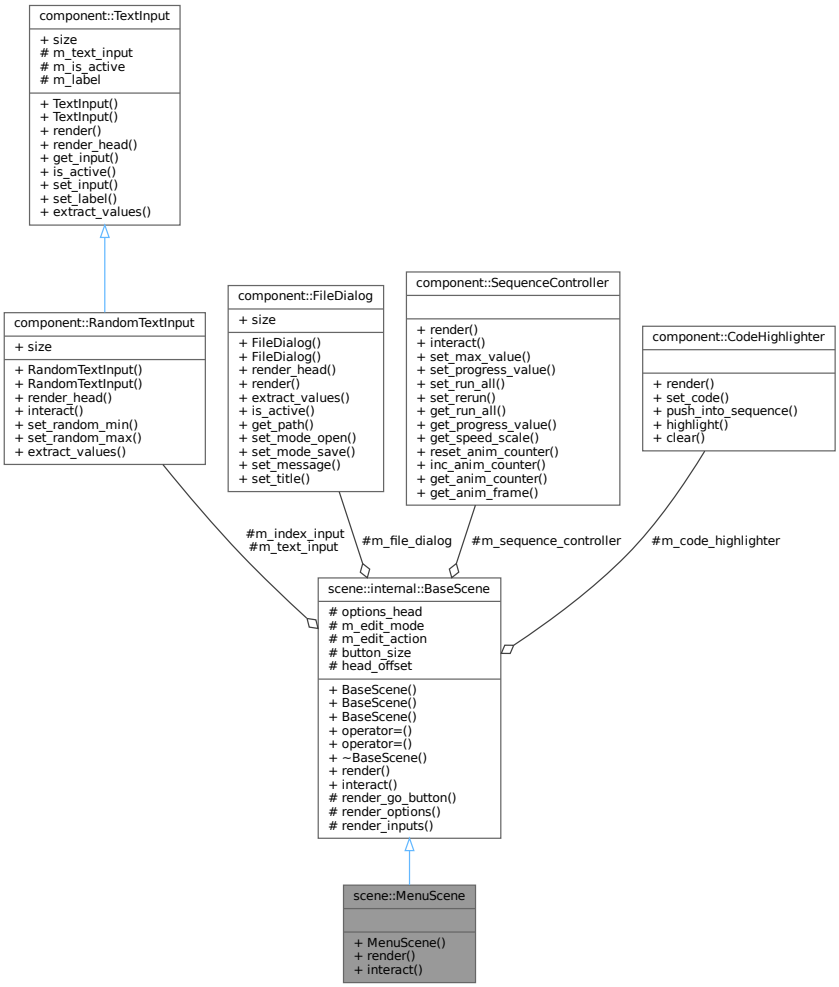
The menu scene.

`#include <menu_scene.hpp>`

Inheritance diagram for scene::MenuScene:



Collaboration diagram for scene::MenuScene:



Public Member Functions

- MenuScene ()

  Construct a new MenuScene object.
- void render () override

  Renders the scene.
- void interact () override

  Interacts with the scene.

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

- BaseScene ()=default

  Construct a new BaseScene object.
- BaseScene (const BaseScene &)=delete

  Copy constructor (deleted)
- BaseScene (BaseScene &&)=delete

Move constructor (deleted)

- BaseScene & operator= (const BaseScene &)=delete

  Copy assignment (deleted)
- BaseScene & operator= (BaseScene &&)=delete

  Move assignment (deleted)
- virtual ~BaseScene ()=default

  Destroy the BaseScene object.
- virtual void render ()

  Renders the scene.
- virtual void interact ()

  Interacts with the scene.

Additional Inherited Members

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

- virtual bool render_go_button () const

  Renders the go button.
- virtual void render_options (SceneOptions &scene_config)

  Renders the options.
- virtual void render_inputs ()

  Renders the inputs.

Protected Attributes inherited from scene::internal::BaseScene

- float options_head {}

  The head of the options.
- component::RandomTextInput m_text_input {"value"}

  The text input for the value.
- component::RandomTextInput m_index_input {"index"}

  The text input for the index.
- component::FileDialog m_file_dialog

  The file dialog.
- component::SequenceController m_sequence_controller

  The sequence controller.
- component::CodeHighlighter m_code_highlighter

  The code highlighter.
- bool m_edit_mode {}

  Whether the edit mode is enabled.
- bool m_edit_action {}

  Whether the edit action is enabled.

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

- static constexpr Vector2 button_size {200, 50}

  The size of the buttons.
- static constexpr int head_offset = 20

  The offset of the widgets.

6.21.1 Detailed Description

The menu scene.

Definition at line 16 of file menu_scene.hpp.

6.21.2 Constructor & Destructor Documentation

6.21.2.1 MenuScene()

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

Construct a new MenuScene object.

Definition at line 14 of file menu_scene.cpp.

6.21.3 Member Function Documentation

6.21.3.1 interact()

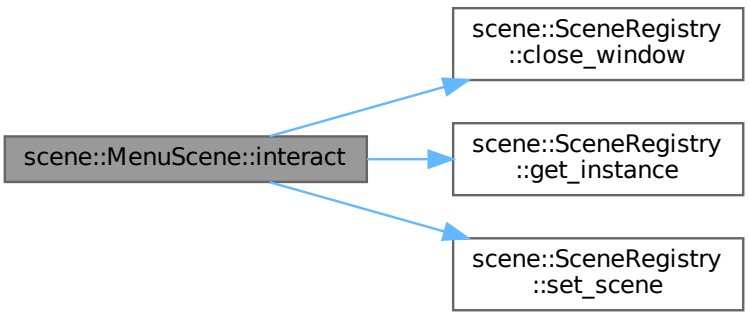
```
void scene::MenuScene::interact ( )  [override], [virtual]
```

Interacts with the scene.

Reimplemented from scene::internal::BaseScene.

Definition at line 125 of file menu_scene.cpp.

Here is the call graph for this function:



6.21.3.2 render()

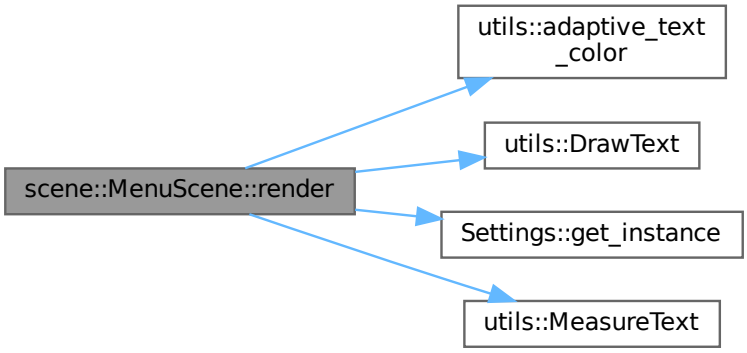
```
void scene::MenuScene::render ( )  [override], [virtual]
```

Renders the scene.

Reimplemented from scene::internal::BaseScene.

Definition at line 52 of file menu_scene.cpp.

Here is the call graph for this function:



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

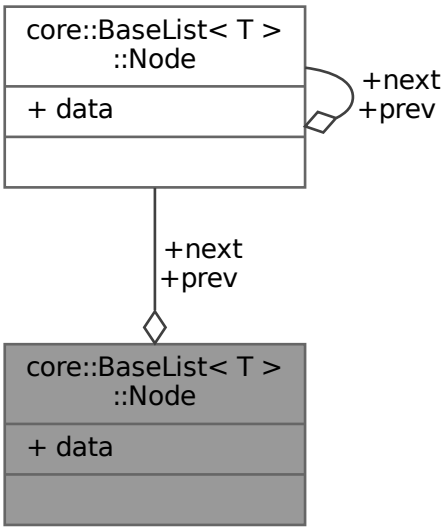
- src/scene/menu_scene.hpp
- src/scene/menu_scene.cpp

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

The node of the list.

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

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



Public Attributes

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

6.22.1 Detailed Description

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

The node of the list.

Definition at line 78 of file base_list.hpp.

6.22.2 Member Data Documentation

6.22.2.1 data

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

Definition at line 79 of file base_list.hpp.

6.22.2.2 next

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

Definition at line 81 of file base_list.hpp.

6.22.2.3 prev

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

Definition at line 80 of file base_list.hpp.

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

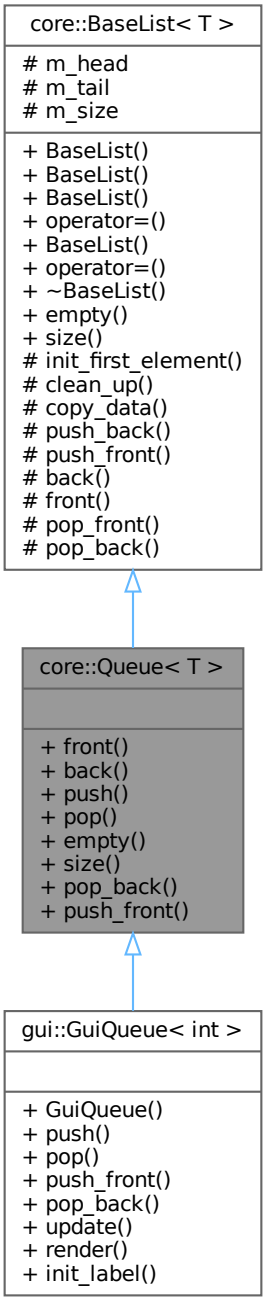
- src/core/base_list.hpp

6.23 core::Queue< T > Class Template Reference

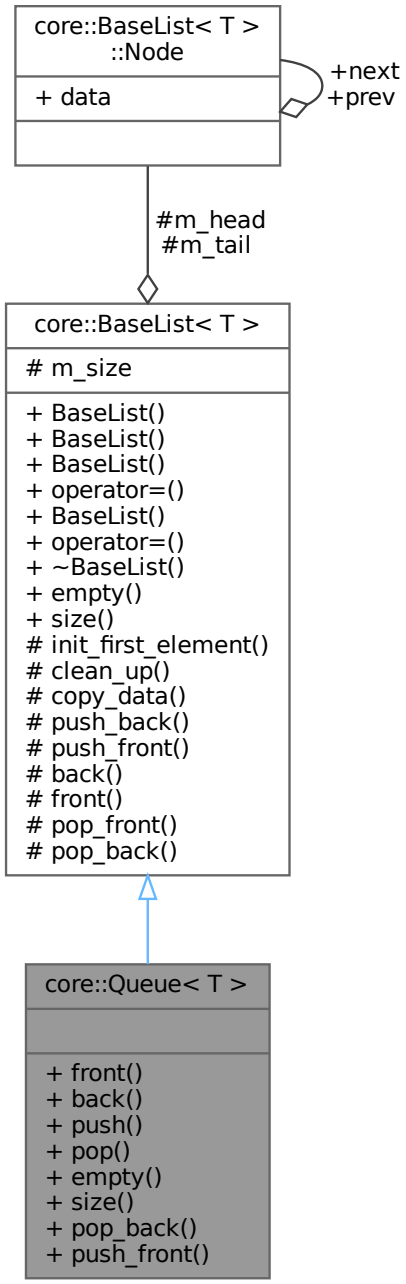
The queue container.

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

Inheritance diagram for core::Queue< T >:



Collaboration diagram for core::Queue< T >:



Public Member Functions

- T & front () const

  Returns the reference to the front element of the queue.
- T & back () const

  Returns the reference to the back element of the queue.
- void push (const T &elem)

Inserts the element at the back of the queue.

- void pop ()

  Removes the front element of the queue.

- bool empty () const

  Check whether the container is empty.

- std::size_t size () const

  Returns the size of the container.

- void pop_back ()
- void push_front (const T &elem)

  Pushes the element to the front of the container.

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

- BaseList ()=default

  Default constructor.

- BaseList (std::initializer_list< T > init_list)

  Constructs the container with the contents of the initializer list.

- BaseList (const BaseList &rhs)

  Copy constructor.

- BaseList & operator= (const BaseList &rhs)

  Copy assignment operator.

- BaseList (BaseList &&rhs) noexcept

  Move constructor.

- BaseList & operator= (BaseList &&rhs) noexcept

  Move assignment operator.

- ~BaseList ()

  Destructor.

- bool empty () const

  Check whether the container is empty.

- std::size_t size () const

  Returns the size of the container.

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)

  Initializes the first element of the container.

- void clean_up ()

  Frees all elements in the container.

- void copy_data (const BaseList &rhs)

  Copies data from another container.

- void push_back (const T &elem)

  Pushes the element to the back of the container.

- void push_front (const T &elem)

Pushes the element to the front of the container.

- T & back () const

  Returns the reference to the element at the back of the container.
- T & front () const

  Returns the reference to the element at the front of the container.
- void pop_front ()

  Removes the element at the back of the container.
- void pop_back ()

  Removes the element at the front of the container.

Protected Attributes inherited from core::BaseList< T >

- Node_ptr m_head {nullptr}

  The head of the list.
- Node_ptr m_tail {nullptr}

  The tail of the list.
- std::size_t m_size {}

  The size of the list.

6.23.1 Detailed Description

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

The queue container.

Template Parameters

T	the type of the elements

Definition at line 14 of file queue.hpp.

6.23.2 Member Function Documentation

6.23.2.1 back()

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

Returns the reference to the back element of the queue.

Returns

T& the reference to the back element of the queue

Definition at line 60 of file queue.hpp.

6.23.2.2 empty()

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

Check whether the container is empty.

Return values

true	The container is empty
false	The container is not empty

Definition at line 63 of file base_list.hpp.

6.23.2.3 front()

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

Returns the reference to the front element of the queue.

Returns

T& the reference to the front element of the queue

Definition at line 55 of file queue.hpp.

6.23.2.4 pop()

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

Removes the front element of the queue.

Definition at line 70 of file queue.hpp.

6.23.2.5 pop_back()

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

Note

For animation purpose only, not for real use

Definition at line 150 of file base_list.hpp.

6.23.2.6 push()

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

Inserts the element at the back of the queue.

Parameters

elem	The element to insert

Definition at line 65 of file queue.hpp.

6.23.2.7 push_front()

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

Pushes the element to the front of the container.

Parameters

elem	The element to be pushed into the front

Definition at line 128 of file base_list.hpp.

6.23.2.8 size()

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

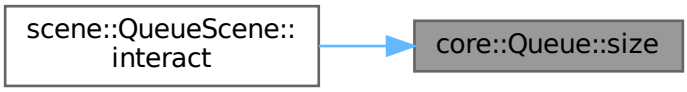
Returns the size of the container.

Returns

The size of the container

Definition at line 69 of file base_list.hpp.

Here is the caller graph for this function:



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

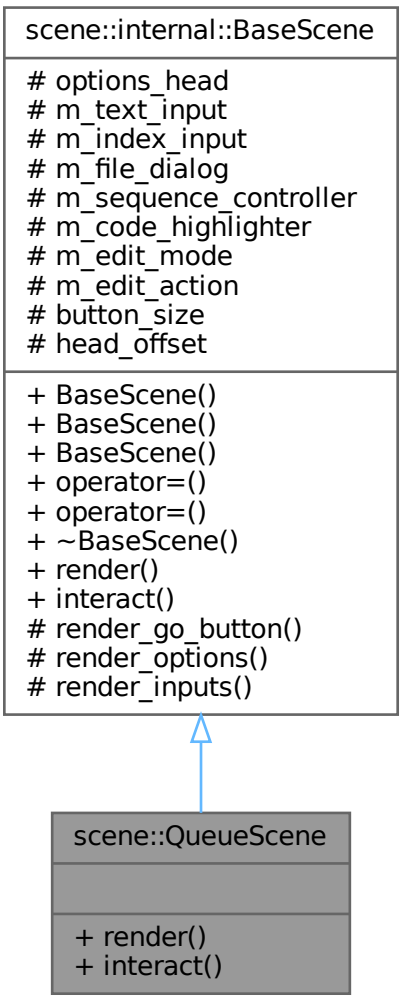
- src/core/queue.hpp

6.24 scene::QueueScene Class Reference

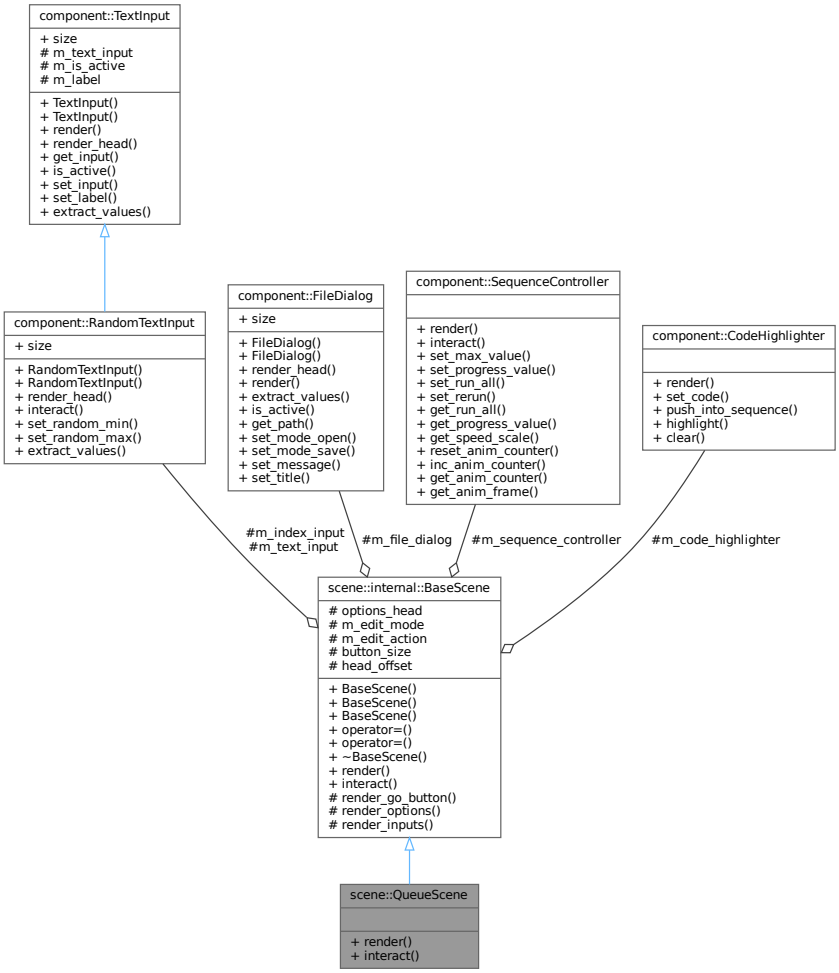
The queue scene.

`#include <queue_scene.hpp>`

Inheritance diagram for scene::QueueScene:



Collaboration diagram for scene::QueueScene:



Public Member Functions

- void render () override

  Renders the scene.
- void interact () override

  Interacts with the scene.

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

- BaseScene ()=default

  Construct a new BaseScene object.
- BaseScene (const BaseScene &)=delete

  Copy constructor (deleted)
- BaseScene (BaseScene &&)=delete

  Move constructor (deleted)
- BaseScene & operator= (const BaseScene &)=delete

  Copy assignment (deleted)

- BaseScene & operator= (BaseScene &&)=delete

  Move assignment (deleted)
- virtual ~BaseScene ()=default

  Destroy the BaseScene object.
- virtual void render ()

  Renders the scene.
- virtual void interact ()

  Interacts with the scene.

Additional Inherited Members

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

- virtual bool render_go_button () const

  Renders the go button.
- virtual void render_options (SceneOptions &scene_config)

  Renders the options.
- virtual void render_inputs ()

  Renders the inputs.

Protected Attributes inherited from scene::internal::BaseScene

- float options_head {}

  The head of the options.
- component::RandomTextInput m_text_input {"value"}

  The text input for the value.
- component::RandomTextInput m_index_input {"index"}

  The text input for the index.
- component::FileDialog m_file_dialog

  The file dialog.
- component::SequenceController m_sequence_controller

  The sequence controller.
- component::CodeHighlighter m_code_highlighter

  The code highlighter.
- bool m_edit_mode {}

  Whether the edit mode is enabled.
- bool m_edit_action {}

  Whether the edit action is enabled.

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

- static constexpr Vector2 button_size {200, 50}

  The size of the buttons.
- static constexpr int head_offset = 20

  The offset of the widgets.

6.24.1 Detailed Description

The queue scene.

Definition at line 19 of file queue_scene.hpp.

6.24.2 Member Function Documentation

6.24.2.1 interact()

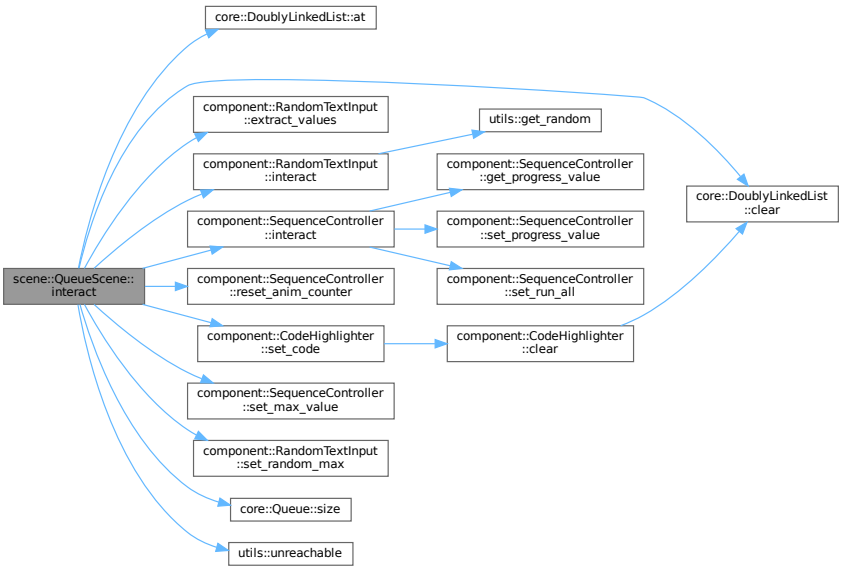
```
void scene::QueueScene::interact ( )  [override], [virtual]
```

Interacts with the scene.

Reimplemented from scene::internal::BaseScene.

Definition at line 71 of file queue_scene.cpp.

Here is the call graph for this function:



6.24.2.2 render()

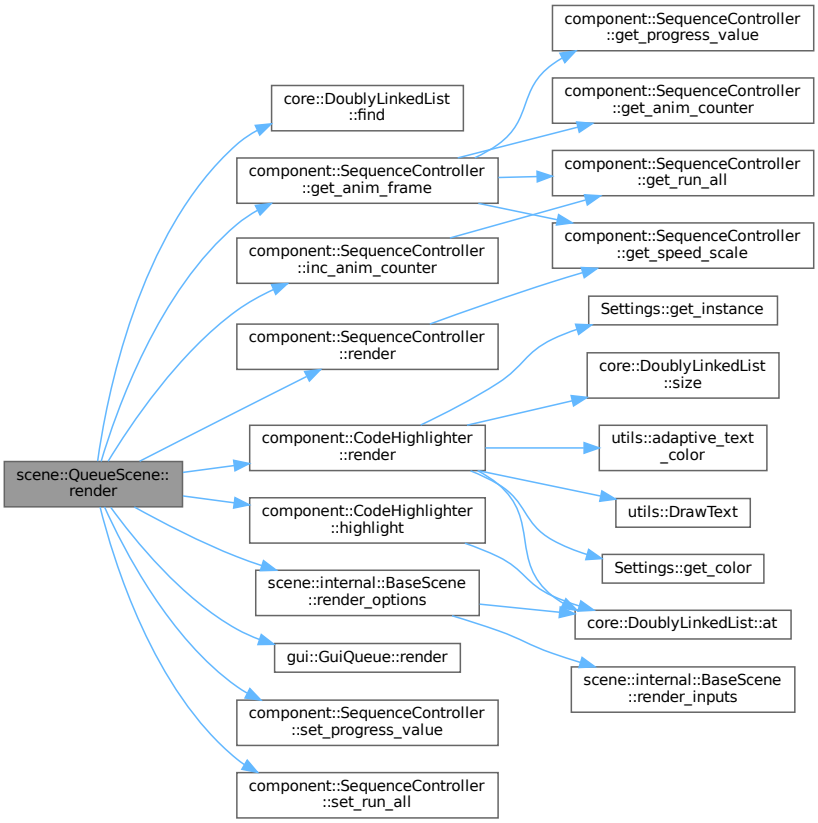
```
void scene::QueueScene::render ( )  [override], [virtual]
```

Renders the scene.

Reimplemented from scene::internal::BaseScene.

Definition at line 51 of file queue_scene.cpp.

Here is the call graph for this function:



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

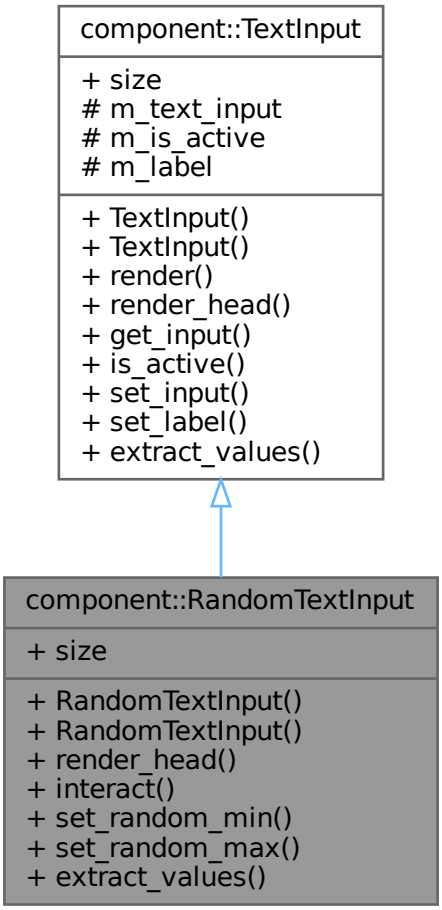
- src/scene/queue_scene.hpp
- src/scene/queue_scene.cpp

6.25 component::RandomTextInput Class Reference

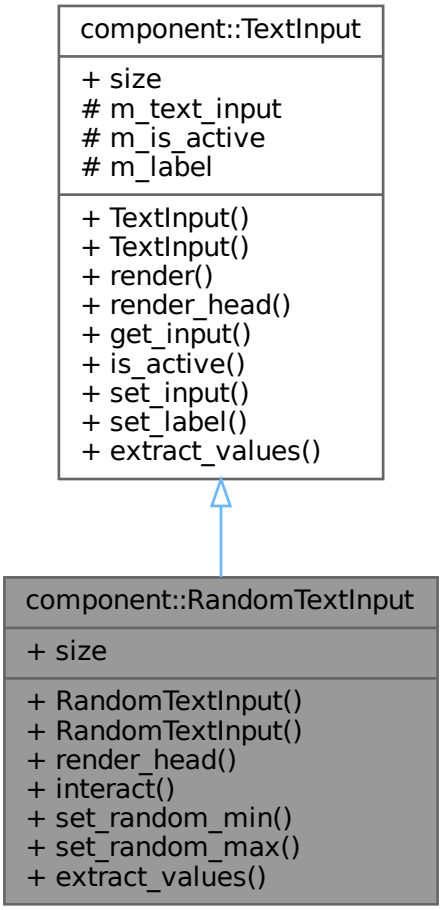
Text input that supports random values.

```
#include <random_text_input.hpp>
```

Inheritance diagram for component::RandomTextInput:



Collaboration diagram for component::RandomTextInput:



Public Member Functions

- RandomTextInput ()=default

  Constructs a RandomTextInput object.
- RandomTextInput (const char *label)

  Constructs a RandomTextInput object.
- void render_head (float &options_head, float head_offset)

  Renders the random text input, updates the head position with offset.
- bool interact ()

  Checks if the random text input is interacted with.
- void set_random_min (int value)

  Sets the min value of the random text input.
- void set_random_max (int value)

  Sets the max value of the random text input.
- core::Deque< int > extract_values ()

  Extracts the values from the text input.

Public Member Functions inherited from component::TextInput

- TextInput ()=default

  Constructs a TextInput object.
- TextInput (const char *label)

  Constructs a TextInput object.
- void render (float x, float y)

  Renders the text input.
- void render_head (float &options_head, float head_offset)

  Renders the text input, updates the head position with offset.
- std::string get_input () const

  Returns the input of the text input.
- bool is_active () const

  Checks if the text input is active.
- void set_input (const char *input, int len)

  Sets the input of the text input.
- void set_label (const char *const label)

  Sets the label of the text input.
- core::Deque< int > extract_values ()

  Extracts the values from the text input.

Static Public Attributes

- static constexpr Vector2 size

  The size of the text input.

Static Public Attributes inherited from component::TextInput

- static constexpr Vector2 size {200, 50}

  The size of the text input.

Additional Inherited Members

Protected Attributes inherited from component::TextInput

- char m_text_input [constants::text_buffer_size] = ""

  The text input.
- bool m_is_active {}

  Whether the text input is active.
- const char * m_label {}

  The label of the text input.

6.25.1 Detailed Description

Text input that supports random values.

Definition at line 17 of file random_text_input.hpp.

6.25.2 Constructor & Destructor Documentation

6.25.2.1 RandomTextInput() [1/2]

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

Constructs a RandomTextInput object.

6.25.2.2 RandomTextInput() [2/2]

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

Constructs a RandomTextInput object.

Parameters

label	the label of the random text input

Definition at line 14 of file random_text_input.cpp.

6.25.3 Member Function Documentation

6.25.3.1 extract_values()

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

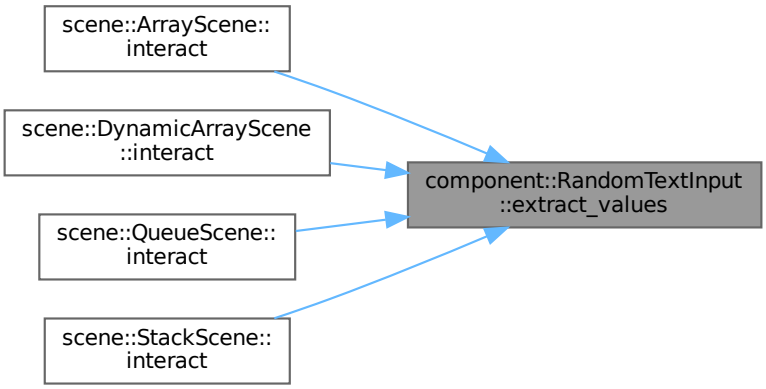
Extracts the values from the text input.

Returns

the values from the text input

Definition at line 78 of file text_input.cpp.

Here is the caller graph for this function:



6.25.3.2 interact()

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

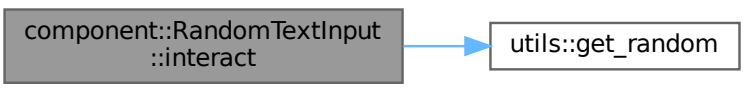
Checks if the random text input is interacted with.

Return values

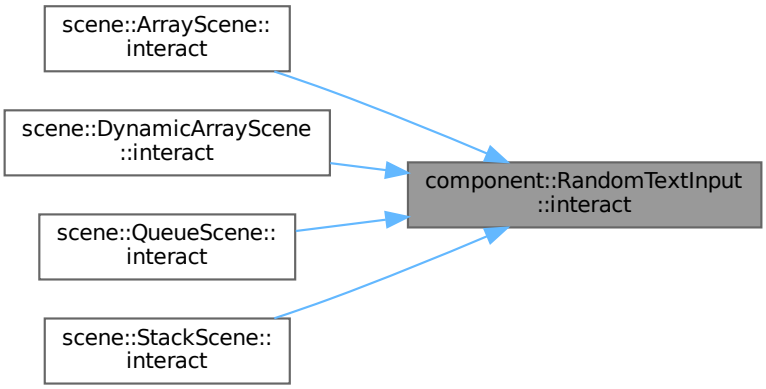
true	The random text input is interacted with
false	The random text input is not interacted with

Definition at line 30 of file random_text_input.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



6.25.3.3 render_head()

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

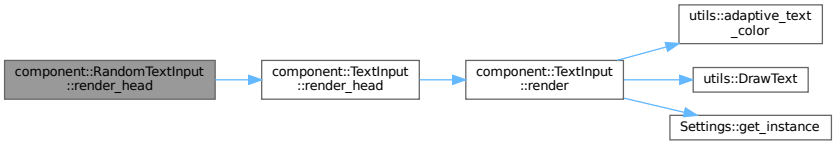
Renders the random text input, updates the head position with offset.

Parameters

options_head	the head position of the options
head_offset	the offset of the head position

Definition at line 20 of file random_text_input.cpp.

Here is the call graph for this function:



6.25.3.4 set_random_max()

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

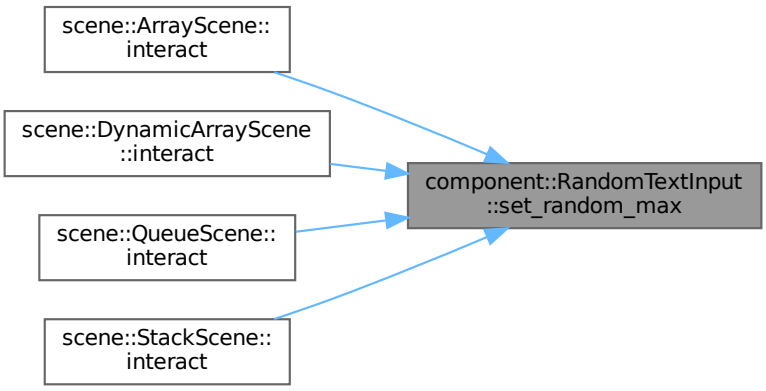
Sets the max value of the random text input.

Parameters

value	the max value of the random text input

Definition at line 18 of file random_text_input.cpp.

Here is the caller graph for this function:



6.25.3.5 set_random_min()

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

Sets the min value of the random text input.

Parameters

value	the min value of the random text input

Definition at line 16 of file random_text_input.cpp.

6.25.4 Member Data Documentation

6.25.4.1 size

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

The size of the text input.

Definition at line 21 of file text_input.hpp.

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

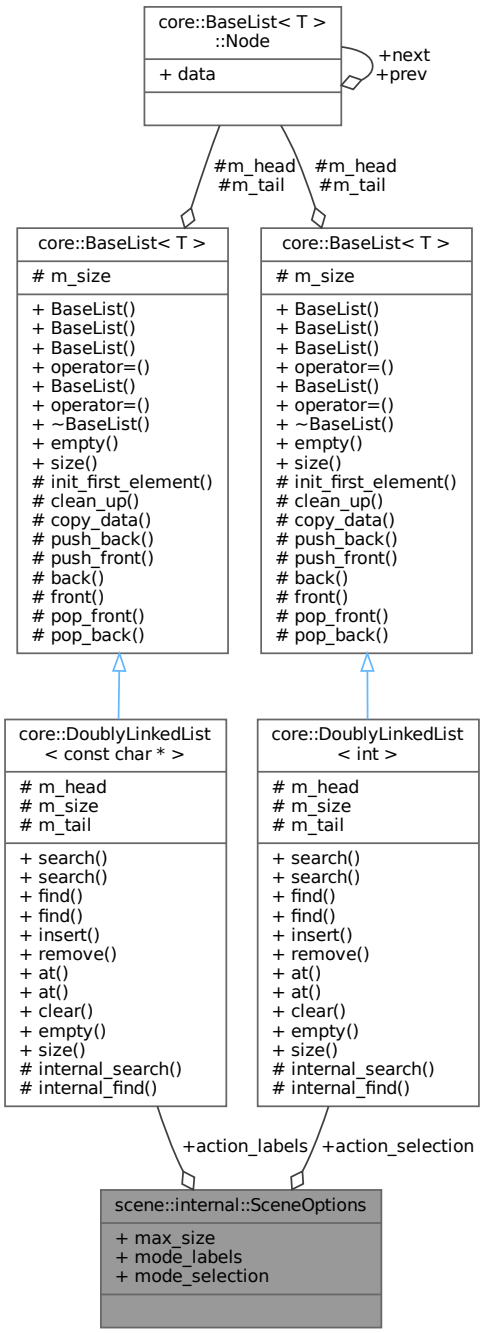
- src/component/random_text_input.hpp
- src/component/random_text_input.cpp

6.26 scene::internal::SceneOptions Struct Reference

The scene options.

```
#include <scene_options.hpp>
```

Collaboration diagram for scene::internal::SceneOptions:



Public Attributes

- const std::size_t max_size {}

  The maximum size of the container.
- const char ∗ mode_labels {}

  The mode labels.
- int mode_selection {}

The currently selected mode.

- core::DoublyLinkedList< const char * > action_labels

  The action labels.

- core::DoublyLinkedList< int > action_selection

  The currently selected actions for each mode.

6.26.1 Detailed Description

The scene options.

Definition at line 14 of file scene_options.hpp.

6.26.2 Member Data Documentation

6.26.2.1 action_labels

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

The action labels.

Definition at line 33 of file scene_options.hpp.

6.26.2.2 action_selection

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

The currently selected actions for each mode.

Definition at line 38 of file scene_options.hpp.

6.26.2.3 max_size

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

The maximum size of the container.

Definition at line 18 of file scene_options.hpp.

6.26.2.4 mode_labels

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

The mode labels.

Definition at line 23 of file scene_options.hpp.

6.26.2.5 mode_selection

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

The currently selected mode.

Definition at line 28 of file scene_options.hpp.

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

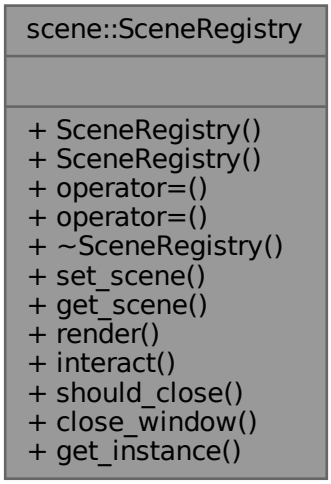
- src/scene/scene_options.hpp

6.27 scene::SceneRegistry Class Reference

The scene registry.

```
#include <scene_registry.hpp>
```

Collaboration diagram for scene::SceneRegistry:



Public Member Functions

- SceneRegistry (const SceneRegistry &)=delete

  Deleted copy constructor.
- SceneRegistry (SceneRegistry &&)=delete

  Deleted move constructor.
- SceneRegistry & operator= (const SceneRegistry &)=delete

  Deleted copy assignment operator.
- SceneRegistry & operator= (SceneRegistry &&)=delete

  Deleted move assignment operator.
- ~SceneRegistry ()=default

  Destroy the Scene Registry object.
- void set_scene (SceneId scene_type)

  Sets the scene.
- SceneId get_scene () const

  Gets the scene.
- void render ()

  Renders the scene.
- void interact ()

  Interacts with the scene.
- bool should_close () const

  Checks if the window should close.
- void close_window ()

  Closes the window.

Static Public Member Functions

- static SceneRegistry & get_instance ()

  Get the instance object.

6.27.1 Detailed Description

The scene registry.

Definition at line 22 of file scene_registry.hpp.

6.27.2 Constructor & Destructor Documentation

6.27.2.1 SceneRegistry() [1/2]

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

Deleted copy constructor.

6.27.2.2 SceneRegistry() [2/2]

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

Deleted move constructor.

6.27.2.3 ~SceneRegistry()

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

Destroy the Scene Registry object.

6.27.3 Member Function Documentation

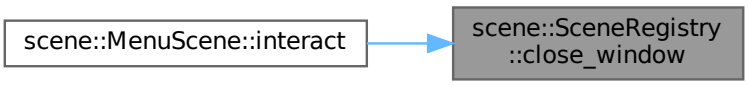
6.27.3.1 close_window()

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

Closes the window.

Definition at line 25 of file scene_registry.cpp.

Here is the caller graph for this function:



6.27.3.2 get_instance()

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

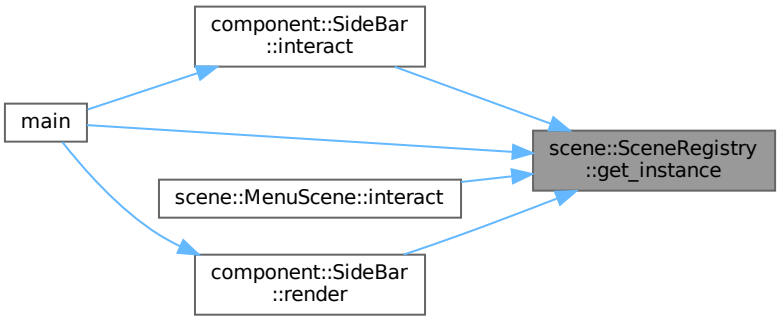
Get the instance object.

Returns

SceneRegistry& The instance

Definition at line 7 of file scene_registry.cpp.

Here is the caller graph for this function:



6.27.3.3 get_scene()

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

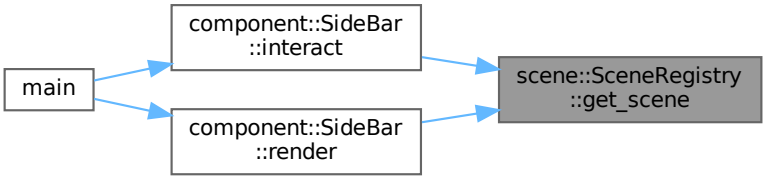
Gets the scene.

Returns

SceneId The scene

Definition at line 17 of file scene_registry.cpp.

Here is the caller graph for this function:



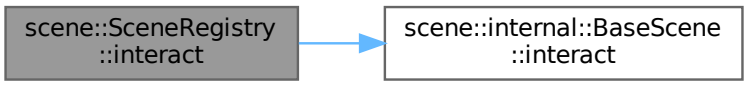
6.27.3.4 interact()

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

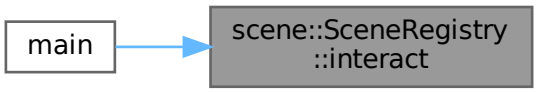
Interacts with the scene.

Definition at line 21 of file scene_registry.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



6.27.3.5 operator=() [1/2]

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

Deleted copy assignment operator.

6.27.3.6 operator=() [2/2]

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

Deleted move assignment operator.

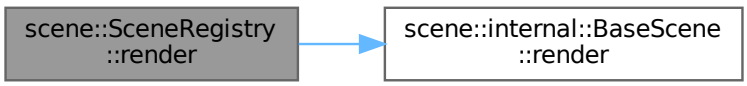
6.27.3.7 render()

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

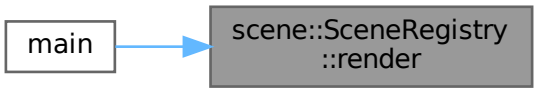
Renders the scene.

Definition at line 19 of file scene_registry.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



6.27.3.8 set_scene()

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

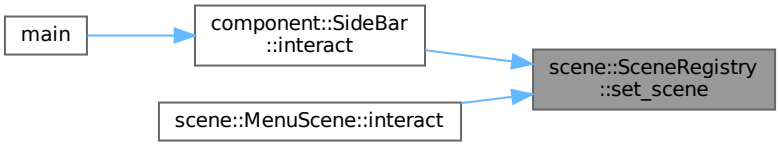
Sets the scene.

Parameters

scene_type	The scene type

Definition at line 12 of file scene_registry.cpp.

Here is the caller graph for this function:



6.27.3.9 should_close()

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

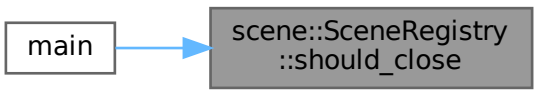
Checks if the window should close.

Return values

true	If the window should close
false	If the window should not close

Definition at line 23 of file scene_registry.cpp.

Here is the caller graph for this function:



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

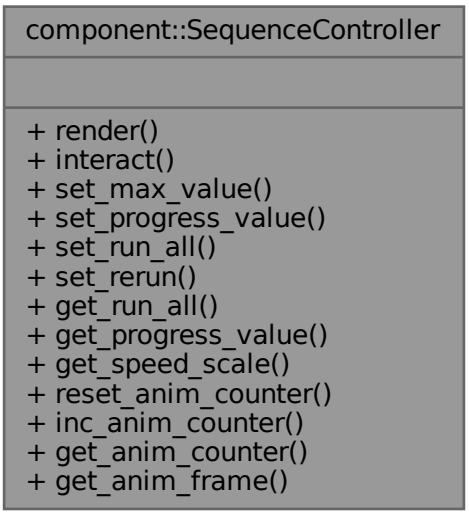
- src/scene/scene_registry.hpp
- src/scene/scene_registry.cpp

6.28 component::SequenceController Class Reference

Controls the display of frames of the animation sequence.

`#include <sequence_controller.hpp>`

Collaboration diagram for component::SequenceController:



Public Member Functions

- void render ()

  Renders the sequence controller.
- bool interact ()

  Checks if the sequence controller was interacted with.
- void set_max_value (int num)

  Sets the maximum number of steps that can be accessed.
- void set_progress_value (int value)

  Sets the value of the progress bar.
- void set_run_all (bool run_all)

  Sets the sequence to be in the run-all-at-once mode.
- void set_rerun ()

  Sets the sequence to rerun from the beginning.
- bool get_run_all () const

  Check if the sequence is in the run-all-at-once mode.
- int get_progress_value () const

  Gets the value of the progress bar.
- float get_speed_scale () const

  Gets the speed scale of the sequence.
- void reset_anim_counter ()

  Resets the animation counter.
- void inc_anim_counter ()

  Increments the animation counter.
- int get_anim_counter () const

  Gets the animation counter.
- int get_anim_frame () const

  Gets the animated frame.

6.28.1 Detailed Description

Controls the display of frames of the animation sequence.

Definition at line 12 of file sequence_controller.hpp.

6.28.2 Member Function Documentation

6.28.2.1 get_anim_counter()

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

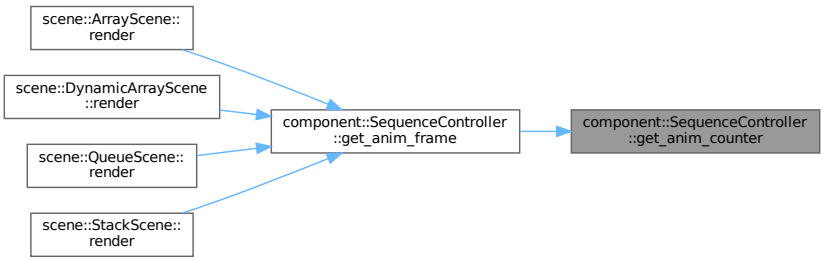
Gets the animation counter.

Returns

the animation counter

Definition at line 35 of file sequence_controller.cpp.

Here is the caller graph for this function:



6.28.2.2 get_anim_frame()

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

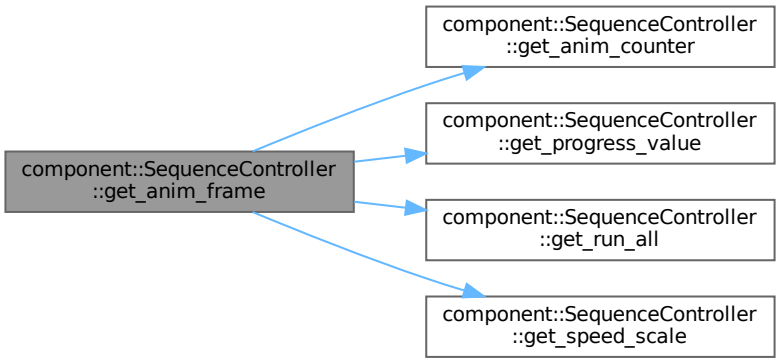
Gets the animated frame.

Returns

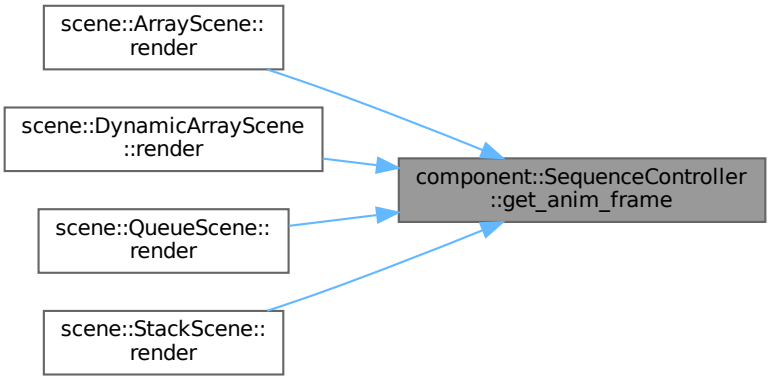
the animated frame

Definition at line 42 of file sequence_controller.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



6.28.2.3 get_progress_value()

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

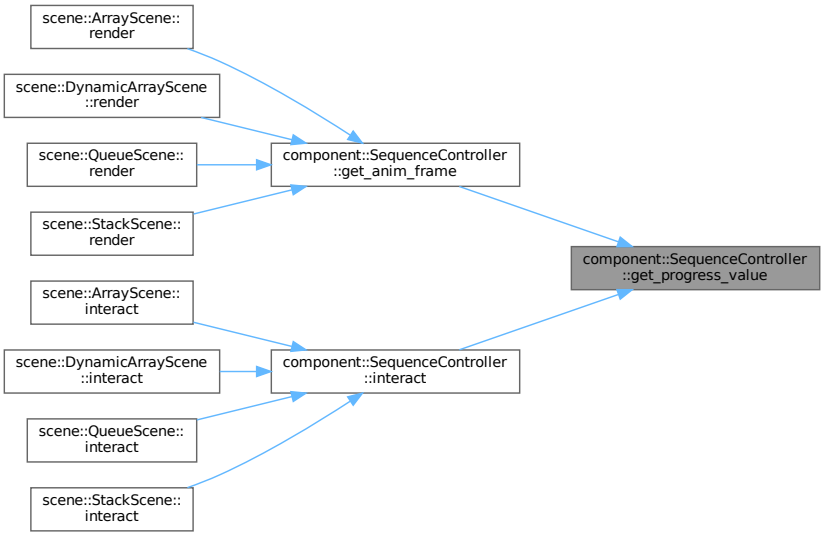
Gets the value of the progress bar.

Returns

the value of the progress bar

Definition at line 21 of file sequence_controller.cpp.

Here is the caller graph for this function:



6.28.2.4 get_run_all()

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

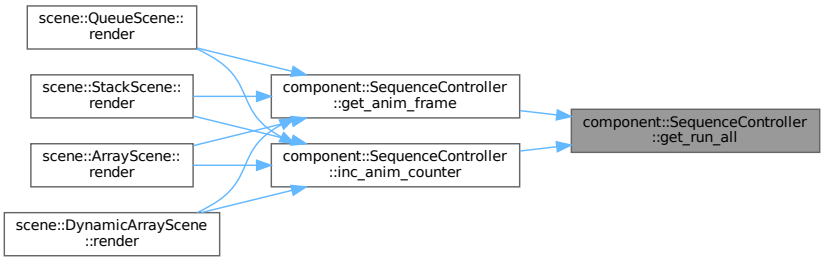
Check if the sequence is in the run-all-at-once mode.

Return values

true	The sequence is in the run-all-at-once mode
false	The sequence is not in the run-all-at-once mode

Definition at line 19 of file sequence_controller.cpp.

Here is the caller graph for this function:



6.28.2.5 get_speed_scale()

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

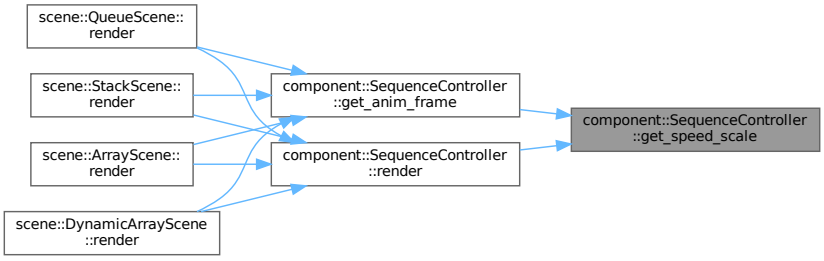
Gets the speed scale of the sequence.

Returns

the speed scale of the sequence

Definition at line 23 of file sequence_controller.cpp.

Here is the caller graph for this function:



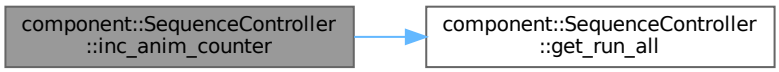
6.28.2.6 inc_anim_counter()

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

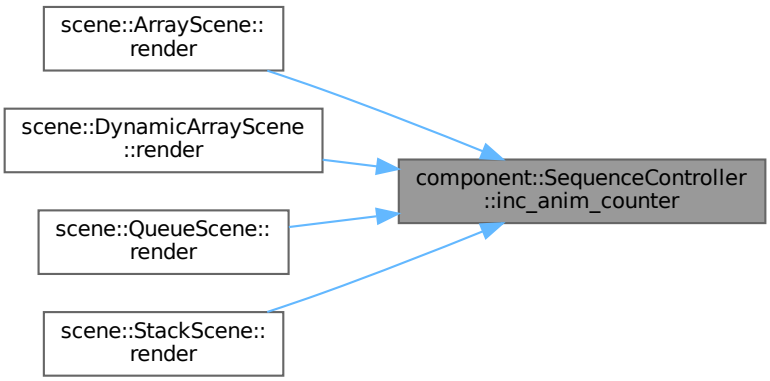
Increments the animation counter.

Definition at line 29 of file sequence_controller.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



6.28.2.7 interact()

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

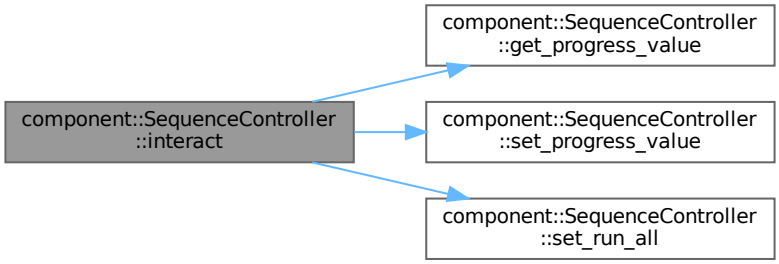
Checks if the sequence controller was interacted with.

Return values

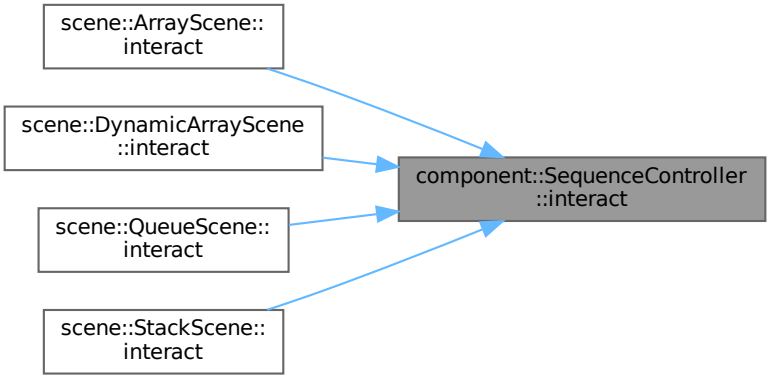
true	The sequence controller was interacted with
false	The sequence controller was not interacted with

Definition at line 90 of file sequence_controller.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



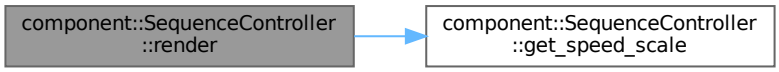
6.28.2.8 render()

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

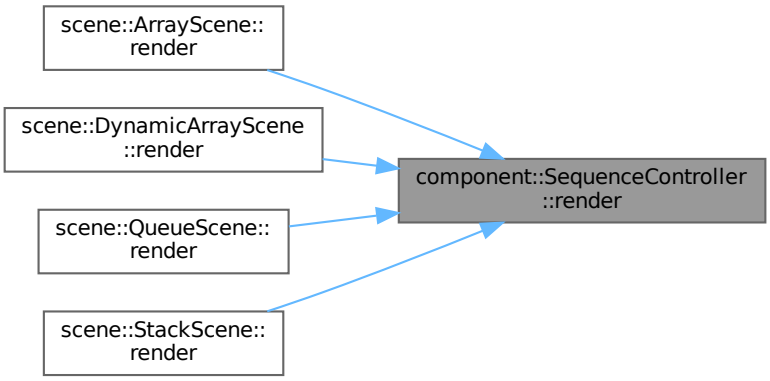
Renders the sequence controller.

Definition at line 51 of file sequence_controller.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



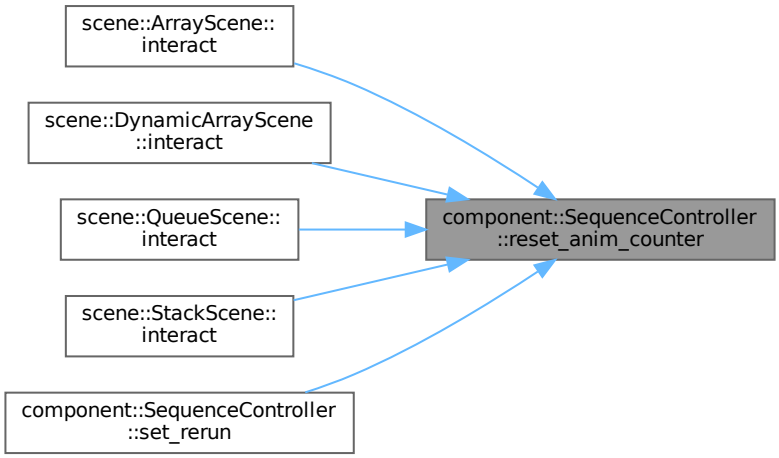
6.28.2.9 reset_anim_counter()

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

Resets the animation counter.

Definition at line 27 of file sequence_controller.cpp.

Here is the caller graph for this function:



6.28.2.10 set_max_value()

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

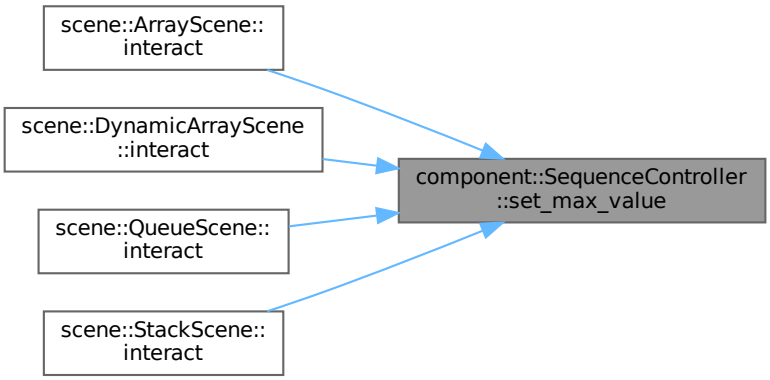
Sets the maximum number of steps that can be accessed.

Parameters

num	the maximum number of steps that can be accessed

Definition at line 11 of file sequence_controller.cpp.

Here is the caller graph for this function:



6.28.2.11 set_progress_value()

```
void component::SequenceController::set_progress_value (
            int value )
```

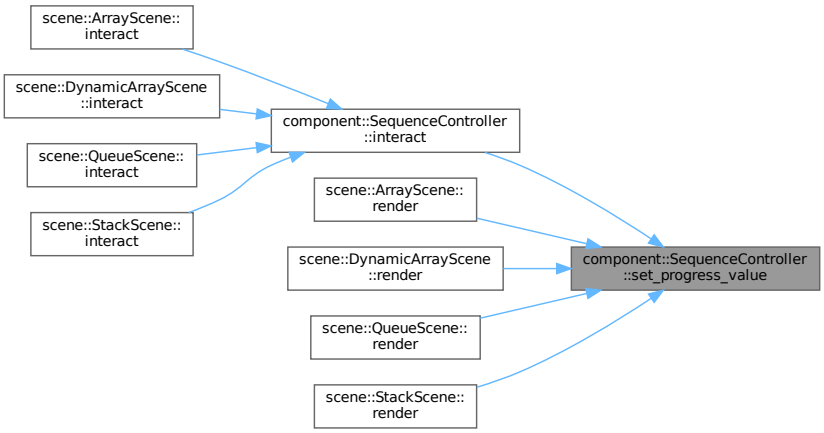
Sets the value of the progress bar.

Parameters

value	the value of the progress bar

Definition at line 13 of file sequence_controller.cpp.

Here is the caller graph for this function:



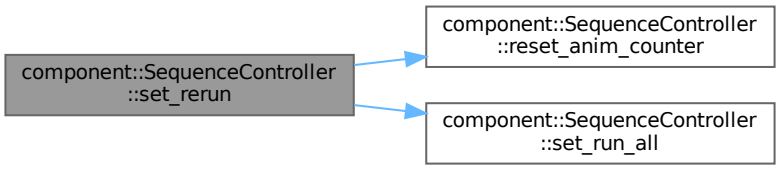
6.28.2.12 set_rerun()

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

Sets the sequence to rerun from the beginning.

Definition at line 37 of file sequence_controller.cpp.

Here is the call graph for this function:



6.28.2.13 set_run_all()

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

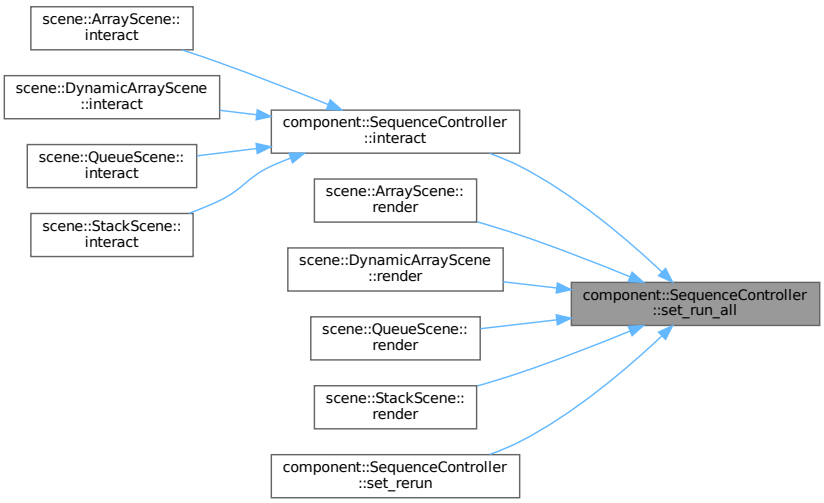
Sets the sequence to be in the run-all-at-once mode.

Parameters

run_all	true if the sequence is in the run-all-at-once mode, false otherwise

Definition at line 17 of file sequence_controller.cpp.

Here is the caller graph for this function:



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

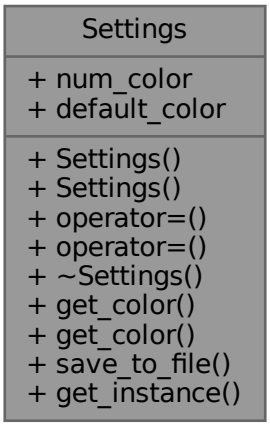
- src/component/sequence_controller.hpp
- src/component/sequence_controller.cpp

6.29 Settings Class Reference

The settings.

`#include <settings.hpp>`

Collaboration diagram for Settings:



Public Member Functions

- Settings (const Settings &)=delete
  Deleted copy constructor.
- Settings (Settings &&)=delete
  Deleted move constructor.
- Settings & operator= (const Settings &)=delete
  Deleted copy assignment operator.
- Settings & operator= (Settings &&)=delete
  Deleted move assignment operator.
- ~Settings ()
  Destructor.
- Color & get_color (std::size_t index)
  Gets the color.
- Color get_color (std::size_t index) const
  Gets the color.
- void save_to_file (const std::string &path)
  Saves the settings to a file.

Static Public Member Functions

- static Settings & get_instance ()
  Gets the instance.

Static Public Attributes

- static constexpr int num_color = 9
  The number of colors.
- static constexpr std::array< unsigned, num_color > default_color
  The default colors.

6.29.1 Detailed Description

The settings.

Definition at line 13 of file settings.hpp.

6.29.2 Constructor & Destructor Documentation

6.29.2.1 Settings() [1/2]

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

Deleted copy constructor.

6.29.2.2 Settings() [2/2]

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

Deleted move constructor.

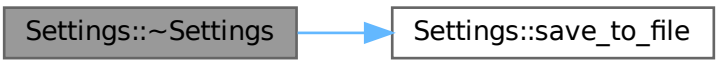
6.29.2.3 ~Settings()

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

Destructor.

Definition at line 24 of file settings.cpp.

Here is the call graph for this function:



6.29.3 Member Function Documentation

6.29.3.1 get_color() [1/2]

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

Gets the color.

Parameters

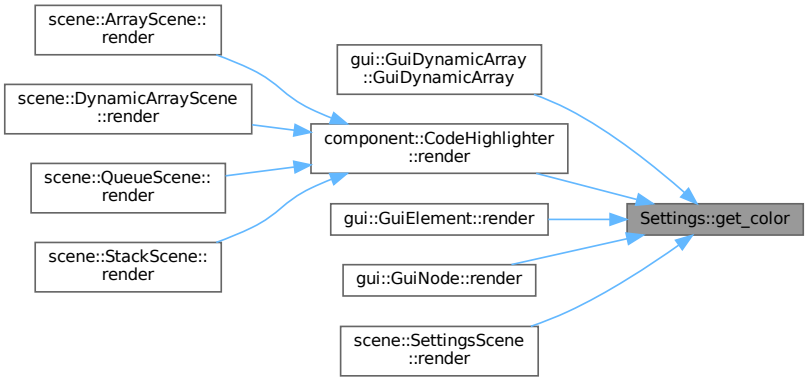
index	The index of the color

Returns

Color& The color

Definition at line 26 of file settings.cpp.

Here is the caller graph for this function:



6.29.3.2 get_color() [2/2]

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

Gets the color.

Parameters

index	The index of the color

Returns

Color The color

Definition at line 28 of file settings.cpp.

6.29.3.3 get_instance()

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

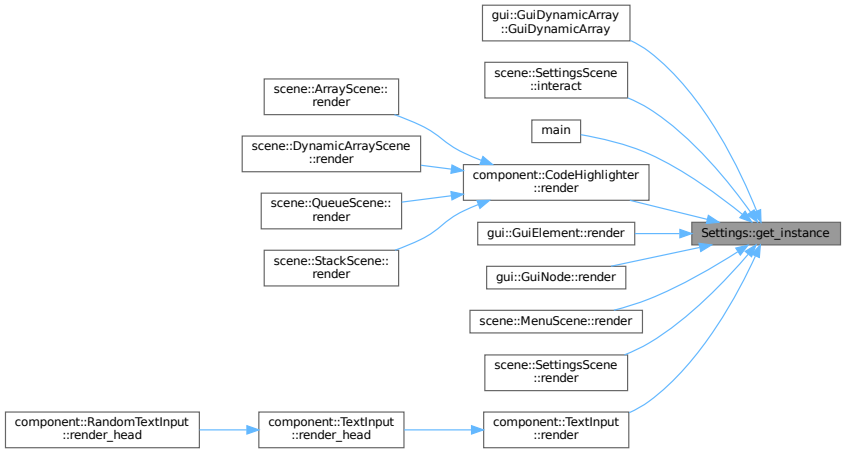
Gets the instance.

Returns

Settings& The instance

Definition at line 10 of file settings.cpp.

Here is the caller graph for this function:



6.29.3.4 operator=() [1/2]

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

Deleted copy assignment operator.

6.29.3.5 operator=() [2/2]

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

Deleted move assignment operator.

6.29.3.6 save_to_file()

```
void Settings::save_to_file (
            const std::string & path )
```

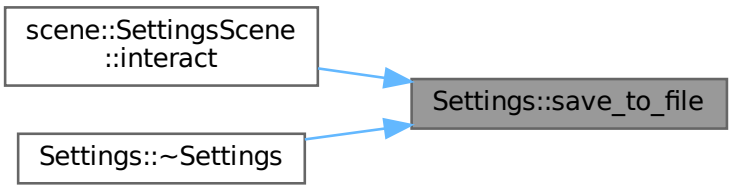
Saves the settings to a file.

Parameters

path	The path to the file

Definition at line 15 of file settings.cpp.

Here is the caller graph for this function:



6.29.4 Member Data Documentation

6.29.4.1 default_color

```
constexpr std::array<unsigned, num_color> Settings::default_color  [static], [constexpr]
```

Initial value:

```
{{
        0x00000000,
        0x82828200,
        0xffa10000,
        0x00e43000,
        0x873cbe00,
        0xe6293700,
        0x0079f100,
        0xff6dc200,
        0xf5f5f500,
    }}
```

The default colors.

Definition at line 23 of file settings.hpp.

6.29.4.2 num_color

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

The number of colors.

Definition at line 18 of file settings.hpp.

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

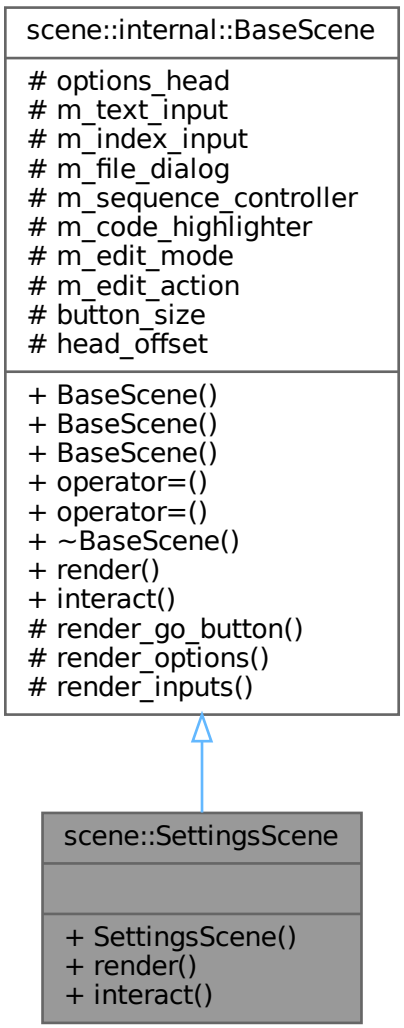
- src/settings.hpp
- src/settings.cpp

6.30 scene::SettingsScene Class Reference

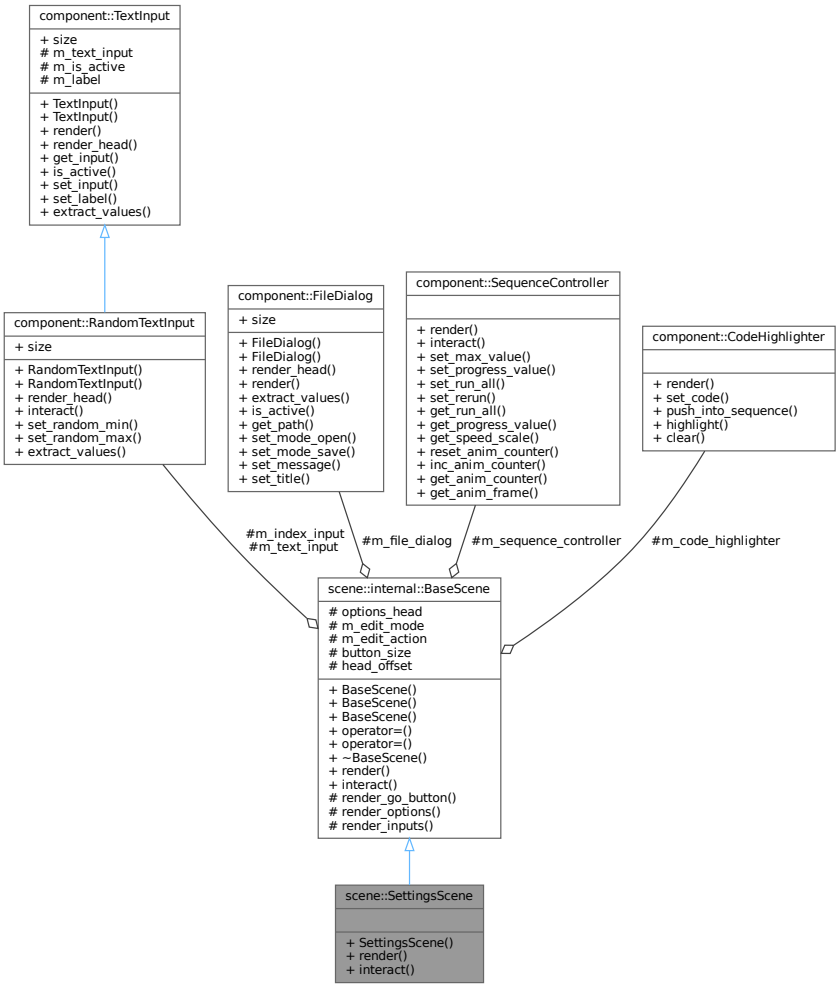
The settings scene.

`#include <settings_scene.hpp>`

Inheritance diagram for scene::SettingsScene:



Collaboration diagram for scene::SettingsScene:



Public Member Functions

- SettingsScene ()

  Construct a new SettingsScene object.
- void render () override

  Renders the scene.
- void interact () override

  Interacts with the scene.

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

- BaseScene ()=default

  Construct a new BaseScene object.
- BaseScene (const BaseScene &)=delete

  Copy constructor (deleted)
- BaseScene (BaseScene &&)=delete

Move constructor (deleted)

- BaseScene & operator= (const BaseScene &)=delete

  Copy assignment (deleted)
- BaseScene & operator= (BaseScene &&)=delete

  Move assignment (deleted)
- virtual ~BaseScene ()=default

  Destroy the BaseScene object.
- virtual void render ()

  Renders the scene.
- virtual void interact ()

  Interacts with the scene.

Additional Inherited Members

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

- virtual bool render_go_button () const

  Renders the go button.
- virtual void render_options (SceneOptions &scene_config)

  Renders the options.
- virtual void render_inputs ()

  Renders the inputs.

Protected Attributes inherited from scene::internal::BaseScene

- float options_head {}

  The head of the options.
- component::RandomTextInput m_text_input {"value"}

  The text input for the value.
- component::RandomTextInput m_index_input {"index"}

  The text input for the index.
- component::FileDialog m_file_dialog

  The file dialog.
- component::SequenceController m_sequence_controller

  The sequence controller.
- component::CodeHighlighter m_code_highlighter

  The code highlighter.
- bool m_edit_mode {}

  Whether the edit mode is enabled.
- bool m_edit_action {}

  Whether the edit action is enabled.

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

- static constexpr Vector2 button_size {200, 50}

  The size of the buttons.
- static constexpr int head_offset = 20

  The offset of the widgets.

6.30.1 Detailed Description

The settings scene.

Definition at line 20 of file settings_scene.hpp.

6.30.2 Constructor & Destructor Documentation

6.30.2.1 SettingsScene()

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

Construct a new SettingsScene object.

Definition at line 47 of file settings_scene.cpp.

6.30.3 Member Function Documentation

6.30.3.1 interact()

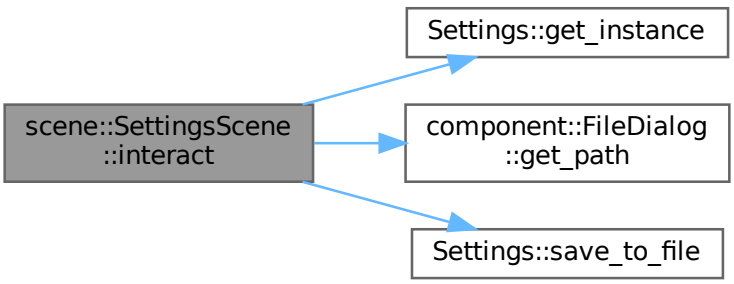
```
void scene::SettingsScene::interact ( )  [override], [virtual]
```

Interacts with the scene.

Reimplemented from scene::internal::BaseScene.

Definition at line 144 of file settings_scene.cpp.

Here is the call graph for this function:



6.30.3.2 render()

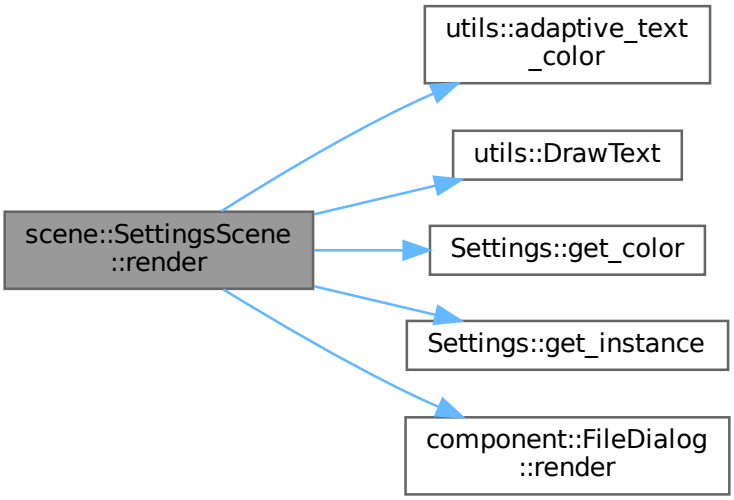
```
void scene::SettingsScene::render ( )  [override], [virtual]
```

Renders the scene.

Reimplemented from scene::internal::BaseScene.

Definition at line 70 of file settings_scene.cpp.

Here is the call graph for this function:



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

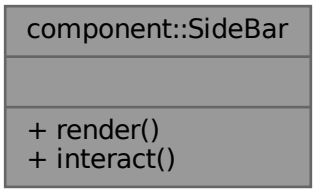
- src/scene/settings_scene.hpp
- src/scene/settings_scene.cpp

6.31 component::SideBar Class Reference

"Side bar" for extra navigation

```
#include <sidebar.hpp>
```

Collaboration diagram for component::SideBar:



Public Member Functions

- void render ()

  Constructs a SideBar object.
- void interact ()

  Interacts with the SideBar object.

6.31.1 Detailed Description

"Side bar" for extra navigation

Note

Should have been renamed to navbar instead

Definition at line 16 of file sidebar.hpp.

6.31.2 Member Function Documentation

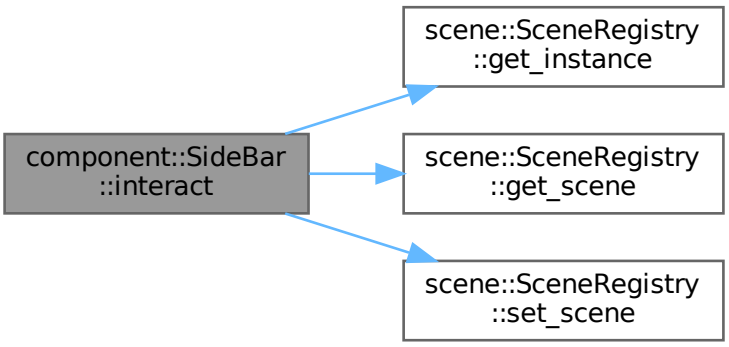
6.31.2.1 interact()

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

Interacts with the SideBar object.

Definition at line 53 of file sidebar.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

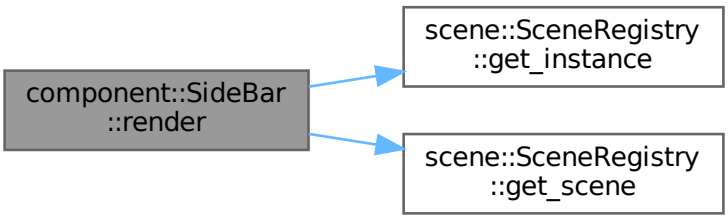
6.31.2.2 render()

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

Constructs a SideBar object.

Definition at line 12 of file sidebar.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

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

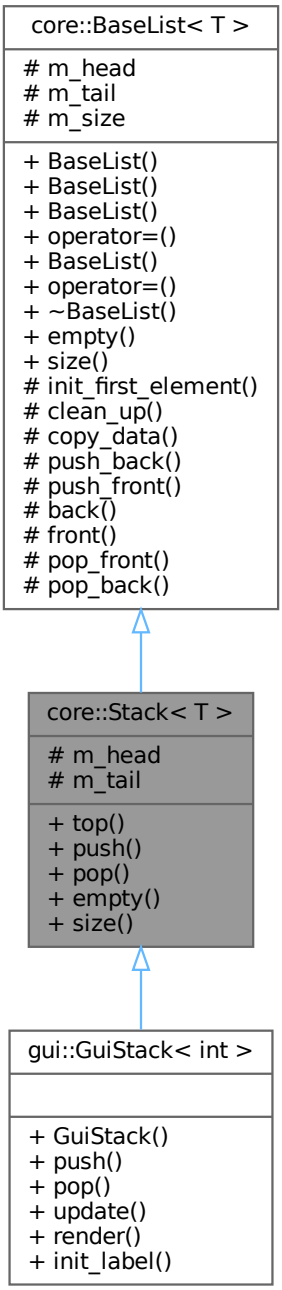
- src/component/sidebar.hpp
- src/component/sidebar.cpp

6.32 core::Stack< T > Class Template Reference

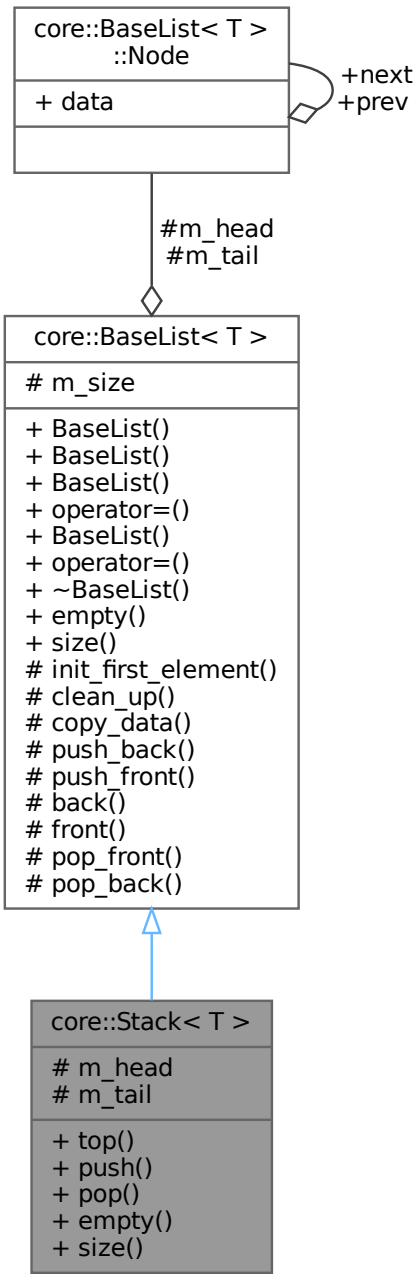
The stack container.

`#include <stack.hpp>`

Inheritance diagram for core::Stack< T >:



Collaboration diagram for core::Stack< T >:



Public Member Functions

- T & top () const

  Returns the reference to the top element of the stack.
- void push (const T &elem)

  Inserts the element at the top of the stack.
- void pop ()

Removes the top element of the stack.

- bool empty () const

  Check whether the container is empty.

- std::size_t size () const

  Returns the size of the container.

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

- BaseList ()=default

  Default constructor.

- BaseList (std::initializer_list< T > init_list)

  Constructs the container with the contents of the initializer list.

- BaseList (const BaseList &rhs)

  Copy constructor.

- BaseList & operator= (const BaseList &rhs)

  Copy assignment operator.

- BaseList (BaseList &&rhs) noexcept

  Move constructor.

- BaseList & operator= (BaseList &&rhs) noexcept

  Move assignment operator.

- ~BaseList ()

  Destructor.

- bool empty () const

  Check whether the container is empty.

- std::size_t size () const

  Returns the size of the container.

Protected Attributes

- Node_ptr m_head

  The head of the list.

- Node_ptr m_tail

  The tail of the list.

Protected Attributes inherited from core::BaseList< T >

- Node_ptr m_head {nullptr}

  The head of the list.

- Node_ptr m_tail {nullptr}

  The tail of the list.

- std::size_t m_size {}

  The size of the list.

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)

  Initializes the first element of the container.
- void clean_up ()

  Frees all elements in the container.
- void copy_data (const BaseList &rhs)

  Copies data from another container.
- void push_back (const T &elem)

  Pushes the element to the back of the container.
- void push_front (const T &elem)

  Pushes the element to the front of the container.
- T & back () const

  Returns the reference to the element at the back of the container.
- T & front () const

  Returns the reference to the element at the front of the container.
- void pop_front ()

  Removes the element at the back of the container.
- void pop_back ()

  Removes the element at the front of the container.

6.32.1 Detailed Description

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

The stack container.

Template Parameters

T	the type of the elements

Definition at line 14 of file stack.hpp.

6.32.2 Member Function Documentation

6.32.2.1 empty()

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

Check whether the container is empty.

Return values

true	The container is empty
false	The container is not empty

Definition at line 63 of file base_list.hpp.

6.32.2.2 pop()

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

Removes the top element of the stack.

Definition at line 57 of file stack.hpp.

6.32.2.3 push()

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

Inserts the element at the top of the stack.

Parameters

elem	The element to insert

Definition at line 52 of file stack.hpp.

6.32.2.4 size()

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

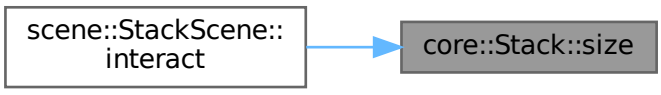
Returns the size of the container.

Returns

The size of the container

Definition at line 69 of file base_list.hpp.

Here is the caller graph for this function:



6.32.2.5 top()

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

Returns the reference to the top element of the stack.

Returns

T& the reference to the top element of the stack

Definition at line 47 of file stack.hpp.

6.32.3 Member Data Documentation

6.32.3.1 m_head

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

The head of the list.

Definition at line 87 of file base_list.hpp.

6.32.3.2 m_tail

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

The tail of the list.

Definition at line 92 of file base_list.hpp.

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

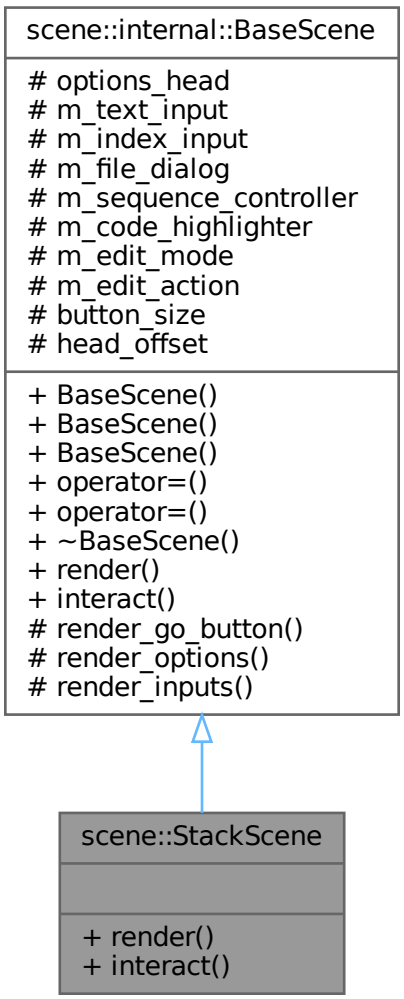
- src/core/stack.hpp

6.33 scene::StackScene Class Reference

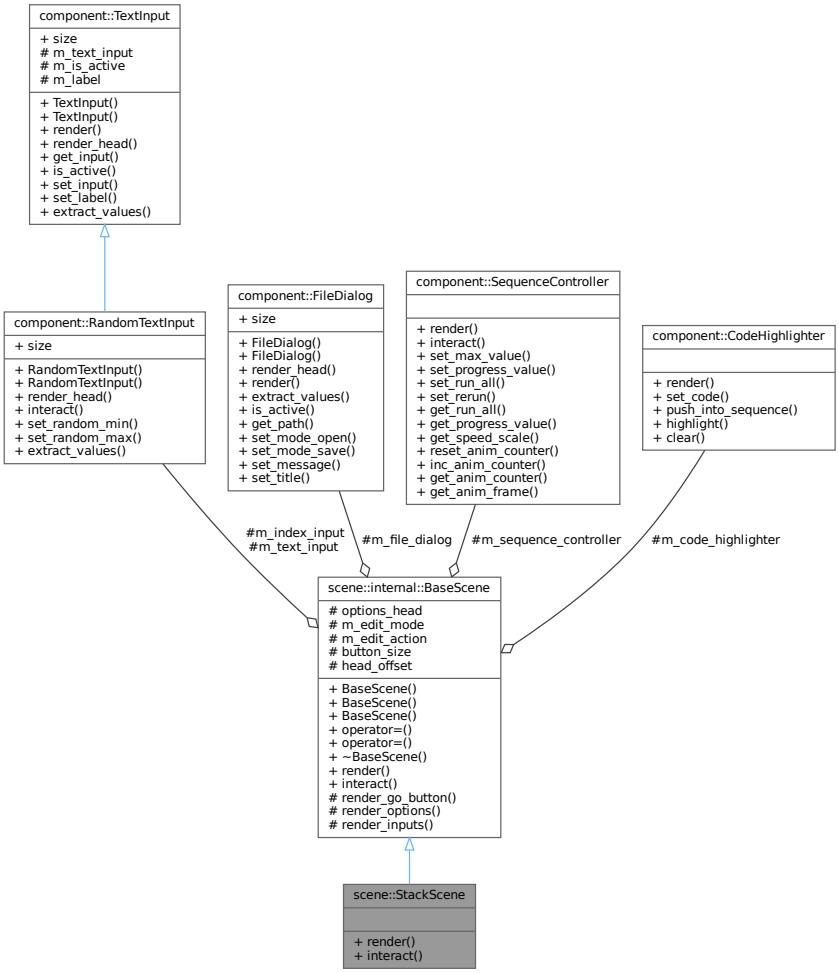
The stack scene.

`#include <stack_scene.hpp>`

Inheritance diagram for scene::StackScene:



Collaboration diagram for scene::StackScene:



Public Member Functions

- void render () override

  Renders the scene.
- void interact () override

  Interacts with the scene.

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

- BaseScene ()=default

  Construct a new BaseScene object.
- BaseScene (const BaseScene &)=delete

  Copy constructor (deleted)
- BaseScene (BaseScene &&)=delete

  Move constructor (deleted)
- BaseScene & operator= (const BaseScene &)=delete

  Copy assignment (deleted)

- BaseScene & operator= (BaseScene &&)=delete

  Move assignment (deleted)
- virtual ~BaseScene ()=default

  Destroy the BaseScene object.
- virtual void render ()

  Renders the scene.
- virtual void interact ()

  Interacts with the scene.

Additional Inherited Members

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

- virtual bool render_go_button () const

  Renders the go button.
- virtual void render_options (SceneOptions &scene_config)

  Renders the options.
- virtual void render_inputs ()

  Renders the inputs.

Protected Attributes inherited from scene::internal::BaseScene

- float options_head {}

  The head of the options.
- component::RandomTextInput m_text_input {"value"}

  The text input for the value.
- component::RandomTextInput m_index_input {"index"}

  The text input for the index.
- component::FileDialog m_file_dialog

  The file dialog.
- component::SequenceController m_sequence_controller

  The sequence controller.
- component::CodeHighlighter m_code_highlighter

  The code highlighter.
- bool m_edit_mode {}

  Whether the edit mode is enabled.
- bool m_edit_action {}

  Whether the edit action is enabled.

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

- static constexpr Vector2 button_size {200, 50}

  The size of the buttons.
- static constexpr int head_offset = 20

  The offset of the widgets.

6.33.1 Detailed Description

The stack scene.

Definition at line 17 of file stack_scene.hpp.

6.33.2 Member Function Documentation

6.33.2.1 interact()

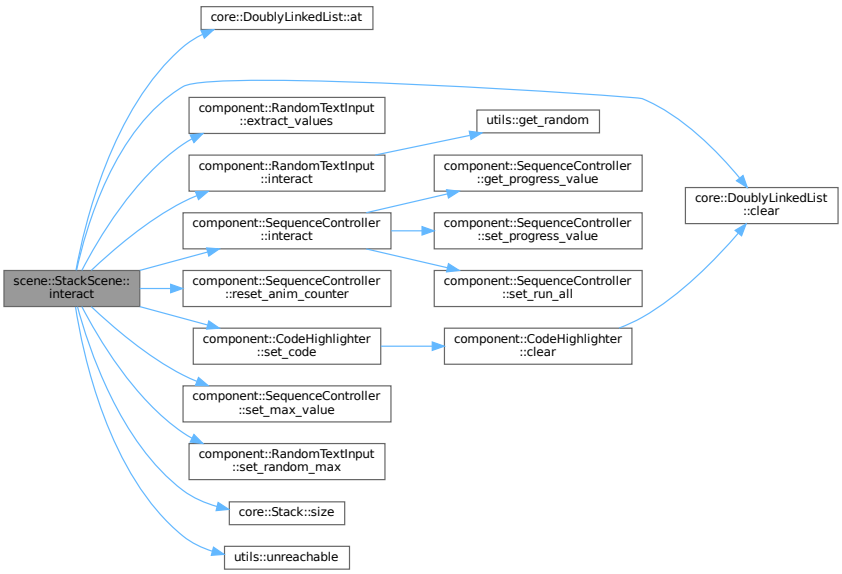
```
void scene::StackScene::interact ( )  [override], [virtual]
```

Interacts with the scene.

Reimplemented from scene::internal::BaseScene.

Definition at line 72 of file stack_scene.cpp.

Here is the call graph for this function:



6.33.2.2 render()

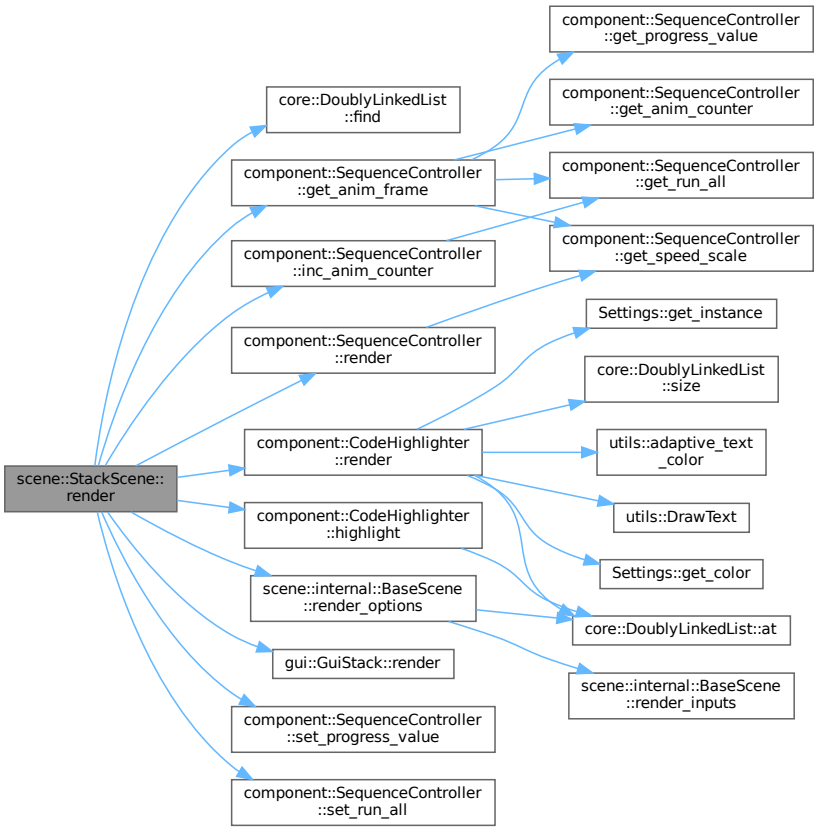
```
void scene::StackScene::render ( )  [override], [virtual]
```

Renders the scene.

Reimplemented from scene::internal::BaseScene.

Definition at line 17 of file stack_scene.cpp.

Here is the call graph for this function:



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

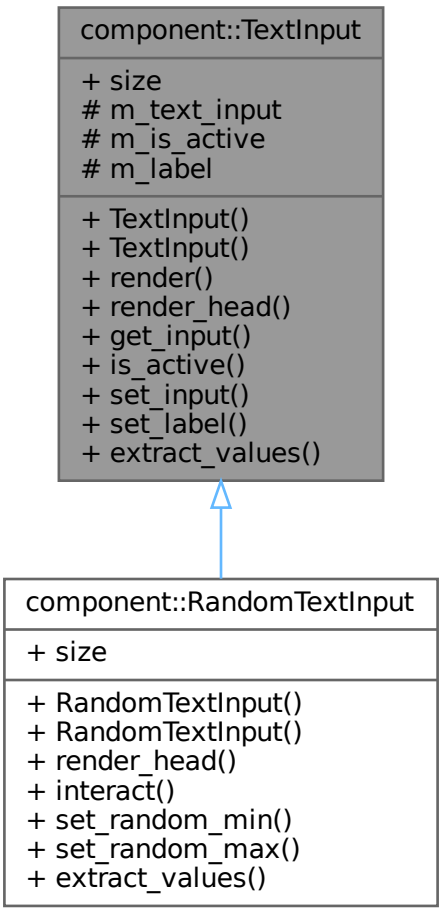
- src/scene/stack_scene.hpp
- src/scene/stack_scene.cpp

6.34 component::TextInput Class Reference

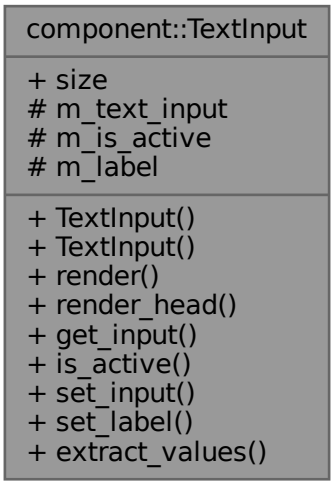
Input for entering text.

`#include <text_input.hpp>`

Inheritance diagram for component::TextInput:



Collaboration diagram for component::TextInput:



Public Member Functions

- TextInput ()=default

  Constructs a TextInput object.
- TextInput (const char *label)

  Constructs a TextInput object.
- void render (float x, float y)

  Renders the text input.
- void render_head (float &options_head, float head_offset)

  Renders the text input, updates the head position with offset.
- std::string get_input () const

  Returns the input of the text input.
- bool is_active () const

  Checks if the text input is active.
- void set_input (const char *input, int len)

  Sets the input of the text input.
- void set_label (const char *const label)

  Sets the label of the text input.
- core::Deque< int > extract_values ()

  Extracts the values from the text input.

Static Public Attributes

- static constexpr Vector2 size {200, 50}

  The size of the text input.

Protected Attributes

- char m_text_input [constants::text_buffer_size] = ""

  The text input.
- bool m_is_active {}

  Whether the text input is active.
- const char * m_label {}

  The label of the text input.

6.34.1 Detailed Description

Input for entering text.

Definition at line 16 of file text_input.hpp.

6.34.2 Constructor & Destructor Documentation

6.34.2.1 TextInput() [1/2]

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

Constructs a TextInput object.

6.34.2.2 TextInput() [2/2]

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

Constructs a TextInput object.

Parameters

label	the label of the text input

Definition at line 14 of file text_input.cpp.

6.34.3 Member Function Documentation

6.34.3.1 extract_values()

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

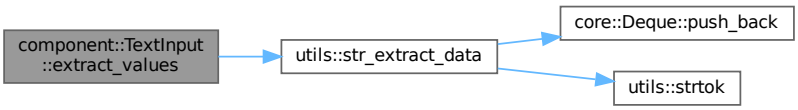
Extracts the values from the text input.

Returns

the values from the text input

Definition at line 48 of file text_input.cpp.

Here is the call graph for this function:



6.34.3.2 get_input()

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

Returns the input of the text input.

Returns

the input of the text input

Definition at line 38 of file text_input.cpp.

6.34.3.3 is_active()

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

Checks if the text input is active.

Return values

true	The text input is active
false	The text input is not active

Definition at line 40 of file text_input.cpp.

6.34.3.4 render()

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

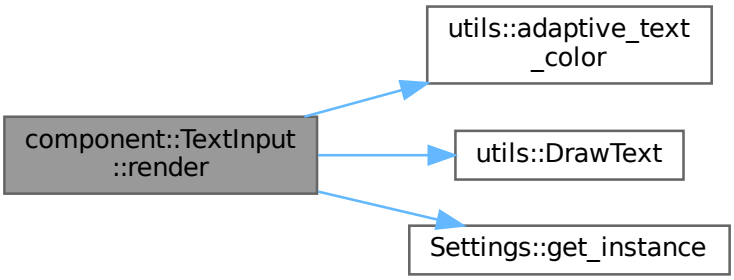
Renders the text input.

Parameters

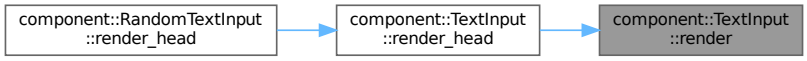
x	the x position of the text input
y	the y position of the text input

Definition at line 16 of file text_input.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



6.34.3.5 render_head()

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

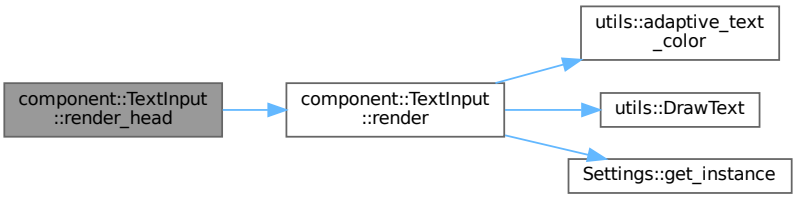
Renders the text input, updates the head position with offset.

Parameters

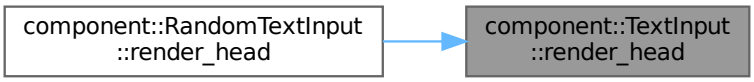
options_head	the head position of the options
head_offset	the offset of the head position

Definition at line 33 of file text_input.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



6.34.3.6 set_input()

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

Sets the input of the text input.

Parameters

input	the input of the text input
len	the length of the input

Definition at line 44 of file text_input.cpp.

6.34.3.7 set_label()

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

Sets the label of the text input.

Parameters

label	the label of the text input

Definition at line 42 of file text_input.cpp.

6.34.4 Member Data Documentation

6.34.4.1 m_is_active

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

Whether the text input is active.

Definition at line 89 of file text_input.hpp.

6.34.4.2 m_label

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

The label of the text input.

Definition at line 94 of file text_input.hpp.

6.34.4.3 m_text_input

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

The text input.

Definition at line 84 of file text_input.hpp.

6.34.4.4 size

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

The size of the text input.

Definition at line 21 of file text_input.hpp.

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

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

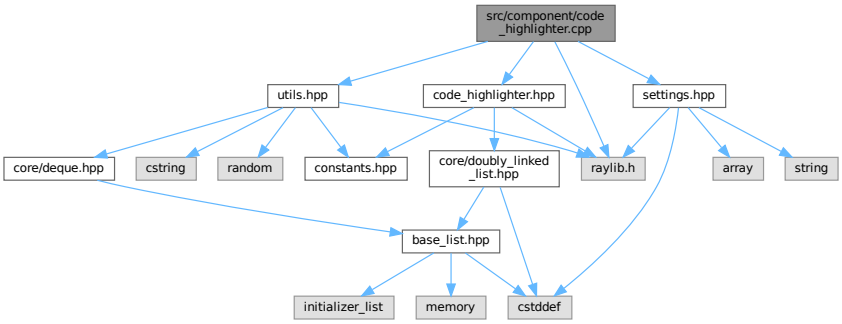
Chapter 7

File Documentation

7.1 src/component/code_highlighter.cpp File Reference

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

Include dependency graph for code_highlighter.cpp:



Namespaces

- namespace component

7.2 code_highlighter.cpp

Go to the documentation of this file.

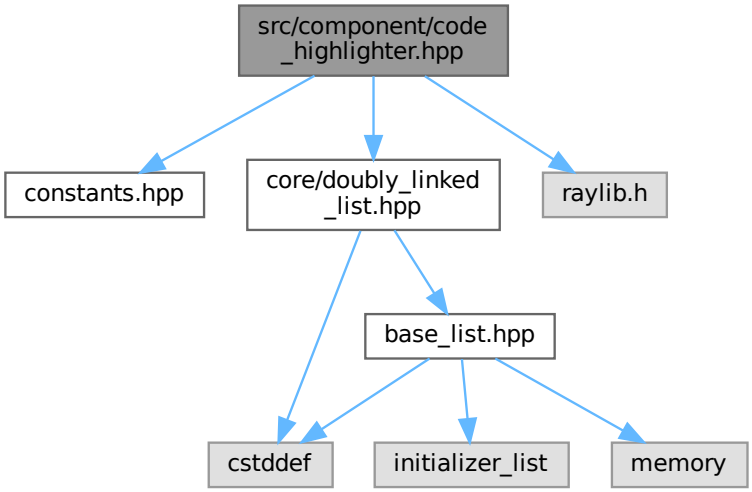
```
00001 #include "code_highlighter.hpp"
00002
00003 #include "raylib.h"
00004 #include "settings.hpp"
00005 #include "utils.hpp"
00006
00007 namespace component {
00008
```

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

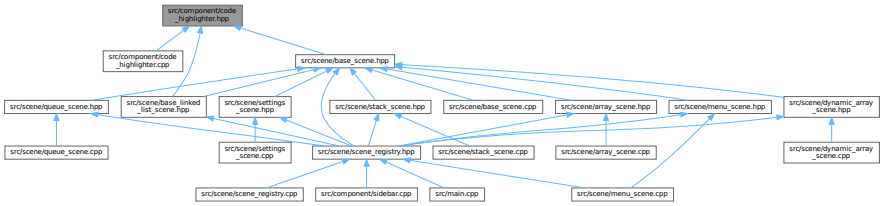
7.3 src/component/code_highlighter.hpp File Reference

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

Include dependency graph for code_highlighter.hpp:



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



Classes

- class component::CodeHighlighter

  Code highlighter that highlights the source code on each step.

Namespaces

- namespace component

7.4 code_highlighter.hpp

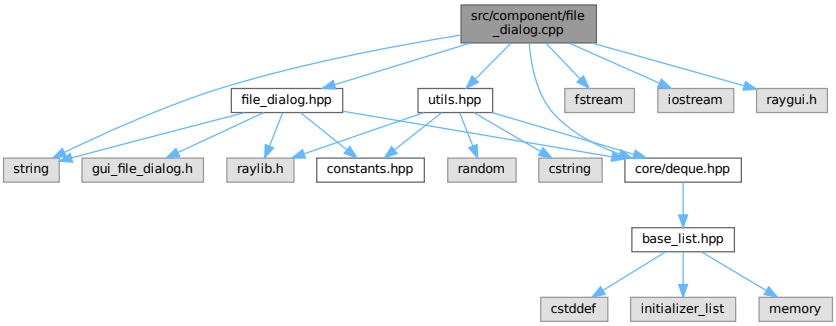
Go to the documentation of this file.

```
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
00014 class CodeHighlighter {
00015 public:
00019     void render();
00020
00025     void set_code(core::DoublyLinkedList<const char*>&& src_code);
00026
00031     void push_into_sequence(int line_number);
00032
00037     void highlight(int frame_idx);
00038
00042     void clear();
00043
00044 private:
00048     static constexpr int width = 400;
00049
00053     static constexpr int height = 30;
00054
00058     static constexpr Vector2 head_pos{constants::scene_width - width,
00059                                       2.5F * height};
00060
00064     core::DoublyLinkedList<const char*> m_src_code;
00065
00069     core::DoublyLinkedList<int> m_sequence;
00070
00074     int m_highlighted_line{-1};
00075 };
00076
00077 }  // namespace component
00078
00079 #endif  // COMPONENT_CODE_HIGHLIGHTER_HPP_
```

7.5 src/component/file_dialog.cpp File Reference

```
#include "file_dialog.hpp"
#include <fstream>
#include <iostream>
#include <string>
#include "core/deque.hpp"
#include "raygui.h"
#include "utils.hpp"
```

Include dependency graph for file_dialog.cpp:



Namespaces

- namespace component

7.6 file_dialog.cpp

Go to the documentation of this file.

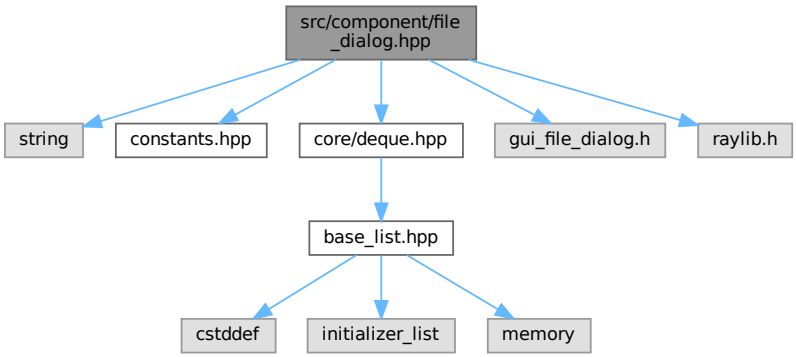
```
00001 #include "file_dialog.hpp"
00002
00003 #include <fstream>
00004 #include <iostream>
00005 #include <string>
00006
00007 #include "core/deque.hpp"
00008 #include "raygui.h"
00009 #include "utils.hpp"
00010
00011 namespace component {
00012
00013 FileDialog::FileDialog(int mode, const char* title,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

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

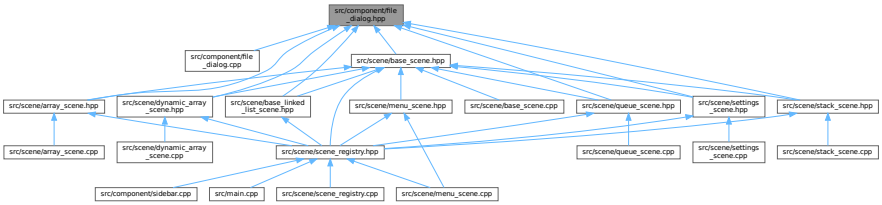
7.7 src/component/file_dialog.hpp File Reference

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

Include dependency graph for file_dialog.hpp:



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



Classes

- class component::FileDialog

  File Dialog for opening and saving files.

Namespaces

- namespace component

7.8 file_dialog.hpp

Go to the documentation of this file.

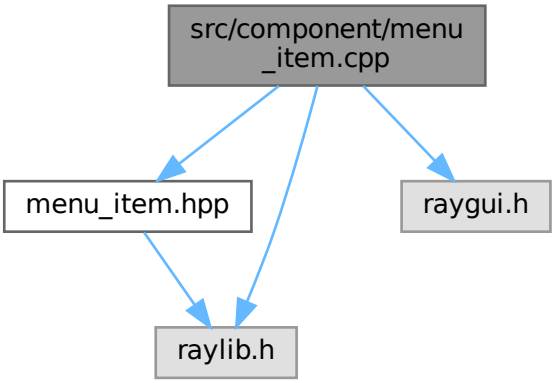
```
00001 #ifndef COMPONENT_FILE_DIALOG_HPP_
00002 #define COMPONENT_FILE_DIALOG_HPP_
00003
00004 #include <string>
00005
00006 #include "constants.hpp"
00007 #include "core/deque.hpp"
00008 #include "gui_file_dialog.h"
00009 #include "raylib.h"
00010
00011 namespace component {
00012
00017 class FileDialog {
00018 public:
00022     static constexpr Vector2 size{200, 50};
00023
00027     FileDialog();
00028
00035     FileDialog(int mode, const char* title, const char* message);
00036
00045     int render_head(float& options_head, float head_offset);
00046
00055     int render(float x, float y);
00056
00061     core::Deque<int> extract_values();
00062
00068     bool is_active() const;
00069
00074     std::string get_path();
00075
00079     void set_mode_open();
00080
00084     void set_mode_save();
00085
00090     void set_message(const char* message);
00091
00096     void set_title(const char* title);
00097
00098 private:
00102     GuiFileDialogState m_file_dialog_state{
00103         InitGuiFileDialog(GetWorkingDirectory())};
00104
00108     char m_file_input[constants::text_buffer_size] = "";  // NOLINT
```

```
00109
00113     int m_mode{};
00114
00118     const char* m_message;
00119
00123     const char* m_title;
00124 };
00125
00126 }  // namespace component
00127
00128 #endif  // COMPONENT_FILE_DIALOG_HPP_
```

7.9 src/component/menu_item.cpp File Reference

```
#include "menu_item.hpp"
#include "raygui.h"
#include "raylib.h"
```

Include dependency graph for menu_item.cpp:



Namespaces

- namespace component

7.10 menu_item.cpp

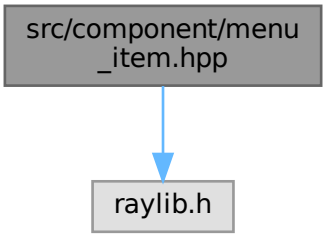
Go to the documentation of this file.

```
00001 #include "menu_item.hpp"
00002
00003 #include "raygui.h"
00004 #include "raylib.h"
00005
00006 namespace component {
00007
00008 MenuItem::MenuItem(const char* text, int x, int y, const char* img_path)
00009     : m_text{text},
00010       m_x{x},
00011       m_y{y},
00012       m_texture{LoadTextureFromImage(LoadImage(img_path))} {}
00013
```

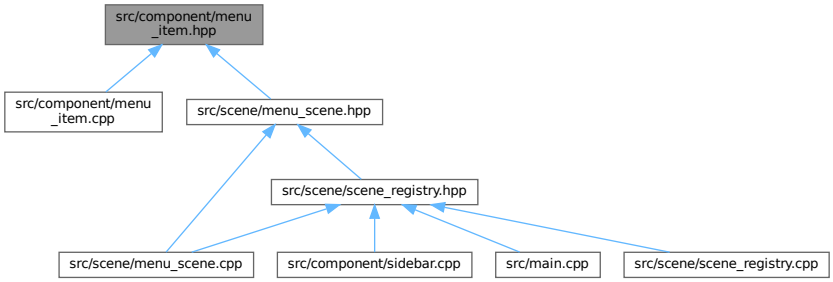
```
00014 int MenuItem::x() const { return m_x; }
00015 int MenuItem::y() const { return m_y; }
00016
00017 void MenuItem::render() {
00018     auto mouse = GetMousePosition();
00019     const Rectangle bound{(float)m_x, (float)m_y, block_width, block_height};
00020     const Rectangle text_bound{(float)m_x + 20,
00021                                (float)m_y + block_height - button_height,
00022                                button_width - 20, button_height};
00023
00024     DrawRectangleRec(bound, RAYWHITE);
00025     DrawTexture(m_texture, m_x, m_y, WHITE);
00026     GuiLabelButton(text_bound, m_text);
00027     DrawRectangleLinesEx(bound, 2, BLACK);
00028
00029     if (CheckCollisionPointRec(mouse, bound)) {
00030         DrawRectangle(m_x, m_y, block_width, block_height,
00031                       ColorAlpha(BLUE, 0.3));
00032         m_clicked = IsMouseButtonPressed(MOUSE_LEFT_BUTTON);
00033     }
00034 }
00035
00036 bool MenuItem::clicked() const { return m_clicked; }
00037
00038 void MenuItem::reset() { m_clicked = false; }
00039
00040 }  // namespace component
```

7.11 src/component/menu_item.hpp File Reference

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



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



Classes

- class component::MenuItem

  Items in the menu screen to navigate to other screens.

Namespaces

- namespace component

7.12 menu_item.hpp

Go to the documentation of this file.

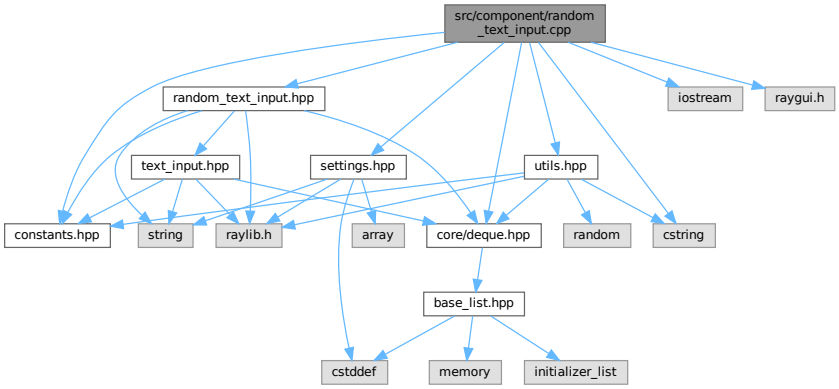
```
00001 #ifndef COMPONENT_MENU_ITEM_HPP_
00002 #define COMPONENT_MENU_ITEM_HPP_
00003
00004 #include "raylib.h"
00005
00006 namespace component {
00007
00012 class MenuItem {
00013 public:
00017     static constexpr int block_width = 300;
00018
00022     static constexpr int block_height = 200;
00023
00027     static constexpr int button_width = block_width;
00028
00032     static constexpr int button_height = 50;
00033
00037     MenuItem() = default;
00038
00046     MenuItem(const char* text, int x, int y, const char* img_path);
00047
00052     int x() const;
00053
00058     int y() const;
00059
00063     void render();
00064
00070     bool clicked() const;
00071
00075     void reset();
00076
00077 private:
00081     int m_x{};
00082
00086     int m_y{};
00087
00091     Texture2D m_texture{};
00092
00096     const char* m_text{};
00097
00101     bool m_clicked{};
00102 };
00103
00104 }  // namespace component
00105
00106 #endif  // COMPONENT_MENU_ITEM_HPP_
```

7.13 src/component/random_text_input.cpp File Reference

```
#include "random_text_input.hpp"
#include <cstring>
#include <iostream>
#include "constants.hpp"
#include "core/deque.hpp"
#include "raygui.h"
```

```
#include "settings.hpp"
#include "utils.hpp"
```

Include dependency graph for random_text_input.cpp:



Namespaces

- namespace component

7.14 random_text_input.cpp

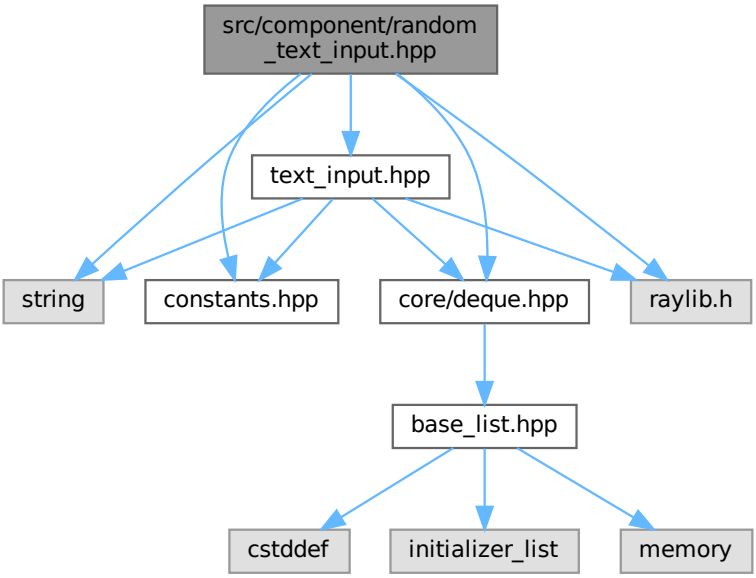
Go to the documentation of this file.

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

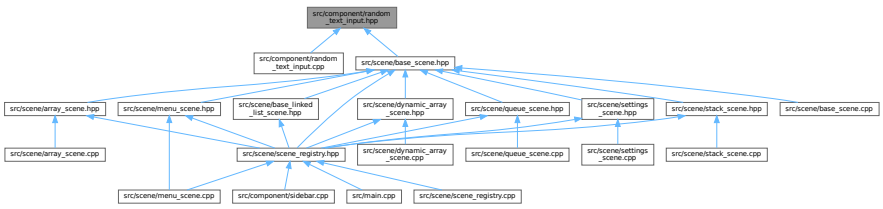
7.15 src/component/random_text_input.hpp File Reference

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

Include dependency graph for random_text_input.hpp:



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



Classes

- class component::RandomTextInput

  Text input that supports random values.

Namespaces

- namespace component

7.16 random_text_input.hpp

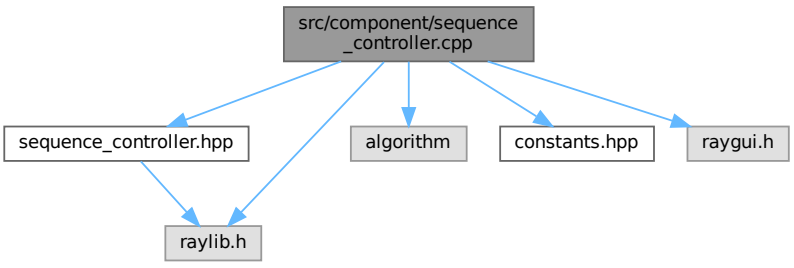
Go to the documentation of this file.

```
00001 #ifndef COMPONENT_RANDOM_TEXT_INPUT_HPP_
00002 #define COMPONENT_RANDOM_TEXT_INPUT_HPP_
00003
00004 #include <string>
00005
00006 #include "constants.hpp"
00007 #include "core/deque.hpp"
00008 #include "raylib.h"
00009 #include "text_input.hpp"
00010
00011 namespace component {
00012
00017 class RandomTextInput : public TextInput {
00018 public:
00019     using TextInput::size;
00020
00024     RandomTextInput() = default;
00025
00030     RandomTextInput(const char* label);
00031
00032     using TextInput::extract_values;
00033
00040     void render_head(float& options_head, float head_offset);
00041
00047     bool interact();
00048
00053     void set_random_min(int value);
00054
00059     void set_random_max(int value);
00060
00061 private:
00065     int m_random_min{constants::min_val};
00066
00070     int m_random_max{constants::max_val};
00071
00075     bool m_set_random{};
00076 };
00077
00078 }  // namespace component
00079
00080 #endif  // COMPONENT_RANDOM_TEXT_INPUT_HPP_
```

7.17 src/component/sequence_controller.cpp File Reference

```
#include "sequence_controller.hpp"
#include <algorithm>
#include "constants.hpp"
#include "raygui.h"
#include "raylib.h"
```

Include dependency graph for sequence_controller.cpp:



Namespaces

- namespace component

7.18 sequence_controller.cpp

Go to the documentation of this file.

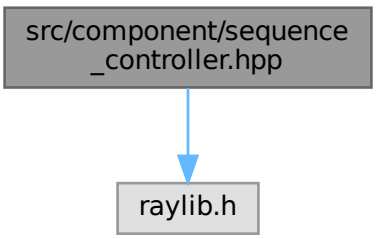
```
00001 #include "sequence_controller.hpp"
00002
00003 #include <algorithm>
00004
00005 #include "constants.hpp"
00006 #include "raygui.h"
00007 #include "raylib.h"
00008
00009 namespace component {
00010
00011 void SequenceController::set_max_value(int num) { m_num_steps = num; }
00012
00013 void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

```
00074
00075     Rectangle speed_shape{prev_speed_shape.x + 1.5F * button_size.x,
00076                           prev_speed_shape.y, 120, button_size.y};
00077
00078     m_prev_speed = GuiButton(prev_speed_shape, "#114#");
00079     m_next_speed = GuiButton(next_speed_shape, "#115#");
00080     GuiStatusBar(speed_shape, TextFormat("Speed: %.2fx", get_speed_scale()));
00081
00082     m_replay = GuiButton(replay_shape, "#75#");
00083     m_prev_frame = GuiButton(prev_frame_shape, "#72#");
00084     m_progress_value =
00085         (int)GuiProgressBar(progress_shape, nullptr, nullptr,
00086                             (float)m_progress_value, 0, (float)m_num_steps);
00087     m_next_frame = GuiButton(next_frame_shape, "#73#");
00088 }
00089
00090 bool SequenceController::interact() {
00091     if (m_replay) {
00092         set_progress_value(0);
00093         set_run_all(true);
00094         return true;
00095     }
00096
00097     if (m_prev_frame) {
00098         set_progress_value(std::max(get_progress_value() - 1, 0));
00099         return true;
00100     }
00101
00102     if (m_next_frame) {
00103         set_progress_value(std::min(get_progress_value() + 1, m_num_steps));
00104         return true;
00105     }
00106
00107     if (m_prev_speed) {
00108         m_speed = std::max(m_speed - 1, 2);
00109         return true;
00110     }
00111
00112     if (m_next_speed) {
00113         m_speed = std::min(m_speed + 1, 6);
00114         return true;
00115     }
00116
00117     return false;
00118 }
00119
00120 }  // namespace component
```

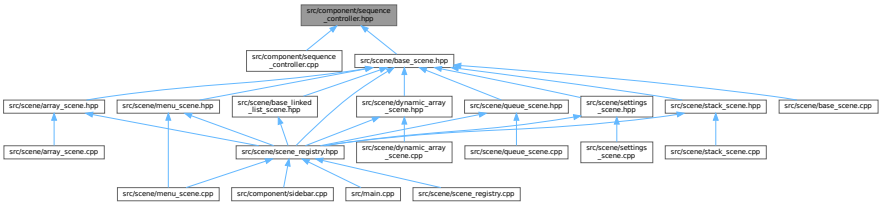
7.19 src/component/sequence_controller.hpp File Reference

`#include "raylib.h"`

Include dependency graph for sequence_controller.hpp:



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



Classes

- class component::SequenceController

  Controls the display of frames of the animation sequence.

Namespaces

- namespace component

7.20 sequence_controller.hpp

Go to the documentation of this file.

```
00001 #ifndef COMPONENT_SEQUENCE_CONTROLLER_HPP_
00002 #define COMPONENT_SEQUENCE_CONTROLLER_HPP_
00003
00004 #include "raylib.h"
00005
00006 namespace component {
00007
00012 class SequenceController {
00013 public:
00017     void render();
00018
00024     bool interact();
00025
00030     void set_max_value(int num);
00031
00036     void set_progress_value(int value);
00037
00043     void set_run_all(bool run_all);
00044
00048     void set_rerun();
00049
00055     bool get_run_all() const;
00056
00061     int get_progress_value() const;
00062
00067     float get_speed_scale() const;
00068
00072     void reset_anim_counter();
00073
00077     void inc_anim_counter();
00078
00083     int get_anim_counter() const;
00084
00089     int get_anim_frame() const;
00090
00091 private:
00095     static constexpr Vector2 button_size{25, 25};
00096
00100     static constexpr int speed_scale = 4;
00101
00105     bool m_replay{};
00106
00110     bool m_prev_frame{};
```

```
00111
00115     bool m_next_frame{};
00116
00120     int m_progress_value{};
00121
00125     int m_num_steps{};
00126
00130     bool m_run_all{};
00131
00135     int m_anim_counter{};
00136
00140     bool m_prev_speed{};
00141
00145     bool m_next_speed{};
00146
00150     int m_speed{speed_scale};
00151 };
00152
00153 }  // namespace component
00154
00155 #endif  // COMPONENT_SEQUENCE_CONTROLLER_HPP_
```

7.21 src/component/sidebar.cpp File Reference

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

Include dependency graph for sidebar.cpp:

Namespaces

- namespace component

7.22 sidebar.cpp

Go to the documentation of this file.

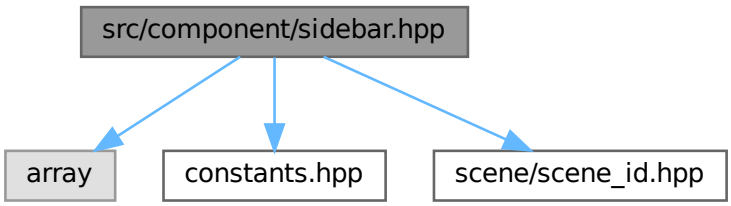
```
00001 #include "sidebar.hpp"
00002
00003 #include "constants.hpp"
00004 #include "raygui.h"
00005 #include "raylib.h"
00006 #include "scene/scene_id.hpp"
00007 #include "scene/scene_registry.hpp"
00008 #include "utils.hpp"
00009
00010 namespace component {
00011
00012 void SideBar::render() {
00013     (m_edit_mode) ? GuiLock() : GuiUnlock();
00014
00015     scene::SceneRegistry& registry = scene::SceneRegistry::get_instance();
00016     int options_head = 2 * constants::sidebar_width;
00017
00018     constexpr float scale = 0.2;
```

```
00019
00020     constexpr Rectangle menu_button_shape{20, 20, button_height * 2,
00021                                           button_height};
00022     constexpr Rectangle selection_shape{
00023         menu_button_shape.x + menu_button_shape.width + 10, menu_button_shape.y,
00024         button_width, button_height};
00025     constexpr Rectangle settings_button_shape{
00026         constants::scene_width - button_height - 20, 20, button_height,
00027         button_height};
00028
00029     m_next_scene = registry.get_scene();
00030
00031     bool menu_is_next = m_next_scene == scene::Menu;
00032     bool settings_is_next = m_next_scene == scene::Settings;
00033
00034     if (!menu_is_next) {
00035         m_return_menu = GuiButton(menu_button_shape, "#118#Menu");
00036     }
00037
00038     int next_scene = m_next_scene;
00039
00040     if (!menu_is_next && !settings_is_next) {
00041         if (GuiDropdownBox(selection_shape, sidebar_labels, &next_scene,
00042                            m_edit_mode)) {
00043             m_pressed = true;
00044             m_edit_mode ^= 1;
00045         }
00046     }
00047
00048     m_next_scene = scene::SceneId(next_scene);
00049
00050     m_return_settings = GuiButton(settings_button_shape, "#142#");
00051 }
00052
00053 void SideBar::interact() {
00054     scene::SceneRegistry& registry = scene::SceneRegistry::get_instance();
00055     bool menu_is_current = registry.get_scene() == scene::Menu;
00056     bool settings_is_current = registry.get_scene() == scene::Settings;
00057
00058     if (!menu_is_current) {
00059         if (m_return_menu) {
00060             registry.set_scene(scene::Menu);
00061             m_return_menu = false;
00062             return;
00063         }
00064     }
00065
00066     if (!menu_is_current && !settings_is_current) {
00067         if (m_pressed) {
00068             registry.set_scene(m_next_scene);
00069             m_pressed = false;
00070             return;
00071         }
00072     }
00073
00074     if (m_return_settings) {
00075         if (settings_is_current) {
00076             registry.set_scene(scene::SceneId(m_scene_before_settings));
00077         } else {
00078             m_scene_before_settings = registry.get_scene();
00079             registry.set_scene(scene::Settings);
00080         }
00081         m_return_settings = false;
00082         return;
00083     }
00084 }
00085
00086 }  // namespace component
```

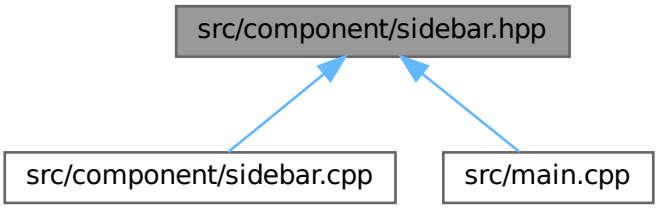
7.23 src/component/sidebar.hpp File Reference

```
#include <array>
#include "constants.hpp"
#include "scene/scene_id.hpp"
```

Include dependency graph for sidebar.hpp:



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



Classes

- class component::SideBar

  "Side bar" for extra navigation

Namespaces

- namespace component

7.24 sidebar.hpp

Go to the documentation of this file.

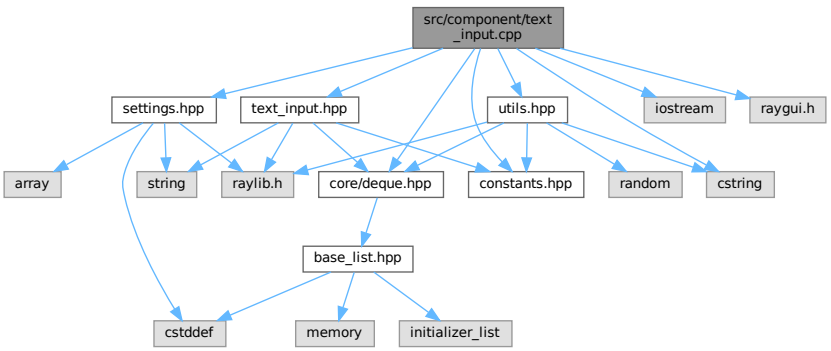
```
00001 #ifndef COMPONENT_SIDEBAR_HPP_
00002 #define COMPONENT_SIDEBAR_HPP_
00003
00004 #include <array>
00005
00006 #include "constants.hpp"
00007 #include "scene/scene_id.hpp"
00008
00009 namespace component {
00010
```

```
00016 class SideBar {
00017 public:
00021     void render();
00022
00026     void interact();
00027
00028 private:
00032     static constexpr int num_scenes = 8;
00033
00037     static constexpr int button_width = constants::sidebar_width;
00038
00042     static constexpr int button_height = 50;
00043
00047     static constexpr const char* sidebar_labels =
00048         "Array;"
00049         "Dynamic Array;"
00050         "Linked List;"
00051         "Doubly Linked List;"
00052         "Circular Linked List;"
00053         "Stack;"
00054         "Queue";
00055
00059     scene::SceneId m_next_scene{};
00060
00064     bool m_edit_mode{};
00065
00069     bool m_return_menu{};
00070
00074     bool m_return_settings{};
00075
00079     int m_scene_before_settings{};
00080
00084     bool m_pressed{};
00085 };
00086
00087 }  // namespace component
00088
00089 #endif  // COMPONENT_SIDEBAR_HPP_
```

7.25 src/component/text_input.cpp File Reference

```
#include "text_input.hpp"
#include <cstring>
#include <iostream>
#include "constants.hpp"
#include "core/deque.hpp"
#include "raygui.h"
#include "settings.hpp"
#include "utils.hpp"
```

Include dependency graph for text_input.cpp:



Namespaces

- namespace component

7.26 text_input.cpp

Go to the documentation of this file.

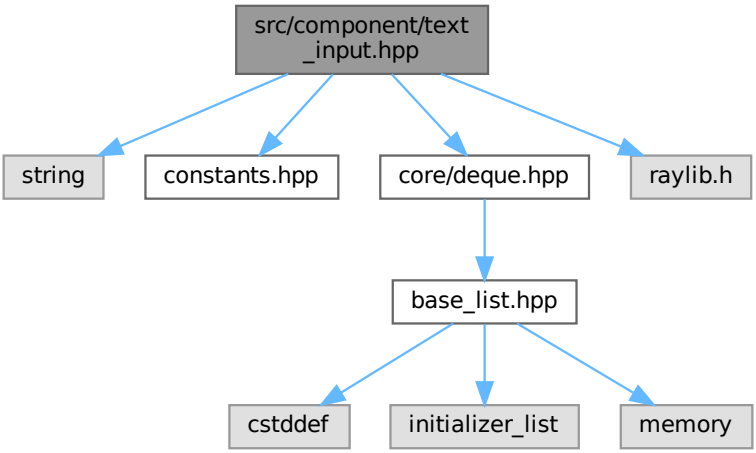
```
00001 #include "text_input.hpp"
00002
00003 #include <cstring>
00004 #include <iostream>
00005
00006 #include "constants.hpp"
00007 #include "core/deque.hpp"
00008 #include "raygui.h"
00009 #include "settings.hpp"
00010 #include "utils.hpp"
00011
00012 namespace component {
00013
00014 TextInput::TextInput(const char* label) : m_label{label} {}
00015
00016 void TextInput::render(float x, float y) {
00017     Rectangle shape{x, y, size.x, size.y};
00018
00019     utils::DrawText(
00020         m_label, {x, y - 25},
00021         utils::adaptive_text_color(
00022             Settings::get_instance().get_color(Settings::num_color - 1)),
00023         20, 2);
00024
00025     DrawRectangleRec(shape, RAYWHITE);
00026
00027     if (GuiTextBox(shape, static_cast<char*>(m_text_input),
00028                    constants::text_buffer_size, m_is_active)) {
00029         m_is_active ^= 1;
00030     }
00031 }
00032
00033 void TextInput::render_head(float& options_head, float head_offset) {
00034     render(options_head, constants::scene_height - size.y);
00035     options_head += (size.x + head_offset);
00036 }
00037
00038 std::string TextInput::get_input() const { return {m_text_input}; }
00039
00040 bool TextInput::is_active() const { return m_is_active; }
00041
00042 void TextInput::set_label(const char* const label) { m_label = label; }
00043
00044 void TextInput::set_input(const char* input, int len) {
00045     std::strncpy(static_cast<char*>(m_text_input), input, len);
00046 }
00047
00048 core::Deque<int> TextInput::extract_values() {
00049     core::Deque<int> nums = utils::str_extract_data(m_text_input);  // NOLINT
00050     return nums;
00051 }
00052
00053 }  // namespace component
```

7.27 src/component/text_input.hpp File Reference

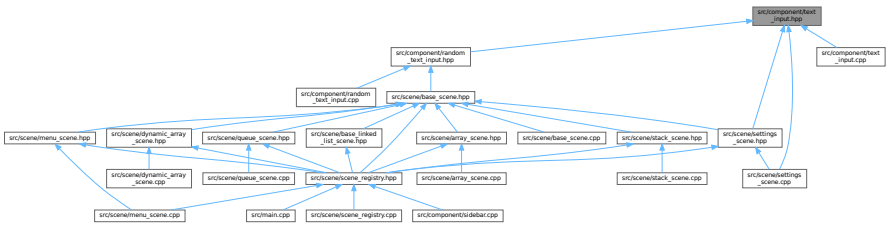
```
#include <string>
#include "constants.hpp"
#include "core/deque.hpp"
```

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

Include dependency graph for text_input.hpp:



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



Classes

- class component::TextInput

  Input for entering text.

Namespaces

- namespace component

7.28 text_input.hpp

Go to the documentation of this file.

```
00001 #ifndef COMPONENT_TEXT_INPUT_HPP_
00002 #define COMPONENT_TEXT_INPUT_HPP_
00003
00004 #include <string>
```

```
00005
00006 #include "constants.hpp"
00007 #include "core/deque.hpp"
00008 #include "raylib.h"
00009
00010 namespace component {
00011
00016 class TextInput {
00017 public:
00021     static constexpr Vector2 size{200, 50};
00022
00026     TextInput() = default;
00027
00032     TextInput(const char* label);
00033
00039     void render(float x, float y);
00040
00046     void render_head(float& options_head, float head_offset);
00047
00052     std::string get_input() const;
00053
00059     bool is_active() const;
00060
00066     void set_input(const char* input, int len);
00067
00072     void set_label(const char* const label);
00073
00078     core::Deque<int> extract_values();
00079
00080 protected:
00084     char m_text_input[constants::text_buffer_size] = "";  // NOLINT
00085
00089     bool m_is_active{};
00090
00094     const char* m_label{};
00095 };
00096
00097 }  // namespace component
00098
00099 #endif  // COMPONENT_TEXT_INPUT_HPP_
```

7.29 src/constants.hpp File Reference

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

Namespaces

- namespace constants

Variables

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

7.30 constants.hpp

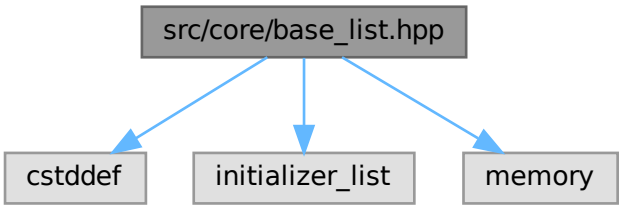
Go to the documentation of this file.

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

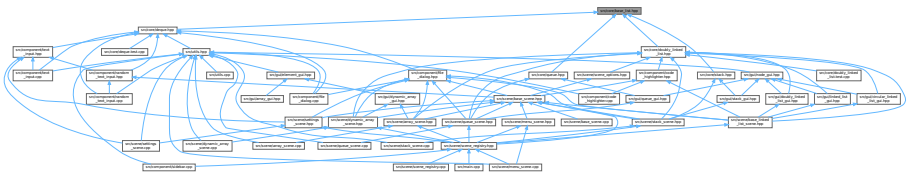
7.31 src/core/base_list.hpp File Reference

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

Include dependency graph for base_list.hpp:



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



Classes

- class core::BaseList< T >

  The base container for implementing other data structures.
- struct core::BaseList< T >::Node

  The node of the list.

Namespaces

- namespace core

7.32 base_list.hpp

Go to the documentation of this file.

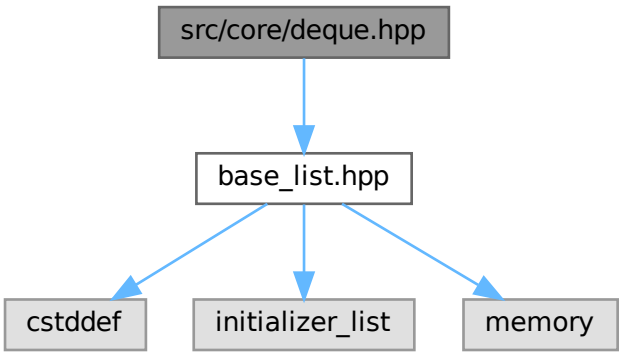
```
00001 #ifndef CORE_BASE_LIST_HPP_
00002 #define CORE_BASE_LIST_HPP_
00003
00004 #include <cstddef>
00005 #include <initializer_list>
00006 #include <memory>
00007
00008 namespace core {
00009
00015 template<typename T>
00016 class BaseList {
00017 public:
00021     BaseList() = default;
00022
00027     BaseList(std::initializer_list<T> init_list);
00028
00033     BaseList(const BaseList& rhs);
00034
00039     BaseList& operator=(const BaseList& rhs);
00040
00045     BaseList(BaseList&& rhs) noexcept;
00046
00051     BaseList& operator=(BaseList&& rhs) noexcept;
00052
00056     ~BaseList();
00057
00063     [[nodiscard]] bool empty() const;
00064
00069     [[nodiscard]] std::size_t size() const;
00070
00071 protected:
00072     struct Node;  // Forward declaration
00073     using Node_ptr = Node*;
00074
00078     struct Node {
00079         T data{};
00080         Node_ptr prev{};
00081         Node_ptr next{};
00082     };
00083
00087     Node_ptr m_head{nullptr};
00088
00092     Node_ptr m_tail{nullptr};
00093
00097     std::size_t m_size{};
00098
00103     void init_first_element(const T& elem);
00104
00109     void clean_up();
00110
00116     void copy_data(const BaseList& rhs);
00117
00122     void push_back(const T& elem);
00123
00128     void push_front(const T& elem);
00129
00134     T& back() const;
00135
00140     T& front() const;
00141
```

```
00145     void pop_front();
00146
00150     void pop_back();
00151 };
00152
00153 template<typename T>
00154 BaseList<T>::BaseList(const BaseList& rhs) {
00155     copy_data(rhs);
00156 }
00157
00158 template<typename T>
00159 BaseList<T>::BaseList(std::initializer_list<T> init_list) {
00160     for (const auto& elem : init_list) {
00161         push_back(elem);
00162     }
00163 }
00164
00165 template<typename T>
00166 BaseList<T>& BaseList<T>::operator=(const BaseList& rhs) {
00167     if (this != &rhs) {
00168         copy_data(rhs);
00169     }
00170
00171     return *this;
00172 }
00173
00174 template<typename T>
00175 BaseList<T>::BaseList(BaseList&& rhs) noexcept
00176     : m_head{rhs.m_head}, m_tail{rhs.m_tail}, m_size{rhs.m_size} {
00177     rhs.m_head = nullptr;
00178     rhs.m_tail = nullptr;
00179     rhs.m_size = 0;
00180 }
00181
00182 template<typename T>
00183 BaseList<T>& BaseList<T>::operator=(BaseList&& rhs) noexcept {
00184     if (this != &rhs) {
00185         clean_up();
00186
00187         m_head = rhs.m_head;
00188         m_tail = rhs.m_tail;
00189         m_size = rhs.m_size;
00190
00191         rhs.m_head = nullptr;
00192         rhs.m_tail = nullptr;
00193         rhs.m_size = 0;
00194     }
00195
00196     return *this;
00197 }
00198
00199 template<typename T>
00200 BaseList<T>::~BaseList() {
00201     clean_up();
00202 }
00203
00204 template<typename T>
00205 bool BaseList<T>::empty() const {
00206     return m_size == 0;
00207 }
00208
00209 template<typename T>
00210 std::size_t BaseList<T>::size() const {
00211     return m_size;
00212 }
00213
00214 template<typename T>
00215 void BaseList<T>::init_first_element(const T& elem) {
00216     m_head = new Node{elem, nullptr, nullptr};
00217     m_tail = m_head;
00218     m_size = 1;
00219 }
00220
00221 template<typename T>
00222 void BaseList<T>::clean_up() {
00223     Node_ptr ptr{nullptr};
00224
00225     while (m_head != nullptr) {
00226         ptr = m_head->next;
00227         delete m_head;
00228         m_head = ptr;
00229     }
00230
00231     m_tail = m_head;
00232     m_size = 0;
00233 }
00234
```

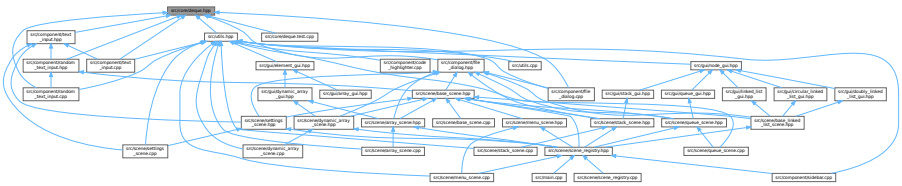
```
00235 template<typename T>
00236 void BaseList<T>::copy_data(const BaseList& rhs) {
00237     for (Node_ptr ptr = rhs.m_head; ptr != nullptr; ptr = ptr->next) {
00238         push_back(ptr->data);
00239     }
00240 }
00241
00242 template<typename T>
00243 void BaseList<T>::push_back(const T& elem) {
00244     if (empty()) {
00245         init_first_element(elem);
00246         return;
00247     }
00248
00249     m_tail->next = new Node{elem, m_tail, nullptr};
00250     m_tail = m_tail->next;
00251     ++m_size;
00252 }
00253
00254 template<typename T>
00255 void BaseList<T>::push_front(const T& elem) {
00256     if (empty()) {
00257         init_first_element(elem);
00258         return;
00259     }
00260
00261     m_head->prev = new Node{elem, nullptr, m_head};
00262     m_head = m_head->prev;
00263     ++m_size;
00264 }
00265
00266 template<typename T>
00267 T& BaseList<T>::back() const {
00268     return m_tail->data;
00269 }
00270
00271 template<typename T>
00272 T& BaseList<T>::front() const {
00273     return m_head->data;
00274 }
00275
00276 template<typename T>
00277 void BaseList<T>::pop_back() {
00278     if (size() <= 1) {
00279         clean_up();
00280         return;
00281     }
00282
00283     m_tail = m_tail->prev;
00284     delete m_tail->next;
00285     m_tail->next = nullptr;
00286     --m_size;
00287 }
00288
00289 template<typename T>
00290 void BaseList<T>::pop_front() {
00291     if (size() <= 1) {
00292         clean_up();
00293         return;
00294     }
00295
00296     m_head = m_head->next;
00297     delete m_head->prev;
00298     m_head->prev = nullptr;
00299     --m_size;
00300 }
00301
00302 }  // namespace core
00303
00304 #endif  // CORE_BASE_LIST_HPP_
```

7.33 src/core/deque.hpp File Reference

`#include "base_list.hpp"`
Include dependency graph for deque.hpp:



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



Classes

- class core::Deque< T >
  The deque container.

Namespaces

- namespace core

7.34 deque.hpp

Go to the documentation of this file.

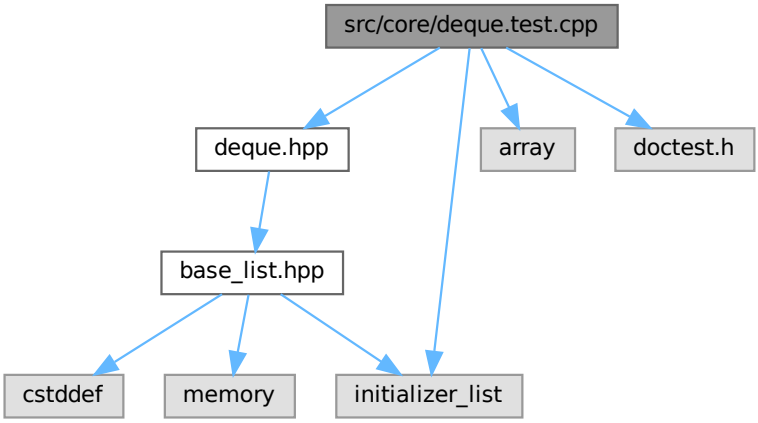
```
00001 #ifndef CORE_DEQUE_HPP_
00002 #define CORE_DEQUE_HPP_
00003
00004 #include "base_list.hpp"
00005
00006 namespace core {
```

```
00007
00013 template<typename T>
00014 class Deque : public BaseList<T> {
00015 private:
00016     using Base = BaseList<T>;
00017
00018 public:
00019     using Base::Base;
00020
00021     using Base::empty;
00022     using Base::size;
00023
00024     using Base::push_back;
00025     using Base::push_front;
00026
00027     using Base::back;
00028     using Base::front;
00029
00030     using Base::pop_back;
00031     using Base::pop_front;
00032 };
00033
00034 }  // namespace core
00035
00036 #endif  // CORE_DEQUE_HPP_
```

7.35 src/core/deque.test.cpp File Reference

```
#include "deque.hpp"
#include <array>
#include <initializer_list>
#include "doctest.h"
```

Include dependency graph for deque.test.cpp:



Functions

- TEST_CASE ("core::Deque functionality")
- __attribute__ ((always_inline)) void check_match(core
- TEST_CASE ("core::Deque special member functions")

Variables

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

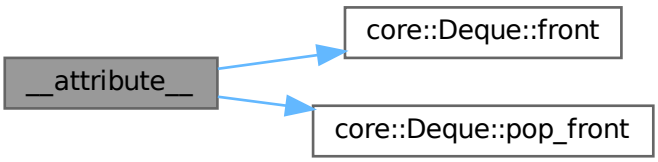
7.35.1 Function Documentation

7.35.1.1 __attribute__()

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

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

Here is the call graph for this function:

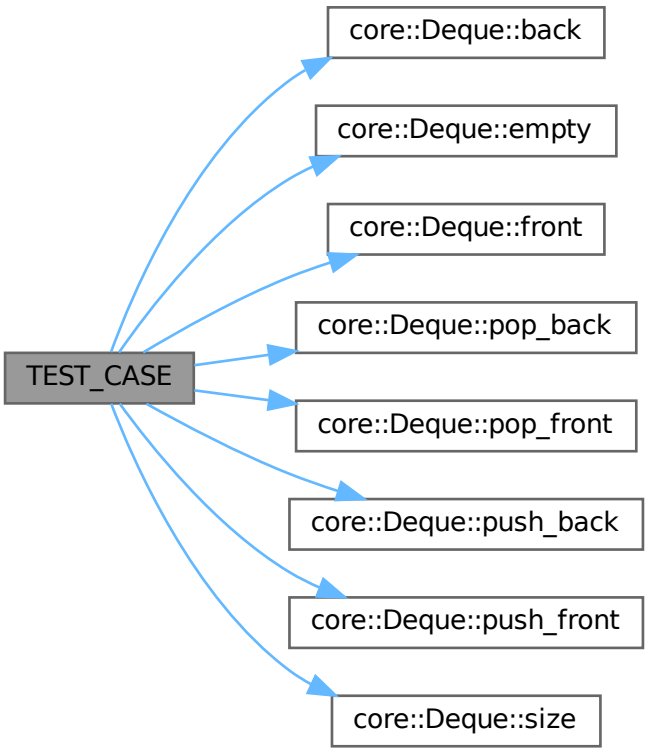


7.35.1.2 TEST_CASE() [1/2]

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

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

Here is the call graph for this function:



7.35.1.3 TEST_CASE() [2/2]

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

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

7.35.2 Variable Documentation

7.35.2.1 list

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

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

7.36 deque.test.cpp

Go to the documentation of this file.

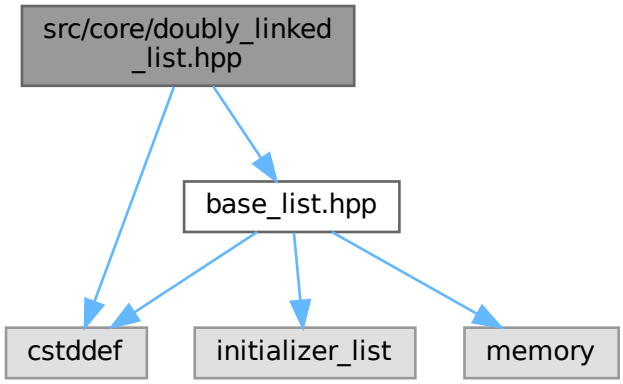
```
00001 #include "deque.hpp"
00002
00003 #include <array>
00004 #include <initializer_list>
00005
00006 #include "doctest.h"
00007
00008 TEST_CASE("core::Deque functionality") {
00009     core::Deque<int> deque;
00010     CHECK(deque.empty());
00011
00012     deque.push_back(2);
00013     deque.push_back(3);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

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

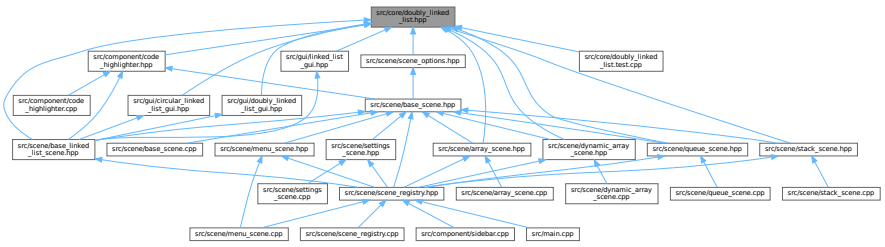
7.37 src/core/doubly_linked_list.hpp File Reference

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

Include dependency graph for doubly_linked_list.hpp:



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



Classes

- class core::DoublyLinkedList< T >

  The doubly linked list container.

Namespaces

- namespace core

7.38 doubly_linked_list.hpp

Go to the documentation of this file.

```
00001 #ifndef CORE_DOUBLY_LINKED_LIST_HPP_
00002 #define CORE_DOUBLY_LINKED_LIST_HPP_
00003
00004 #include <cstddef>
00005
00006 #include "base_list.hpp"
00007
00008 namespace core {
00009
00015 template<typename T>
00016 class DoublyLinkedList : public BaseList<T> {
00017 protected:
00018     using Base = BaseList<T>;
00019     using Node = typename Base::Node;
00020     using Node_ptr = Node*;
00021     using cNode_ptr = const Node*;
00022
00023 public:
00024     using Base::Base;
00025
00026     using Base::empty;
00027     using Base::size;
00028
00034     Node_ptr search(const T& elem);
00035
00041     Node_ptr find(std::size_t index);
00042
00048     cNode_ptr search(const T& elem) const;
00049
00055     cNode_ptr find(std::size_t index) const;
00056
00063     Node_ptr insert(std::size_t index, const T& elem);
00064
00070     Node_ptr remove(std::size_t index);
00071
00077     T& at(std::size_t index);
00078
00084     T at(std::size_t index) const;
00085
00089     void clear();
00090
00091 protected:
00092     using Base::m_head;
00093     using Base::m_size;
00094     using Base::m_tail;
00095
00101     Node_ptr internal_search(const T& elem);
00102
00108     Node_ptr internal_find(std::size_t index);
00109 };
00110
00111 template<typename T>
00112 typename DoublyLinkedList<T>::Node_ptr DoublyLinkedList<T>::internal_search(
00113     const T& elem) {
00114     Node_ptr ptr{m_head};
00115
00116     while (ptr != nullptr) {
00117         if (ptr->data == elem) {
00118             break;
00119         }
00120
00121         ptr = ptr->next;
00122     }
```

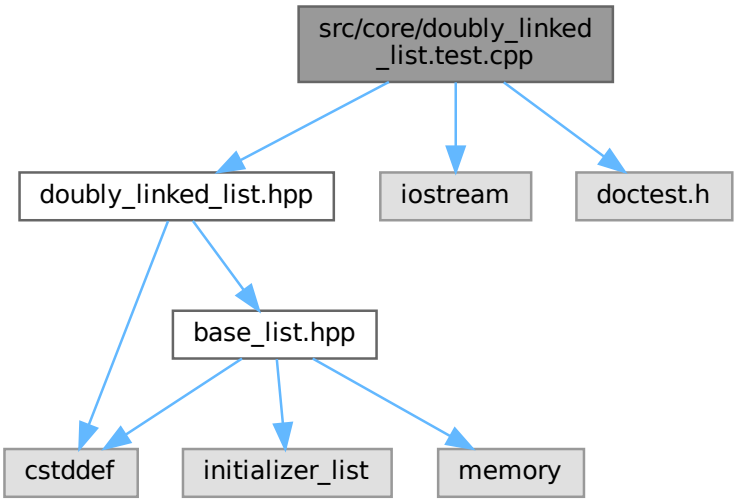
```
00123
00124     return ptr;
00125 }
00126
00127 template<typename T>
00128 typename DoublyLinkedList<T>::Node_ptr DoublyLinkedList<T>::internal_find(
00129     std::size_t index) {
00130     Node_ptr ptr{m_head};
00131     std::size_t pos = 0;
00132
00133     while (ptr != nullptr && pos < index) {
00134         ptr = ptr->next;
00135         ++pos;
00136     }
00137
00138     return ptr;
00139 }
00140
00141 template<typename T>
00142 typename DoublyLinkedList<T>::Node_ptr DoublyLinkedList<T>::search(
00143     const T& elem) {
00144     return internal_search(elem);
00145 }
00146
00147 template<typename T>
00148 typename DoublyLinkedList<T>::Node_ptr DoublyLinkedList<T>::find(
00149     std::size_t index) {
00150     return internal_find(index);
00151 }
00152
00153 template<typename T>
00154 typename DoublyLinkedList<T>::cNode_ptr DoublyLinkedList<T>::search(
00155     const T& elem) const {
00156     return internal_search(elem);
00157 }
00158
00159 template<typename T>
00160 typename DoublyLinkedList<T>::cNode_ptr DoublyLinkedList<T>::find(
00161     std::size_t index) const {
00162     return internal_find(index);
00163 }
00164
00165 template<typename T>
00166 typename DoublyLinkedList<T>::Node_ptr DoublyLinkedList<T>::insert(
00167     std::size_t index, const T& elem) {
00168     if (index == 0) {
00169         Base::push_front(elem);
00170         return m_head;
00171     }
00172
00173     if (index >= m_size) {
00174         Base::push_back(elem);
00175         return m_tail;
00176     }
00177
00178     Node_ptr ptr = find(index);
00179     auto new_node = new Node{elem, ptr->prev, ptr};
00180
00181     ptr->prev->next = new_node;
00182     ptr->prev = new_node;
00183     ++m_size;
00184
00185     return new_node;
00186 }
00187
00188 template<typename T>
00189 typename DoublyLinkedList<T>::Node_ptr DoublyLinkedList<T>::remove(
00190     std::size_t index) {
00191     if (index >= m_size) {
00192         return nullptr;
00193     }
00194
00195     if (index == 0) {
00196         Base::pop_front();
00197         return m_head;
00198     }
00199
00200     if (index + 1 == m_size) {
00201         Base::pop_back();
00202         return nullptr;
00203     }
00204
00205     Node_ptr ptr = find(index);
00206     Node_ptr ret = ptr->next;
00207
00208     ptr->next->prev = ptr->prev;
00209     ptr->prev->next = ptr->next;
```

```
00210
00211     delete ptr;
00212     --m_size;
00213
00214     return ret;
00215 }
00216
00217 template<typename T>
00218 T& DoublyLinkedList<T>::at(std::size_t index) {
00219     return find(index)->data;
00220 }
00221
00222 template<typename T>
00223 T DoublyLinkedList<T>::at(std::size_t index) const {
00224     return find(index)->data;
00225 }
00226
00227 template<typename T>
00228 void DoublyLinkedList<T>::clear() {
00229     while (!empty()) {
00230         Base::pop_front();
00231     }
00232 }
00233
00234 }  // namespace core
00235
00236 #endif  // CORE_DOUBLY_LINKED_LIST_HPP_
```

7.39 src/core/doubly_linked_list.test.cpp File Reference

```
#include "doubly_linked_list.hpp"
#include <iostream>
#include "doctest.h"
```

Include dependency graph for doubly_linked_list.test.cpp:



Functions

- TEST_CASE ("core::DoublyLinkedList functionality")

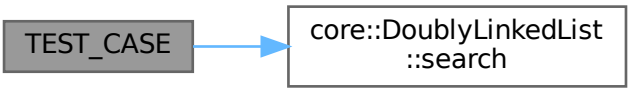
7.39.1 Function Documentation

7.39.1.1 TEST_CASE()

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

Definition at line 7 of file doubly_linked_list.test.cpp.

Here is the call graph for this function:



7.40 doubly_linked_list.test.cpp

Go to the documentation of this file.

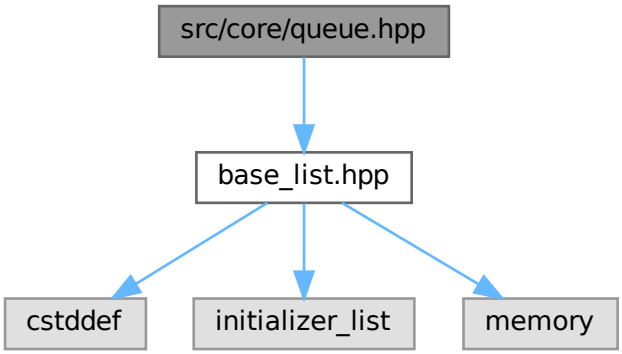
```
00001 #include "doubly_linked_list.hpp"
00002
00003 #include <iostream>
00004
00005 #include "doctest.h"
00006
00007 TEST_CASE("core::DoublyLinkedList functionality") {
00008     // List: {1, 2, 3}
00009     SUBCASE("Node_ptr search(const T& elem)") {
00010         core::DoublyLinkedList<int> dll{1, 2, 3};
00011         CHECK(dll.search(4) == nullptr);
00012         CHECK(dll.search(3)->data == 3);
00013     }
00014
00015     // List: {1, 2, 3}
00016     SUBCASE("Node_ptr find(std::size_t index)") {
00017         core::DoublyLinkedList<int> dll{1, 2, 3};
00018         CHECK(dll.find(8) == nullptr);
00019
00020         auto* ptr1 = dll.search(3);
00021         auto* ptr2 = dll.find(1);
00022
00023         CHECK(ptr1->data == 3);
00024         CHECK(ptr2->data == 2);
00025
00026         CHECK(ptr1->prev == ptr2);
00027         CHECK(ptr2->next == ptr1);
00028     }
00029
00030     SUBCASE("Node_ptr insert(std::size_t index, const T& elem)") {
00031         core::DoublyLinkedList<int> dll{1, 2, 3};
00032         auto* ptr0 = dll.search(1);
00033
00034         // List: {-1, 1, 2, 3}
00035         auto* ptr = dll.insert(0, -1);
00036
00037         CHECK(dll.size() == 4);
00038         CHECK(ptr->next == ptr0);
00039
```

```
00040         auto* ptrN = dll.search(3);
00041         // List: {-1, 1, 2, 3, 4}
00042         ptr = dll.insert(4, 4);
00043
00044         CHECK(dll.size() == 5);
00045         CHECK(ptr->prev == ptrN);
00046
00047         // List: {-1, 1, 20, 2, 3, 4}
00048         ptr = dll.insert(2, 20);  // NOLINT
00049         CHECK(ptr->prev == dll.find(1));
00050         CHECK(ptr->next == dll.find(3));
00051         CHECK(dll.size() == 6);
00052
00053         // List: {-1, 1, 20, 2, 3, 4, 69}
00054         dll.insert(69, 69);  // NOLINT
00055         CHECK(dll.search(69) == dll.find(6));
00056         CHECK(dll.size() == 7);
00057     }
00058
00059     // List: {-1, 1, 20, 2, 3, 4, 69}
00060     SUBCASE("Node_ptr remove(std::size_t index)") {
00061         core::DoublyLinkedList<int> dll{-1, 1, 20, 2, 3, 4, 69};  // NOLINT
00062
00063         CHECK(dll.remove(1000) == nullptr);
00064         CHECK(dll.size() == 7);
00065
00066         // List: {-1, 1, 20, 2, 3, 4}
00067         CHECK(dll.remove(6) == nullptr);
00068         CHECK(dll.size() == 6);
00069
00070         // List: {1, 20, 2, 3, 4}
00071         auto* ptr = dll.remove(0);
00072         CHECK(dll.size() == 5);
00073         CHECK(ptr->data == 1);
00074
00075         // List: {1, 2, 3, 4}
00076         ptr = dll.remove(1);
00077         CHECK(dll.size() == 4);
00078         CHECK(ptr->data == 2);
00079     }
00080 }
```

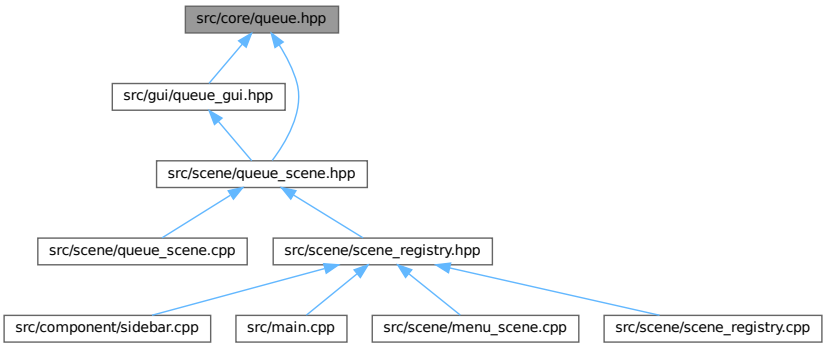
7.41 src/core/queue.hpp File Reference

`#include "base_list.hpp"`

Include dependency graph for queue.hpp:



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



Classes

- class core::Queue< T >

  The queue container.

Namespaces

- namespace core

7.42 queue.hpp

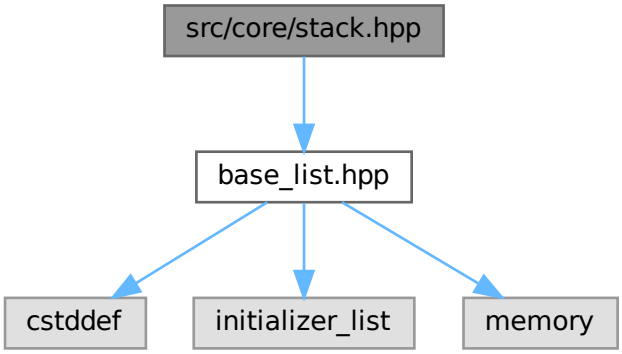
Go to the documentation of this file.

```
00001 #ifndef CORE_QUEUE_HPP_
00002 #define CORE_QUEUE_HPP_
00003
00004 #include "base_list.hpp"
00005
00006 namespace core {
00007
00013 template<typename T>
00014 class Queue : public BaseList<T> {
00015 private:
00016     using Base = BaseList<T>;
00017
00018 public:
00019     using Base::Base;
00020
00021     using Base::empty;
00022     using Base::size;
00023
00027     using Base::pop_back;
00028     using Base::push_front;
00029
00034     T& front() const;
00035
00040     T& back() const;
00041
00046     void push(const T& elem);
00047
00051     void pop();
00052 };
00053
00054 template<typename T>
00055 T& Queue<T>::front() const {
00056     return Base::front();
00057 }
```

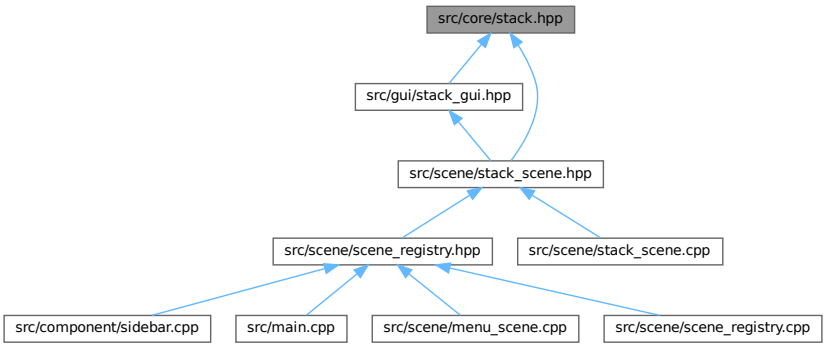
```
00058
00059 template<typename T>
00060 T& Queue<T>::back() const {
00061     return Base::back();
00062 }
00063
00064 template<typename T>
00065 void Queue<T>::push(const T& elem) {
00066     Base::push_back(elem);
00067 }
00068
00069 template<typename T>
00070 void Queue<T>::pop() {
00071     Base::pop_front();
00072 }
00073
00074 }  // namespace core
00075
00076 #endif  // CORE_QUEUE_HPP_
```

7.43 src/core/stack.hpp File Reference

`#include "base_list.hpp"`
Include dependency graph for stack.hpp:



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



Classes

- class core::Stack< T >

  The stack container.

Namespaces

- namespace core

7.44 stack.hpp

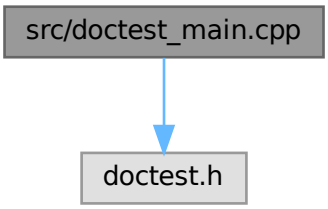
Go to the documentation of this file.

```
00001 #ifndef CORE_STACK_HPP_
00002 #define CORE_STACK_HPP_
00003
00004 #include "base_list.hpp"
00005
00006 namespace core {
00007
00013 template<typename T>
00014 class Stack : public BaseList<T> {
00015 private:
00016     using Base = BaseList<T>;
00017
00018 public:
00019     using Base::Base;
00020
00021     using Base::empty;
00022     using Base::size;
00023
00028     T& top() const;
00029
00034     void push(const T& elem);
00035
00039     void pop();
00040
00041 protected:
00042     using Base::m_head;
00043     using Base::m_tail;
00044 };
00045
00046 template<typename T>
00047 T& Stack<T>::top() const {
00048     return Base::front();
00049 }
00050
00051 template<typename T>
00052 void Stack<T>::push(const T& elem) {
00053     Base::push_front(elem);
00054 }
00055
00056 template<typename T>
00057 void Stack<T>::pop() {
00058     Base::pop_front();
00059 }
00060
00061 }  // namespace core
00062
00063 #endif  // CORE_STACK_HPP_
```

7.45 src/doctest_main.cpp File Reference

```
#include "doctest.h"
```

Include dependency graph for doctest_main.cpp:



Macros

- #define DOCTEST_CONFIG_IMPLEMENT_WITH_MAIN

7.45.1 Macro Definition Documentation

7.45.1.1 DOCTEST_CONFIG_IMPLEMENT_WITH_MAIN

```
#define DOCTEST_CONFIG_IMPLEMENT_WITH_MAIN
```

Definition at line 1 of file doctest_main.cpp.

7.46 doctest_main.cpp

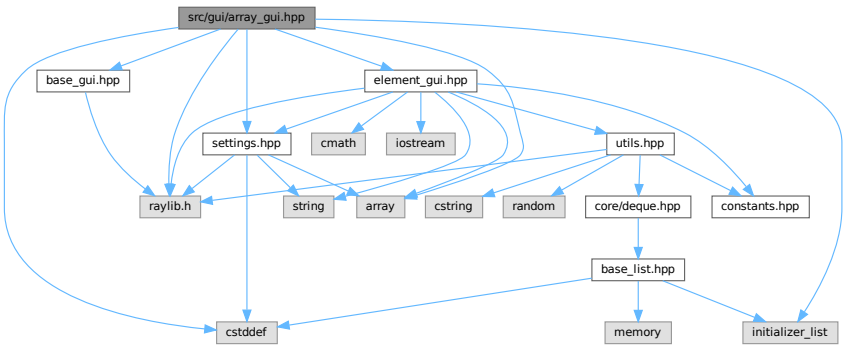
Go to the documentation of this file.

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

7.47 src/gui/array_gui.hpp File Reference

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

Include dependency graph for array_gui.hpp:



Classes

- class gui::GuiArray< T, N >

  The GUI array container.

Namespaces

- namespace gui

7.48 array_gui.hpp

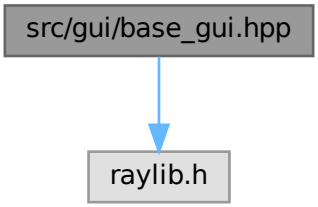
Go to the documentation of this file.

```
00001 #ifndef GUI_ARRAY_GUI_HPP_
00002 #define GUI_ARRAY_GUI_HPP_
00003
00004 #include <array>
00005 #include <cstddef>
00006 #include <initializer_list>
00007
00008 #include "base_gui.hpp"
00009 #include "element_gui.hpp"
00010 #include "raylib.h"
00011 #include "settings.hpp"
00012
00013 namespace gui {
00014
00021 template<typename T, std::size_t N>
00022 class GuiArray : public internal::Base {
00023 public:
00027     GuiArray();
00028
00033     GuiArray(std::array<GuiElement<T>, N>&& init_list);
```

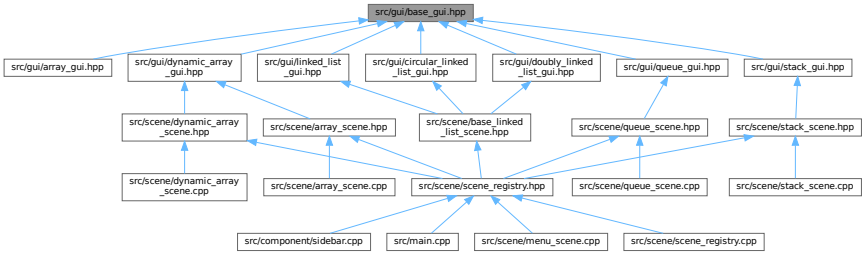
```
00034
00038     void update() override;
00039
00043     void render() override;
00044
00050     T& operator[](std::size_t idx);
00051
00057     T operator[](std::size_t idx) const;
00058
00065     void set_color_index(std::size_t idx, int color_index);
00066
00067 private:
00071     static constexpr Vector2 head_pos{
00072         constants::scene_width / 2.0F - 15 * GuiElement<T>::side,
00073         constants::scene_height / 2.0F};
00074
00078     std::array<GuiElement<T>, N> m_array{};
00079
00085     void render_link(Vector2 src, Vector2 dest) override;
00086 };
00087
00088 template<typename T, std::size_t N>
00089 GuiArray<T, N>::GuiArray() {
00090     for (std::size_t i = 0; i < N; ++i) {
00091         m_array[i] = GuiElement<T>{0, i};
00092         m_array[i].set_color_index(0);
00093     }
00094 }
00095
00096 template<typename T, std::size_t N>
00097 GuiArray<T, N>::GuiArray(std::array<GuiElement<T>, N>&& init_list)
00098     : m_array{init_list} {}
00099
00100 template<typename T, std::size_t N>
00101 void GuiArray<T, N>::render_link(Vector2 src, Vector2 dest) {}
00102
00103 template<typename T, std::size_t N>
00104 void GuiArray<T, N>::render() {
00105     update();
00106
00107     for (std::size_t i = 0; i < N; ++i) {
00108         m_array[i].render();
00109     }
00110 }
00111
00112 template<typename T, std::size_t N>
00113 void GuiArray<T, N>::update() {
00114     // TODO: if not outdated then return
00115
00116     for (std::size_t i = 0; i < N; ++i) {
00117         m_array[i].set_pos(
00118             {head_pos.x + 4 * GuiElement<T>::side * i, head_pos.y});
00119     }
00120 }
00121
00122 template<typename T, std::size_t N>
00123 T& GuiArray<T, N>::operator[](std::size_t idx) {
00124     return m_array[idx].get_value();
00125 }
00126
00127 template<typename T, std::size_t N>
00128 T GuiArray<T, N>::operator[](std::size_t idx) const {
00129     return m_array[idx].get_value();
00130 }
00131
00132 template<typename T, std::size_t N>
00133 void GuiArray<T, N>::set_color_index(std::size_t idx, int color_index) {
00134     m_array[idx].set_color_index(color_index);
00135 }
00136
00137 }  // namespace gui
00138
00139 #endif  // GUI_ARRAY_GUI_HPP_
```

7.49 src/gui/base_gui.hpp File Reference

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



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



Classes

- class gui::internal::Base

  The base class for all GUI classes.

Namespaces

- namespace gui
- namespace gui::internal

7.50 base_gui.hpp

Go to the documentation of this file.

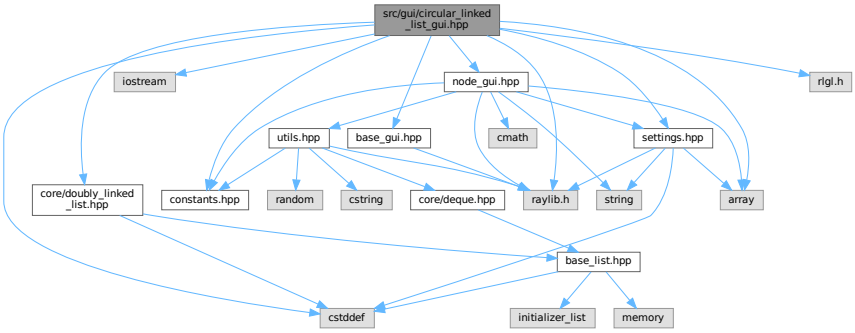
```
00001 #ifndef GUI_BASE_GUI_HPP_
00002 #define GUI_BASE_GUI_HPP_
00003
00004 #include "raylib.h"
00005
00006 namespace gui::internal {
```

```
00007
00012 class Base {
00013 public:
00017     Base() = default;
00018
00022     Base(const Base&) = default;
00023
00027     Base(Base&&) = default;
00028
00032     Base& operator=(const Base&) = default;
00033
00037     Base& operator=(Base&&) = default;
00038
00042     virtual ~Base() = default;
00043
00047     virtual void update() = 0;
00048
00052     virtual void render() = 0;
00053
00054 private:
00060     virtual void render_link(Vector2 src, Vector2 dest) = 0;
00061 };
00062
00063 }  // namespace gui::internal
00064
00065 #endif  // GUI_BASE_GUI_HPP_
```

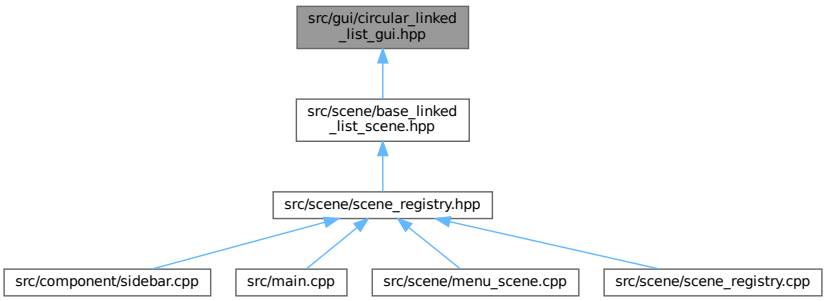
7.51 src/gui/circular_linked_list_gui.hpp File Reference

```
#include <array>
#include <cstddef>
#include <iostream>
#include "base_gui.hpp"
#include "constants.hpp"
#include "core/doubly_linked_list.hpp"
#include "node_gui.hpp"
#include "raylib.h"
#include "rlgl.h"
#include "settings.hpp"
```

Include dependency graph for circular_linked_list_gui.hpp:



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



Classes

- class gui::GuiCircularLinkedList< T >

  The GUI circular linked list container.

Namespaces

- namespace gui

7.52 circular_linked_list_gui.hpp

Go to the documentation of this file.

```
00001 #ifndef GUI_CIRCULAR_LINKED_LIST_GUI_HPP_
00002 #define GUI_CIRCULAR_LINKED_LIST_GUI_HPP_
00003
00004 #include <array>
00005 #include <cstddef>
00006 #include <iostream>
00007
00008 #include "base_gui.hpp"
00009 #include "constants.hpp"
00010 #include "core/doubly_linked_list.hpp"
00011 #include "node_gui.hpp"
00012 #include "raylib.h"
00013 #include "rlgl.h"
00014 #include "settings.hpp"
00015
00016 namespace gui {
00017
00023 template<typename T>
00024 class GuiCircularLinkedList : public core::DoublyLinkedList<GuiNode<T»,
00025                               public internal::Base {
00026 private:
00027     using Base = core::DoublyLinkedList<GuiNode<T>>;
00028
00029 public:
00030     using Base::Base;
00031
00032     using Base::empty;
00033     using Base::size;
00034
00039     GuiCircularLinkedList(std::initializer_list<GuiNode<T>> init_list);
00040
00046     void insert(std::size_t index, const T& elem);
00047
00051     void update() override;
00052
00056     void render() override;
```

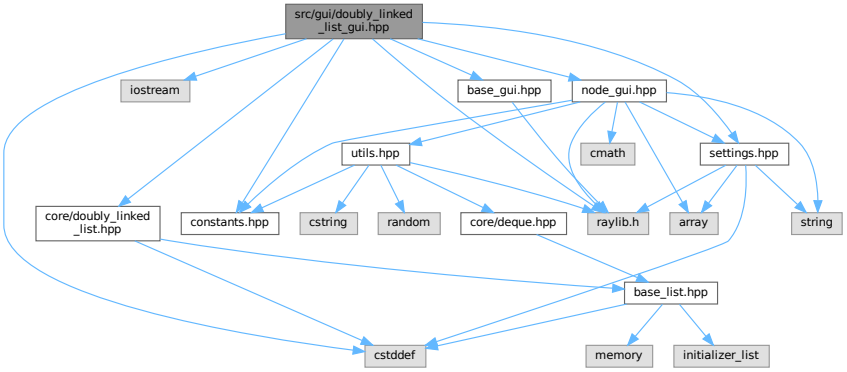
```
00057
00061     void init_label();
00062
00063 private:
00067     static constexpr Vector2 head_pos{
00068         constants::scene_width / 2.0F - 15 * GuiNode<T>::radius,
00069         constants::scene_height / 2.0F};
00070
00071     using Base::m_head;
00072     using Base::m_tail;
00073
00079     void render_link(Vector2 src, Vector2 dest) override;
00080
00084     void render_back_link();
00085 };
00086
00087 template<typename T>
00088 void GuiCircularLinkedList<T>::init_label() {
00089     if (m_head != nullptr) {
00090         m_head->data.set_label("head");
00091     }
00092
00093     if (m_tail != nullptr) {
00094         if (m_head == m_tail) {
00095             m_tail->data.set_label("head/tail");
00096         } else {
00097             m_tail->data.set_label("tail");
00098         }
00099     }
00100 }
00101
00102 template<typename T>
00103 GuiCircularLinkedList<T>::GuiCircularLinkedList(
00104     std::initializer_list<GuiNode<T>> init_list)
00105     : core::DoublyLinkedList<GuiNode<T»(init_list) {
00106     init_label();
00107 }
00108
00109 template<typename T>
00110 void GuiCircularLinkedList<T>::insert(std::size_t index, const T& elem) {
00111     Base::insert(index, GuiNode{elem});
00112 }
00113
00114 template<typename T>
00115 void GuiCircularLinkedList<T>::render_link(Vector2 src, Vector2 dest) {
00116     constexpr int radius = GuiNode<T>::radius;
00117     constexpr float scaled_len = radius / 8.0F;
00118
00119     // straight line
00120     Vector2 link_pos{src.x + radius, src.y - scaled_len};
00121     Vector2 link_size{dest.x - src.x - 2 * radius, 2 * scaled_len};
00122
00123     // arrow
00124     constexpr int arrow_size = scaled_len * 5;
00125     Vector2 head{dest.x - radius + scaled_len / 2, src.y};
00126     Vector2 side_top{head.x - arrow_size, head.y - arrow_size};
00127     Vector2 side_bot{head.x - arrow_size, head.y + arrow_size};
00128
00129     // draw both
00130     const Settings& settings = Settings::get_instance();
00131     DrawRectangleV(link_pos, link_size, settings.get_color(1));
00132     DrawTriangle(head, side_top, side_bot, settings.get_color(1));
00133 }
00134
00135 template<typename T>
00136 void GuiCircularLinkedList<T>::render_back_link() {
00137     if (m_head == nullptr && m_tail == nullptr) {
00138         return;
00139     }
00140
00141     constexpr int num_points = 5;
00142     const Vector2 head_pos = m_head->data.get_pos();
00143     const Vector2 tail_pos = m_tail->data.get_pos();
00144     constexpr int radius = GuiNode<T>::radius;
00145     constexpr float scaled_len = radius / 8.0F;
00146
00147     std::array<Vector2, num_points> points{{
00148         tail_pos,
00149         {tail_pos.x + 2 * radius, tail_pos.y},
00150         {tail_pos.x + 2 * radius, tail_pos.y + 3 * radius},
00151         {head_pos.x, tail_pos.y + 3 * radius},
00152         head_pos,
00153     }};
00154
00155     constexpr int arrow_size = scaled_len * 5;
00156     Vector2 head{head_pos.x, head_pos.y + radius - scaled_len / 2};
00157     Vector2 side_left{head.x - arrow_size, head.y + arrow_size};
```

```
00158     Vector2 side_right{head.x + arrow_size, head.y + arrow_size};
00159
00160     const Settings& settings = Settings::get_instance();
00161     rlSetLineWidth(2 * scaled_len);
00162     DrawLineStrip(points.data(), num_points, settings.get_color(1));
00163     DrawTriangle(head, side_left, side_right, settings.get_color(1));
00164 }
00165
00166 template<typename T>
00167 void GuiCircularLinkedList<T>::render() {
00168     update();
00169
00170     render_back_link();
00171     for (auto* ptr = m_head; ptr != nullptr; ptr = ptr->next) {
00172         if (ptr->next != nullptr) {
00173             render_link(ptr->data.get_pos(), ptr->next->data.get_pos());
00174         }
00175
00176         ptr->data.render();
00177     }
00178 }
00179
00180 template<typename T>
00181 void GuiCircularLinkedList<T>::update() {
00182     // TODO: if not outdated then return
00183
00184     std::size_t pos = 0;
00185
00186     for (auto* ptr = m_head; ptr != nullptr; ptr = ptr->next) {
00187         ptr->data.set_pos(
00188             {head_pos.x + 4 * GuiNode<T>::radius * pos, head_pos.y});
00189         ++pos;
00190     }
00191 }
00192
00193 }  // namespace gui
00194
00195 #endif  // GUI_CIRCULAR_LINKED_LIST_GUI_HPP_
```

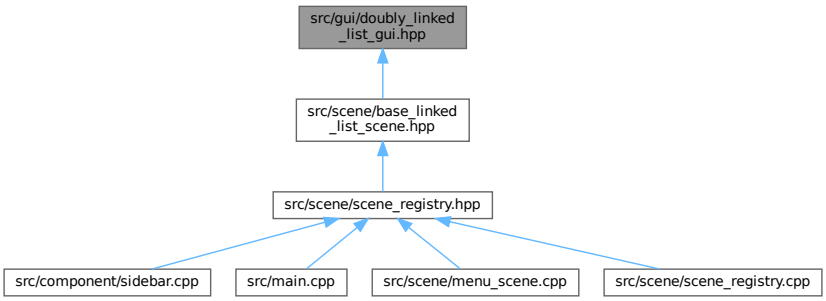
7.53 src/gui/doubly_linked_list_gui.hpp File Reference

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

Include dependency graph for doubly_linked_list_gui.hpp:



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



Classes

- class gui::GuiDoublyLinkedList< T >

  The GUI doubly linked list container.

Namespaces

- namespace gui

7.54 doubly_linked_list_gui.hpp

Go to the documentation of this file.

```
00001 #ifndef GUI_DOUBLY_LINKED_LIST_GUI_HPP_
00002 #define GUI_DOUBLY_LINKED_LIST_GUI_HPP_
00003
00004 #include <cstddef>
00005 #include <iostream>
00006
00007 #include "base_gui.hpp"
00008 #include "constants.hpp"
00009 #include "core/doubly_linked_list.hpp"
00010 #include "node_gui.hpp"
00011 #include "raylib.h"
00012 #include "settings.hpp"
00013
00014 namespace gui {
00015
00021 template<typename T>
00022 class GuiDoublyLinkedList : public core::DoublyLinkedList<GuiNode<T»,
00023                             public internal::Base {
00024 private:
00025     using Base = core::DoublyLinkedList<GuiNode<T>>;
00026
00027 public:
00028     using Base::Base;
00029
00030     using Base::empty;
00031     using Base::size;
00032
00037     GuiDoublyLinkedList(std::initializer_list<GuiNode<T>> init_list);
00038
00044     void insert(std::size_t index, const T& elem);
00045
00049     void update() override;
00050
00054     void render() override;
00055
00059     void init_label();
```

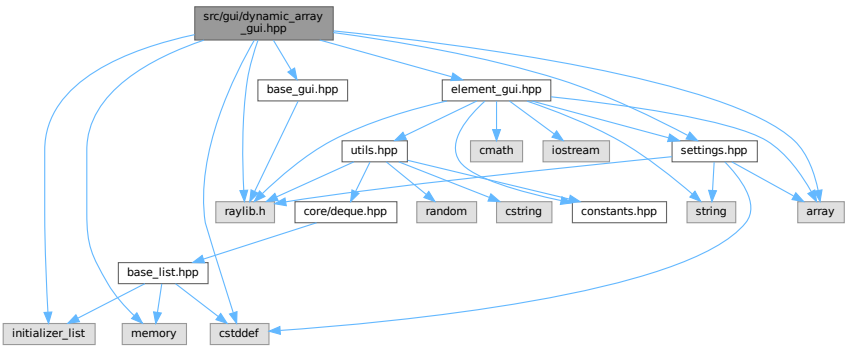
```
00060
00061 private:
00065     static constexpr Vector2 head_pos{
00066         constants::scene_width / 2.0F - 15 * GuiNode<T>::radius,
00067         constants::scene_height / 2.0F};
00068
00069     using Base::m_head;
00070     using Base::m_tail;
00071
00077     void render_link(Vector2 src, Vector2 dest) override;
00078 };
00079
00080 template<typename T>
00081 void GuiDoublyLinkedList<T>::init_label() {
00082     if (m_head != nullptr) {
00083         m_head->data.set_label("head");
00084     }
00085
00086     if (m_tail != nullptr) {
00087         if (m_head == m_tail) {
00088             m_tail->data.set_label("head/tail");
00089         } else {
00090             m_tail->data.set_label("tail");
00091         }
00092     }
00093 }
00094
00095 template<typename T>
00096 GuiDoublyLinkedList<T>::GuiDoublyLinkedList(
00097     std::initializer_list<GuiNode<T>> init_list)
00098     : core::DoublyLinkedList<GuiNode<T»(init_list) {
00099     init_label();
00100 }
00101
00102 template<typename T>
00103 void GuiDoublyLinkedList<T>::insert(std::size_t index, const T& elem) {
00104     Base::insert(index, GuiNode{elem});
00105 }
00106
00107 template<typename T>
00108 void GuiDoublyLinkedList<T>::render_link(Vector2 src, Vector2 dest) {
00109     constexpr int radius = GuiNode<T>::radius;
00110     constexpr float scaled_len = radius / 8.0F;
00111
00112     // straight line
00113     Vector2 link_pos{src.x + radius, src.y - scaled_len};
00114     Vector2 link_size{dest.x - src.x - 2 * radius, 2 * scaled_len};
00115
00116     // right arrow
00117     constexpr int arrow_size = scaled_len * 5;
00118     Vector2 right_head{dest.x - radius + scaled_len / 2, src.y};
00119     Vector2 right_side_top{right_head.x - arrow_size,
00120                            right_head.y - arrow_size};
00121     Vector2 right_side_bot{right_head.x - arrow_size,
00122                            right_head.y + arrow_size};
00123
00124     // left arrow
00125     Vector2 left_head{src.x + radius - scaled_len / 2, src.y};
00126     Vector2 left_side_top{left_head.x + arrow_size, left_head.y - arrow_size};
00127     Vector2 left_side_bot{left_head.x + arrow_size, left_head.y + arrow_size};
00128
00129     // draw all
00130     const Settings& settings = Settings::get_instance();
00131     DrawRectangleV(link_pos, link_size, settings.get_color(1));
00132     DrawTriangle(right_head, right_side_top, right_side_bot,
00133                  settings.get_color(1));
00134     DrawTriangle(left_head, left_side_bot, left_side_top,
00135                  settings.get_color(1));
00136 }
00137
00138 template<typename T>
00139 void GuiDoublyLinkedList<T>::render() {
00140     update();
00141
00142     for (auto* ptr = m_head; ptr != nullptr; ptr = ptr->next) {
00143         if (ptr->next != nullptr) {
00144             render_link(ptr->data.get_pos(), ptr->next->data.get_pos());
00145         }
00146
00147         ptr->data.render();
00148     }
00149 }
00150
00151 template<typename T>
00152 void GuiDoublyLinkedList<T>::update() {
00153     // TODO: if not outdated then return
00154
```

```
00155     std::size_t pos = 0;
00156
00157     for (auto* ptr = m_head; ptr != nullptr; ptr = ptr->next) {
00158         ptr->data.set_pos(
00159             {head_pos.x + 4 * GuiNode<T>::radius * pos, head_pos.y});
00160         ++pos;
00161     }
00162 }
00163
00164 }  // namespace gui
00165
00166 #endif  // GUI_DOUBLY_LINKED_LIST_GUI_HPP_
```

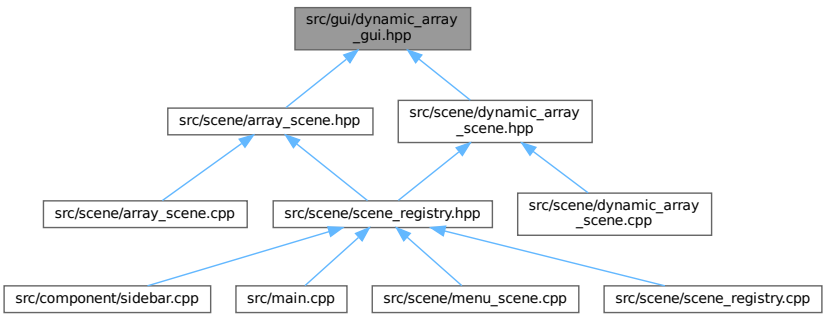
7.55 src/gui/dynamic_array_gui.hpp File Reference

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

Include dependency graph for dynamic_array_gui.hpp:



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



Classes

- class gui::GuiDynamicArray< T >

  The GUI dynamic array container.

Namespaces

- namespace gui

7.56 dynamic_array_gui.hpp

Go to the documentation of this file.

```
00001 #ifndef GUI_DYNAMIC_ARRAY_GUI_HPP_
00002 #define GUI_DYNAMIC_ARRAY_GUI_HPP_
00003
00004 #include <array>
00005 #include <cstddef>
00006 #include <initializer_list>
00007 #include <memory>
00008
00009 #include "base_gui.hpp"
00010 #include "element_gui.hpp"
00011 #include "raylib.h"
00012 #include "settings.hpp"
00013
00014 namespace gui {
00015
00021 template<typename T>
00022 class GuiDynamicArray : public internal::Base {
00023 public:
00027     GuiDynamicArray();
00028
00033     GuiDynamicArray(std::initializer_list<T> init_list);
00034
00039     GuiDynamicArray(const GuiDynamicArray& other);
00040
00045     GuiDynamicArray(GuiDynamicArray&& other) noexcept;
00046
00053     GuiDynamicArray& operator=(const GuiDynamicArray& other);
00054
00061     GuiDynamicArray& operator=(GuiDynamicArray&& other) noexcept;
00062
00066     ~GuiDynamicArray() override;
00067
00071     void update() override;
00072
00076     void render() override;
00077
00083     T& operator[](std::size_t idx);
00084
00090     T operator[](std::size_t idx) const;
00091
00097     void set_color_index(std::size_t idx, int color_index);
00098
00103     void reserve(std::size_t capacity);
00104
00108     void shrink_to_fit();
00109
00114     std::size_t capacity() const;
00115
00120     std::size_t size() const;
00121
00126     void push(const T& value);
00127
00131     void pop();
00132
00133 private:
00137     static constexpr Vector2 head_pos{
00138         constants::scene_width / 2.0F - 15 * GuiElement<T>::side,
00139         constants::scene_height / 2.0F};
00140
00144     std::size_t m_capacity{2};
00145
00149     std::size_t m_size{};
00150
```

```
00154     GuiElement<T>* m_ptr{nullptr};
00155
00161     void render_link(Vector2 src, Vector2 dest) override;
00162 };
00163
00164 template<typename T>
00165 void GuiDynamicArray<T>::reserve(std::size_t capacity) {
00166     capacity = std::min(capacity, static_cast<std::size_t>(8));
00167     if (m_capacity > capacity) {
00168         return;
00169     }
00170
00171     auto* new_ptr = static_cast<GuiElement<T>*>(
00172         ::operator new[](capacity * sizeof(GuiElement<T>)));
00173     for (auto i = 0; i < m_size; ++i) {
00174         ::new (&new_ptr[i]) GuiElement<T>(std::move(m_ptr[i]));
00175         m_ptr[i].~GuiElement<T>();
00176     }
00177     for (auto i = m_size; i < capacity; ++i) {
00178         ::new (&new_ptr[i]) GuiElement<T>();
00179         new_ptr[i].set_index(i);
00180     }
00181
00182     ::operator delete[](m_ptr);
00183     m_ptr = new_ptr;
00184     m_capacity = capacity;
00185 }
00186
00187 template<typename T>
00188 void GuiDynamicArray<T>::shrink_to_fit() {
00189     if (m_capacity == m_size) {
00190         return;
00191     }
00192
00193     auto* new_ptr = static_cast<GuiElement<T>*>(
00194         ::operator new[](m_size * sizeof(GuiElement<T>)));
00195     for (auto i = 0; i < m_size; ++i) {
00196         ::new (&new_ptr[i]) GuiElement<T>(std::move(m_ptr[i]));
00197     }
00198     for (auto i = 0; i < m_capacity; ++i) {
00199         m_ptr[i].~GuiElement<T>();
00200     }
00201
00202     ::operator delete[](m_ptr);
00203     m_ptr = new_ptr;
00204     m_capacity = m_size;
00205 }
00206
00207 template<typename T>
00208 GuiDynamicArray<T>::GuiDynamicArray() : m_ptr{new GuiElement<T>[m_capacity]} {
00209     for (auto i = 0; i < m_capacity; ++i) {
00210         m_ptr[i].set_index(i);
00211     }
00212 }
00213
00214 template<typename T>
00215 GuiDynamicArray<T>::GuiDynamicArray(std::initializer_list<T> init_list)
00216     : m_size{init_list.size()}, m_ptr{new GuiElement<T>[m_capacity]} {
00217     reserve(m_size);
00218
00219     for (std::size_t idx = 0; auto elem : init_list) {
00220         *(m_ptr + idx).set_value(elem);
00221         *(m_ptr + idx).set_color(Settings::get_instance().get_color(0));
00222     }
00223 }
00224
00225 template<typename T>
00226 GuiDynamicArray<T>::GuiDynamicArray(const GuiDynamicArray<T>& other)
00227     : m_capacity{other.m_capacity},
00228       m_size{other.m_size},
00229       m_ptr{new GuiElement<T>[m_capacity]} {
00230     for (auto i = 0; i < m_capacity; ++i) {
00231         m_ptr[i] = other.m_ptr[i];
00232     }
00233 }
00234
00235 template<typename T>
00236 GuiDynamicArray<T>::GuiDynamicArray(GuiDynamicArray<T>&& other) noexcept
00237     : m_capacity{other.m_capacity}, m_size{other.m_size}, m_ptr{other.m_ptr} {
00238     other.m_capacity = 0;
00239     other.m_size = 0;
00240     other.m_ptr = nullptr;
00241 }
00242
00243 template<typename T>
00244 GuiDynamicArray<T>& GuiDynamicArray<T>::operator=(
00245     const GuiDynamicArray<T>& other) {
```

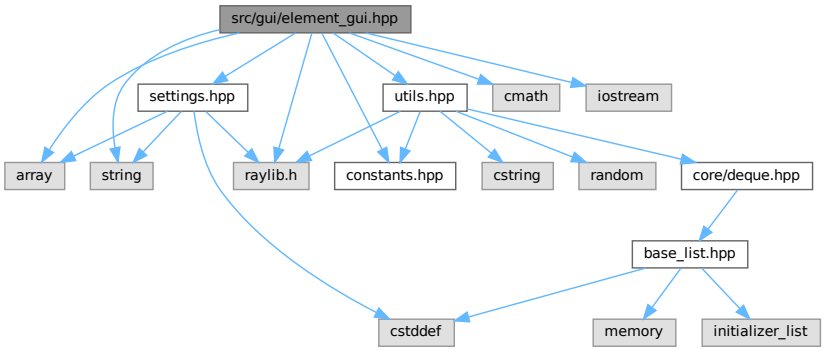
```
00246     if (&other != this) {
00247         m_capacity = other.m_capacity;
00248         m_size = other.m_size;
00249 
00250         m_ptr = new GuiDynamicArray<T>[m_capacity];
00251         for (auto i = 0; i < m_capacity; ++i) {
00252             m_ptr[i] = other.m_ptr[i];
00253         }
00254     }
00255 
00256     return *this;
00257 }
00258 
00259 template<typename T>
00260 GuiDynamicArray<T>& GuiDynamicArray<T>::operator=(
00261     GuiDynamicArray&& other) noexcept {
00262     m_capacity = other.m_capacity;
00263     m_size = other.m_size;
00264     m_ptr = other.m_ptr;
00265 
00266     other.m_capacity = 0;
00267     other.m_size = 0;
00268     other.m_ptr = nullptr;
00269 
00270     return *this;
00271 }
00272 
00273 template<typename T>
00274 GuiDynamicArray<T>::~GuiDynamicArray() {
00275     delete[] m_ptr;
00276 }
00277 
00278 template<typename T>
00279 void GuiDynamicArray<T>::render_link(Vector2 src, Vector2 dest) {}
00280 
00281 template<typename T>
00282 void GuiDynamicArray<T>::render() {
00283     update();
00284 
00285     std::size_t idx = 0;
00286 
00287     for (std::size_t i = 0; i < m_capacity; ++i) {
00288         m_ptr[i].render();
00289     }
00290 }
00291 
00292 template<typename T>
00293 void GuiDynamicArray<T>::update() {
00294     // TODO: if not outdated then return
00295 
00296     for (std::size_t i = 0; i < m_capacity; ++i) {
00297         m_ptr[i].set_pos(
00298             {head_pos.x + 4 * GuiElement<T>::side * i, head_pos.y});
00299     }
00300 }
00301 
00302 template<typename T>
00303 T& GuiDynamicArray<T>::operator[](std::size_t idx) {
00304     return m_ptr[idx].get_value();
00305 }
00306 
00307 template<typename T>
00308 T GuiDynamicArray<T>::operator[](std::size_t idx) const {
00309     return m_ptr[idx].get_value();
00310 }
00311 
00312 template<typename T>
00313 void GuiDynamicArray<T>::set_color_index(std::size_t idx, int color_index) {
00314     m_ptr[idx].set_color_index(color_index);
00315 }
00316 
00317 template<typename T>
00318 std::size_t GuiDynamicArray<T>::capacity() const {
00319     return m_capacity;
00320 }
00321 
00322 template<typename T>
00323 std::size_t GuiDynamicArray<T>::size() const {
00324     return m_size;
00325 }
00326 
00327 template<typename T>
00328 void GuiDynamicArray<T>::push(const T& value) {
00329     if (m_size == m_capacity) {
00330         reserve(std::max(m_capacity * 2, static_cast<std::size_t>(1)));
00331     }
00332 
```

```
00333     m_ptr[m_size].set_color_index(0);
00334     m_ptr[m_size].set_value(value);
00335     ++m_size;
00336 }
00337
00338 template<typename T>
00339 void GuiDynamicArray<T>::pop() {
00340     if (m_size >= 1) {
00341         m_ptr[m_size - 1].set_color_index(1);
00342         m_ptr[m_size - 1].set_value(0);
00343         --m_size;
00344     }
00345 }
00346
00347 }  // namespace gui
00348
00349 #endif  // GUI_DYNAMIC_ARRAY_GUI_HPP_
```

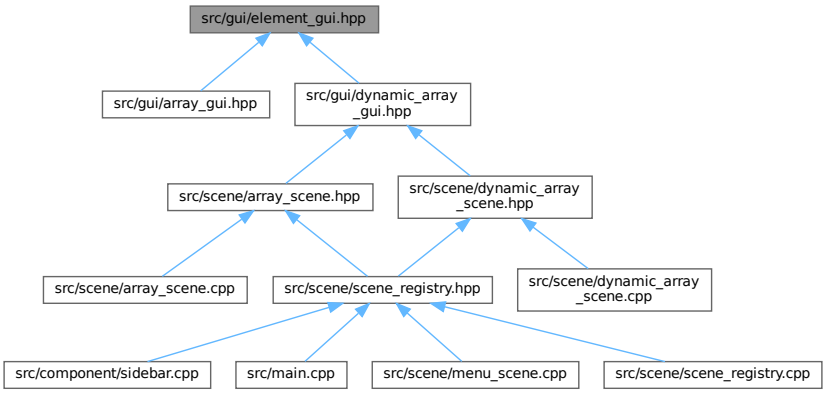
7.57 src/gui/element_gui.hpp File Reference

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

Include dependency graph for element_gui.hpp:



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



Classes

- class gui::GuiElement< T >

  The GUI element (used in arrays)

Namespaces

- namespace gui

7.58 element_gui.hpp

Go to the documentation of this file.

```
00001 #ifndef GUI_ELEMENT_GUI_HPP_
00002 #define GUI_ELEMENT_GUI_HPP_
00003
00004 #include <array>
00005 #include <cmath>
00006 #include <iostream>
00007 #include <string>
00008
00009 #include "constants.hpp"
00010 #include "raylib.h"
00011 #include "settings.hpp"
00012 #include "utils.hpp"
00013
00014 namespace gui {
00015
00021 template<typename T>
00022 class GuiElement {
00023 public:
00027     static constexpr int side = 20;
00028
00032     static constexpr Vector2 init_pos{
00033         constants::sidebar_width +
00034             static_cast<float>(constants::scene_width -
00035                                constants::sidebar_width) /
00036                 2,
00037         0};
00038
00042     GuiElement() = default;
00043
00049     GuiElement(const T& value, std::size_t index);
00050
00054     void render();
```

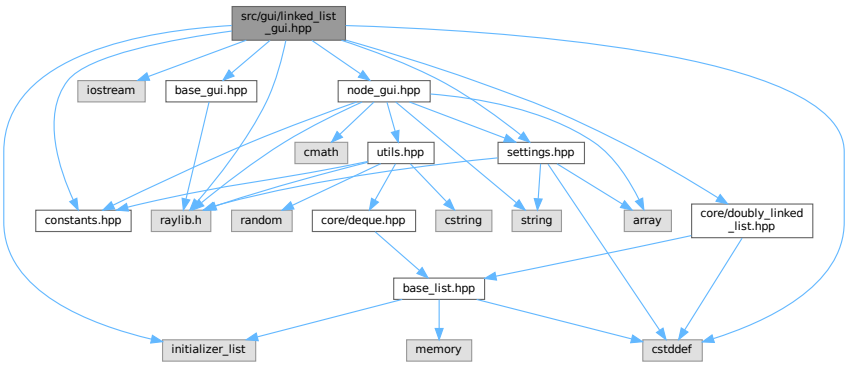
```
00055
00060     void set_pos(Vector2 pos);
00061
00066     void set_color_index(int color_index);
00067
00072     [[nodiscard]] Vector2 get_pos() const;
00073
00078     T& get_value();
00079
00084     T get_value() const;
00085
00090     void set_value(const T& value);
00091
00096     void set_index(std::size_t index);
00097
00098 private:
00102     T m_value{};
00103
00107     std::size_t m_index{};
00108
00112     Vector2 m_pos{init_pos};
00113
00117     static constexpr float eps = 1e-3;
00118
00122     int m_color_index{1};
00123 };
00124
00125 template<typename T>
00126 GuiElement<T>::GuiElement(const T& value, std::size_t index)
00127     : m_value{value}, m_index{index} {}
00128
00129 template<typename T>
00130 void GuiElement<T>::render() {
00131     constexpr int label_font_size = 25;
00132     constexpr int label_font_spacing = 2;
00133     const std::string label = std::to_string(m_value);
00134     const std::string index = std::to_string(m_index);
00135     const Settings& settings = Settings::get_instance();
00136
00137     const Vector2 label_size =
00138         utils::MeasureText(label.c_str(), label_font_size, label_font_spacing);
00139
00140     const Vector2 label_pos{m_pos.x - label_size.x / 2,
00141                             m_pos.y - label_size.y / 2};
00142
00143     const Vector2 index_size =
00144         utils::MeasureText(index.c_str(), label_font_size, label_font_spacing);
00145
00146     const Vector2 index_pos{m_pos.x - index_size.x / 2,
00147                             m_pos.y - 2 * side - index_size.y / 2};
00148
00149     const Color value_color =
00150         utils::adaptive_text_color(settings.get_color(m_color_index));
00151     const Color index_color =
00152         utils::adaptive_text_color(settings.get_color(Settings::num_color - 1));
00153
00154     DrawRectangle(m_pos.x - side,  // NOLINT
00155                   m_pos.y - side,  // NOLINT
00156                   2 * side, 2 * side, settings.get_color(m_color_index));
00157
00158     utils::DrawText(label.c_str(), label_pos, value_color, label_font_size,
00159                     label_font_spacing);
00160
00161     utils::DrawText(index.c_str(), index_pos, index_color, label_font_size,
00162                     label_font_spacing);
00163 }
00164
00165 template<typename T>
00166 void GuiElement<T>::set_pos(Vector2 pos) {
00167     m_pos = pos;
00168 }
00169
00170 template<typename T>
00171 void GuiElement<T>::set_color_index(int color_index) {
00172     m_color_index = color_index;
00173 }
00174
00175 template<typename T>
00176 T& GuiElement<T>::get_value() {
00177     return m_value;
00178 }
00179
00180 template<typename T>
00181 T GuiElement<T>::get_value() const {
00182     return m_value;
00183 }
00184
```

```
00185 template<typename T>
00186 void GuiElement<T>::set_value(const T& value) {
00187     m_value = value;
00188 }
00189
00190 template<typename T>
00191 void GuiElement<T>::set_index(std::size_t index) {
00192     m_index = index;
00193 }
00194
00195 }  // namespace gui
00196
00197 #endif  // GUI_ELEMENT_GUI_HPP_
```

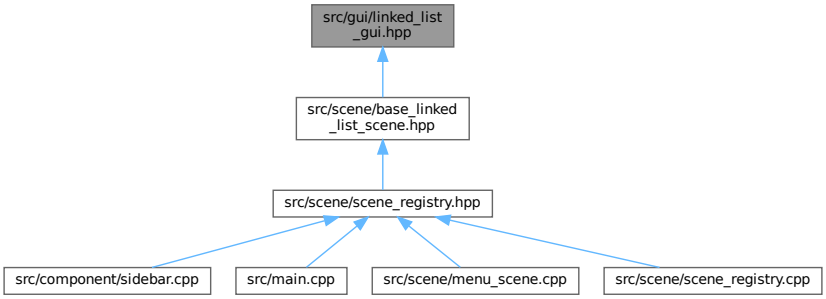
7.59 src/gui/linked_list_gui.hpp File Reference

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

Include dependency graph for linked_list_gui.hpp:



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



Classes

- class gui::GuiLinkedList< T >

  The GUI linked list container.

Namespaces

- namespace gui

7.60 linked_list_gui.hpp

Go to the documentation of this file.

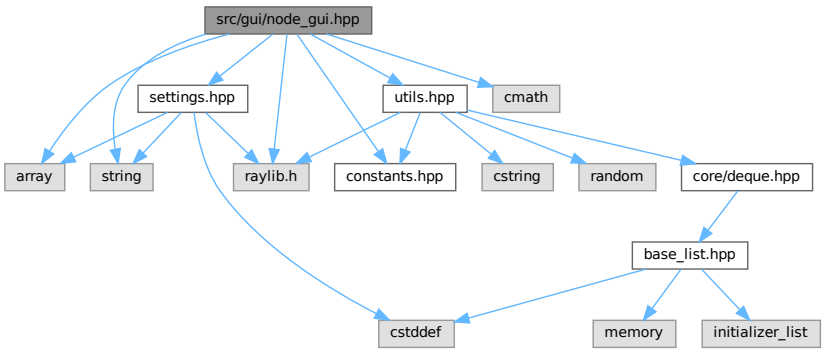
```
00001 #ifndef GUI_LINKED_LIST_GUI_HPP_
00002 #define GUI_LINKED_LIST_GUI_HPP_
00003
00004 #include <cstddef>
00005 #include <initializer_list>
00006 #include <iostream>
00007
00008 #include "base_gui.hpp"
00009 #include "constants.hpp"
00010 #include "core/doubly_linked_list.hpp"
00011 #include "node_gui.hpp"
00012 #include "raylib.h"
00013 #include "settings.hpp"
00014
00015 namespace gui {
00016
00022 template<typename T>
00023 class GuiLinkedList : public core::DoublyLinkedList<GuiNode<T»,
00024                       public internal::Base {
00025 private:
00026     using Base = core::DoublyLinkedList<GuiNode<T>>;
00027
00028 public:
00029     using Base::Base;
00030
00031     using Base::empty;
00032     using Base::size;
00033
00038     GuiLinkedList(std::initializer_list<GuiNode<T>> init_list);
00039
00045     void insert(std::size_t index, const T& elem);
00046
00050     void update() override;
00051
00055     void render() override;
00056
00060     void init_label();
00061
00062 private:
00066     static constexpr Vector2 head_pos{
00067         constants::scene_width / 2.0F - 15 * GuiNode<T>::radius,
00068         constants::scene_height / 2.0F};
00069
00070     using Base::m_head;
00071     using Base::m_tail;
00072
00078     void render_link(Vector2 src, Vector2 dest) override;
00079 };
00080
00081 template<typename T>
00082 void GuiLinkedList<T>::init_label() {
00083     if (m_head != nullptr) {
00084         m_head->data.set_label("head");
00085     }
00086
00087     if (m_tail != nullptr) {
00088         if (m_head == m_tail) {
00089             m_tail->data.set_label("head/tail");
00090         } else {
00091             m_tail->data.set_label("tail");
00092         }
00093     }
00094 }
```

```
00095
00096 template<typename T>
00097 GuiLinkedList<T>::GuiLinkedList(std::initializer_list<GuiNode<T>> init_list)
00098     : core::DoublyLinkedList<GuiNode<T»(init_list) {
00099     init_label();
00100 }
00101
00102 template<typename T>
00103 void GuiLinkedList<T>::insert(std::size_t index, const T& elem) {
00104     Base::insert(index, GuiNode{elem});
00105 }
00106
00107 template<typename T>
00108 void GuiLinkedList<T>::render_link(Vector2 src, Vector2 dest) {
00109     constexpr int radius = GuiNode<T>::radius;
00110     constexpr float scaled_len = radius / 8.0F;
00111
00112     // straight line
00113     Vector2 link_pos{src.x + radius, src.y - scaled_len};
00114     Vector2 link_size{dest.x - src.x - 2 * radius, 2 * scaled_len};
00115
00116     // arrow
00117     constexpr int arrow_size = scaled_len * 5;
00118     Vector2 head{dest.x - radius + scaled_len / 2, src.y};
00119     Vector2 side_top{head.x - arrow_size, head.y - arrow_size};
00120     Vector2 side_bot{head.x - arrow_size, head.y + arrow_size};
00121
00122     // draw both
00123     DrawRectangleV(link_pos, link_size, Settings::get_instance().get_color(1));
00124     DrawTriangle(head, side_top, side_bot,
00125                  Settings::get_instance().get_color(1));
00126 }
00127
00128 template<typename T>
00129 void GuiLinkedList<T>::render() {
00130     update();
00131
00132     for (auto* ptr = m_head; ptr != nullptr; ptr = ptr->next) {
00133         if (ptr->next != nullptr) {
00134             render_link(ptr->data.get_pos(), ptr->next->data.get_pos());
00135         }
00136
00137         ptr->data.render();
00138     }
00139 }
00140
00141 template<typename T>
00142 void GuiLinkedList<T>::update() {
00143     // TODO: if not outdated then return
00144
00145     std::size_t pos = 0;
00146
00147     for (auto* ptr = m_head; ptr != nullptr; ptr = ptr->next) {
00148         ptr->data.set_pos(
00149             {head_pos.x + 4 * GuiNode<T>::radius * pos, head_pos.y});
00150         ++pos;
00151     }
00152 }
00153
00154 }  // namespace gui
00155
00156 #endif  // GUI_LINKED_LIST_GUI_HPP_
```

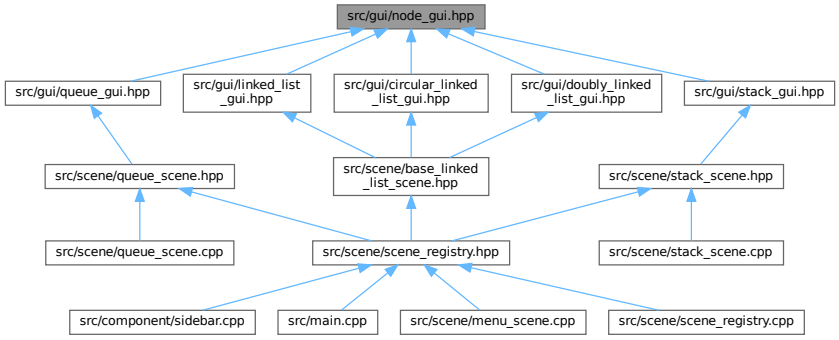
7.61 src/gui/node_gui.hpp File Reference

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

Include dependency graph for node_gui.hpp:



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



Classes

- class gui::GuiNode< T >

  The GUI node (used in linked lists)

Namespaces

- namespace gui

7.62 node_gui.hpp

Go to the documentation of this file.

```
00001 #ifndef GUI_NODE_GUI_HPP_
00002 #define GUI_NODE_GUI_HPP_
00003
00004 #include <array>
00005 #include <cmath>
00006 #include <string>
```

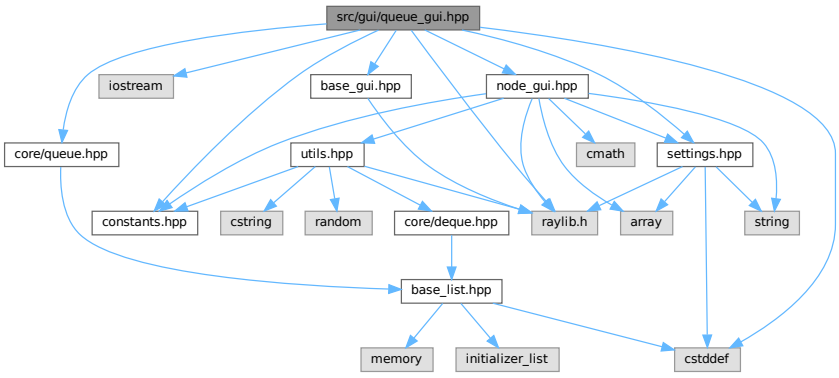
```
00007
00008 #include "constants.hpp"
00009 #include "raylib.h"
00010 #include "settings.hpp"
00011 #include "utils.hpp"
00012
00013 namespace gui {
00014
00020 template<typename T>
00021 class GuiNode {
00022 public:
00026     static constexpr int radius = 20;
00027
00031     explicit GuiNode(const T& value);
00032
00036     void render();
00037
00042     void set_pos(Vector2 pos);
00043
00048     [[nodiscard]] Vector2 get_pos() const;
00049
00054     void set_color_index(int color_index);
00055
00060     void set_value(const T& value);
00061
00066     T& get_value();
00067
00072     void set_label(const char* label);
00073
00074 private:
00078     T m_value{};
00079
00083     int m_color_index{0};
00084
00088     Vector2 m_pos{constants::sidebar_width +
00089                       static_cast<float>(constants::scene_width -
00090                                          constants::sidebar_width) /
00091                           2,
00092                   0};
00093
00097     static constexpr float eps = 1e-3;
00098
00102     const char* m_label{};
00103 };
00104
00105 template<typename T>
00106 GuiNode<T>::GuiNode(const T& value) : m_value{value} {}
00107
00108 template<typename T>
00109 void GuiNode<T>::render() {
00110     constexpr int label_font_size = 25;
00111     constexpr int label_font_spacing = 2;
00112     const std::string value = std::to_string(m_value);
00113     const Settings& settings = Settings::get_instance();
00114
00115     const Vector2 value_size =
00116         utils::MeasureText(value.c_str(), label_font_size, label_font_spacing);
00117
00118     const Vector2 value_pos{m_pos.x - value_size.x / 2,
00119                             m_pos.y - value_size.y / 2};
00120
00121     const Vector2 label_size =
00122         utils::MeasureText(m_label, label_font_size, label_font_spacing);
00123
00124     const Vector2 label_pos{m_pos.x - label_size.x / 2,
00125                             m_pos.y - 2 * label_size.y};
00126
00127     const Color value_color =
00128         utils::adaptive_text_color(settings.get_color(m_color_index));
00129
00130     DrawCircleV(m_pos, radius, settings.get_color(m_color_index));
00131     utils::DrawText(value.c_str(), value_pos, value_color, label_font_size,
00132                     label_font_spacing);
00133
00134     utils::DrawText(m_label, label_pos, settings.get_color(5), label_font_size,
00135                     label_font_spacing);
00136 }
00137
00138 template<typename T>
00139 void GuiNode<T>::set_color_index(int color_index) {
00140     m_color_index = color_index;
00141 }
00142
00143 template<typename T>
00144 void GuiNode<T>::set_value(const T& value) {
00145     m_value = value;
00146 }
```

```
00147
00148 template<typename T>
00149 T& GuiNode<T>::get_value() {
00150     return m_value;
00151 }
00152
00153 template<typename T>
00154 void GuiNode<T>::set_pos(Vector2 pos) {
00155     m_pos = pos;
00156 }
00157
00158 template<typename T>
00159 Vector2 GuiNode<T>::get_pos() const {
00160     return m_pos;
00161 }
00162
00163 template<typename T>
00164 void GuiNode<T>::set_label(const char* label) {
00165     m_label = label;
00166 }
00167
00168 }  // namespace gui
00169
00170 #endif  // GUI_NODE_GUI_HPP_
```

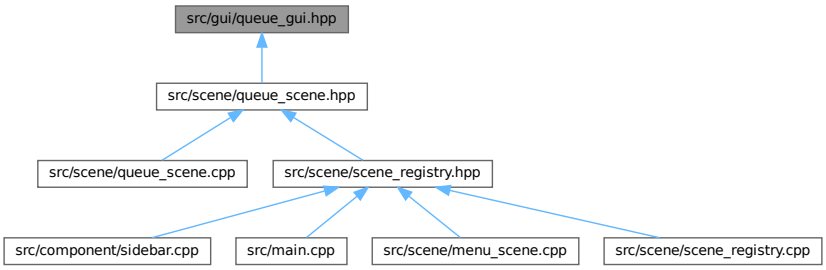
7.63 src/gui/queue_gui.hpp File Reference

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

Include dependency graph for queue_gui.hpp:



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



Classes

- class gui::GuiQueue< T >

  The GUI queue container.

Namespaces

- namespace gui

7.64 queue_gui.hpp

Go to the documentation of this file.

```
00001 #ifndef GUI_QUEUE_GUI_HPP_
00002 #define GUI_QUEUE_GUI_HPP_
00003
00004 #include <cstddef>
00005 #include <iostream>
00006
00007 #include "base_gui.hpp"
00008 #include "constants.hpp"
00009 #include "core/queue.hpp"
00010 #include "node_gui.hpp"
00011 #include "raylib.h"
00012 #include "settings.hpp"
00013
00014 namespace gui {
00015
00021 template<typename T>
00022 class GuiQueue : public core::Queue<GuiNode<T>, public internal::Base {
00023 private:
00024     using Base = core::Queue<GuiNode<T>>;
00025
00026 public:
00027     using Base::Base;
00028
00029     using Base::empty;
00030     using Base::size;
00031
00036     GuiQueue(std::initializer_list<GuiNode<T>> init_list);
00037
00042     void push(const T& elem);
00043
00047     void pop();
00048
00054     void push_front(const T& elem);
00055
00059     void pop_back();
00060
00064     void update() override;
00065
```

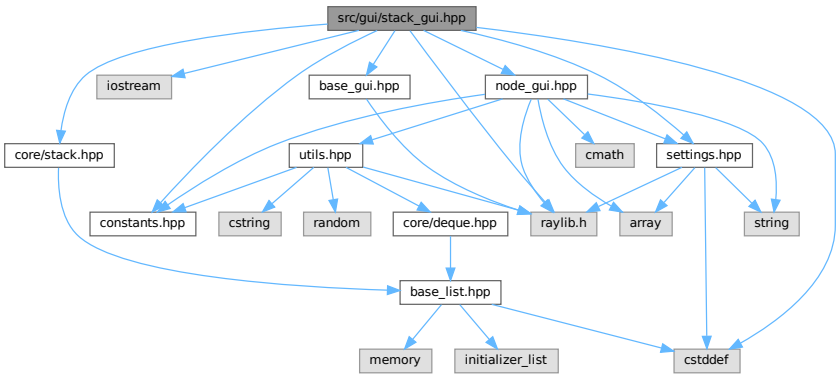
```
00069     void render() override;
00070
00074     void init_label();
00075
00076 private:
00080     static constexpr Vector2 head_pos{
00081         constants::scene_width / 2.0F - 15 * GuiNode<T>::radius,
00082         constants::scene_height / 2.0F};
00083
00084     using Base::m_head;
00085     using Base::m_tail;
00086
00092     void render_link(Vector2 src, Vector2 dest) override;
00093 };
00094
00095 template<typename T>
00096 void GuiQueue<T>::init_label() {
00097     if (m_head != nullptr) {
00098         m_head->data.set_label("head");
00099     }
00100
00101     if (m_tail != nullptr) {
00102         if (m_head == m_tail) {
00103             m_tail->data.set_label("head/tail");
00104         } else {
00105             m_tail->data.set_label("tail");
00106         }
00107     }
00108 }
00109
00110 template<typename T>
00111 GuiQueue<T>::GuiQueue(std::initializer_list<GuiNode<T>> init_list)
00112     : core::Queue<GuiNode<T»(init_list) {
00113     init_label();
00114 }
00115
00116 template<typename T>
00117 void GuiQueue<T>::push(const T& elem) {
00118     Base::push(GuiNode<T>{elem});
00119 }
00120
00121 template<typename T>
00122 void GuiQueue<T>::pop() {
00123     Base::pop();
00124 }
00125
00126 template<typename T>
00127 void GuiQueue<T>::push_front(const T& elem) {
00128     Base::push_front(GuiNode<T>{elem});
00129 }
00130
00131 template<typename T>
00132 void GuiQueue<T>::pop_back() {
00133     Base::pop_back();
00134 }
00135
00136 template<typename T>
00137 void GuiQueue<T>::render_link(Vector2 src, Vector2 dest) {
00138     constexpr int radius = GuiNode<T>::radius;
00139     constexpr float scaled_len = radius / 8.0F;
00140
00141     // straight line
00142     Vector2 link_pos{src.x + radius, src.y - scaled_len};
00143     Vector2 link_size{dest.x - src.x - 2 * radius, 2 * scaled_len};
00144
00145     // arrow
00146     constexpr int arrow_size = scaled_len * 5;
00147     Vector2 head{dest.x - radius + scaled_len / 2, src.y};
00148     Vector2 side_top{head.x - arrow_size, head.y - arrow_size};
00149     Vector2 side_bot{head.x - arrow_size, head.y + arrow_size};
00150
00151     // draw both
00152     DrawRectangleV(link_pos, link_size, Settings::get_instance().get_color(1));
00153     DrawTriangle(head, side_top, side_bot,
00154                  Settings::get_instance().get_color(1));
00155 }
00156
00157 template<typename T>
00158 void GuiQueue<T>::render() {
00159     update();
00160
00161     for (auto* ptr = m_head; ptr != nullptr; ptr = ptr->next) {
00162         if (ptr->next != nullptr) {
00163             render_link(ptr->data.get_pos(), ptr->next->data.get_pos());
00164         }
00165
00166         ptr->data.render();
```

```
00167     }
00168 }
00169
00170 template<typename T>
00171 void GuiQueue<T>::update() {
00172     // TODO: if not outdated then return
00173
00174     std::size_t pos = 0;
00175
00176     for (auto* ptr = m_head; ptr != nullptr; ptr = ptr->next) {
00177         ptr->data.set_pos(
00178             {head_pos.x + 4 * GuiNode<T>::radius * pos, head_pos.y});
00179         ++pos;
00180     }
00181 }
00182
00183 }  // namespace gui
00184
00185 #endif  // GUI_QUEUE_GUI_HPP_
```

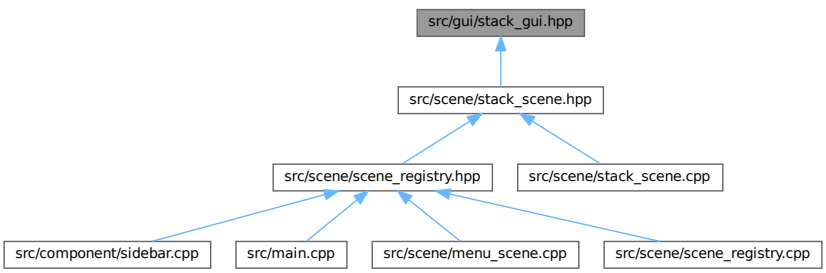
7.65 src/gui/stack_gui.hpp File Reference

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

Include dependency graph for stack_gui.hpp:



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



Classes

- class gui::GuiStack< T >

  The GUI stack container.

Namespaces

- namespace gui

7.66 stack_gui.hpp

Go to the documentation of this file.

```
00001 #ifndef GUI_STACK_GUI_HPP_
00002 #define GUI_STACK_GUI_HPP_
00003
00004 #include <cstddef>
00005 #include <iostream>
00006
00007 #include "base_gui.hpp"
00008 #include "constants.hpp"
00009 #include "core/stack.hpp"
00010 #include "node_gui.hpp"
00011 #include "raylib.h"
00012 #include "settings.hpp"
00013
00014 namespace gui {
00015
00021 template<typename T>
00022 class GuiStack : public core::Stack<GuiNode<T»>, public internal::Base {
00023 private:
00024     using Base = core::Stack<GuiNode<T>>;
00025
00026 public:
00027     using Base::Base;
00028
00029     using Base::empty;
00030     using Base::size;
00031
00036     GuiStack(std::initializer_list<GuiNode<T>> init_list);
00037
00042     void push(const T& elem);
00043
00047     void pop();
00048
00052     void update() override;
00053
00057     void render() override;
00058
00062     void init_label();
00063
00064 private:
00068     static constexpr Vector2 head_pos{
00069         constants::scene_width / 2.0F - GuiNode<T>::radius / 2.0F,
00070         GuiNode<T>::radius * 4.0F};
00071
00072     using Base::m_head;
00073     using Base::m_tail;
00074
00080     void render_link(Vector2 src, Vector2 dest) override;
00081 };
00082
00083 template<typename T>
00084 void GuiStack<T>::init_label() {
00085     if (m_head != nullptr) {
00086         m_head->data.set_label("head");
00087     }
00088 }
00089
00090 template<typename T>
00091 GuiStack<T>::GuiStack(std::initializer_list<GuiNode<T>> init_list)
00092     : core::Stack<GuiNode<T»(init_list) {
00093     init_label();
00094 }
00095
00096 template<typename T>
```

```
00097 void GuiStack<T>::push(const T& elem) {
00098     Base::push(GuiNode<T>{elem});
00099 }
00100
00101 template<typename T>
00102 void GuiStack<T>::pop() {
00103     Base::pop();
00104 }
00105
00106 template<typename T>
00107 void GuiStack<T>::render_link(Vector2 src, Vector2 dest) {
00108     constexpr int radius = GuiNode<T>::radius;
00109     constexpr float scaled_len = radius / 8.0F;
00110
00111     // straight line
00112     Vector2 link_pos{src.x - scaled_len, src.y + radius};
00113     Vector2 link_size{2 * scaled_len, dest.y - src.y - 2 * radius};
00114
00115     // arrow
00116     constexpr int arrow_size = scaled_len * 5;
00117     Vector2 head{src.x, dest.y - radius + scaled_len / 2};
00118     Vector2 side_left{head.x - arrow_size, head.y - arrow_size};
00119     Vector2 side_right{head.x + arrow_size, head.y - arrow_size};
00120
00121     // draw both
00122     DrawRectangleV(link_pos, link_size, Settings::get_instance().get_color(1));
00123     DrawTriangle(head, side_right, side_left,
00124                  Settings::get_instance().get_color(1));
00125 }
00126
00127 template<typename T>
00128 void GuiStack<T>::render() {
00129     update();
00130
00131     for (auto* ptr = m_head; ptr != nullptr; ptr = ptr->next) {
00132         if (ptr->next != nullptr) {
00133             render_link(ptr->data.get_pos(), ptr->next->data.get_pos());
00134         }
00135
00136         ptr->data.render();
00137     }
00138 }
00139
00140 template<typename T>
00141 void GuiStack<T>::update() {
00142     // TODO: if not outdated then return
00143
00144     std::size_t pos = 0;
00145
00146     for (auto* ptr = m_head; ptr != nullptr; ptr = ptr->next) {
00147         ptr->data.set_pos(
00148             {head_pos.x, head_pos.y + 4 * GuiNode<T>::radius * pos});
00149         ++pos;
00150     }
00151 }
00152
00153 }  // namespace gui
00154
00155 #endif  // GUI_STACK_GUI_HPP_
```

7.67 src/main.cpp File Reference

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

Include dependency graph for main.cpp:

Functions

- int main ()

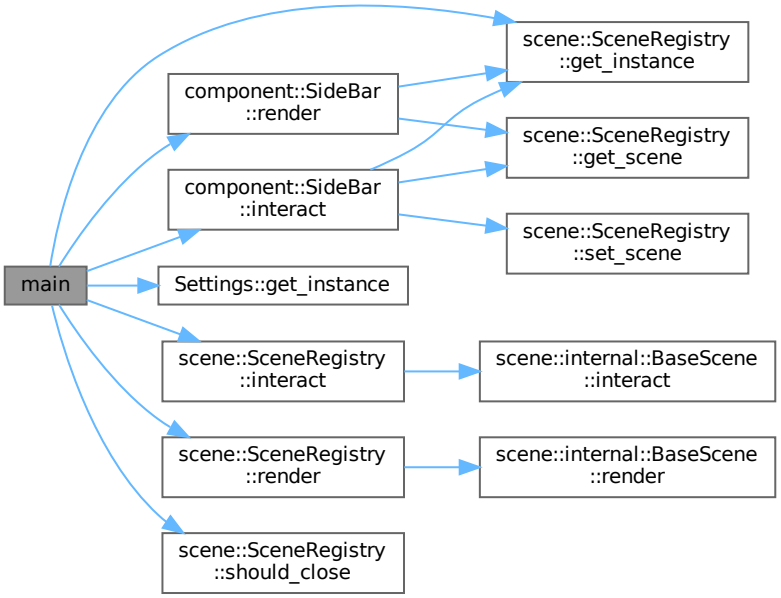
7.67.1 Function Documentation

7.67.1.1 main()

```
int main ( )
```

Definition at line 9 of file main.cpp.

Here is the call graph for this function:



7.68 main.cpp

Go to the documentation of this file.

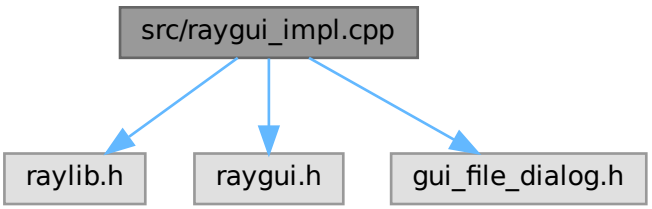
```
00001 #include <iostream>
00002
00003 #include "component/sidebar.hpp"
00004 #include "constants.hpp"
00005 #include "raygui.h"
00006 #include "scene/scene_registry.hpp"
00007 #include "settings.hpp"
00008
00009 int main() {
00010     InitWindow(constants::scene_width, constants::scene_height,
00011                "VisuAlgo.net clone in C++ by @jalsol");
00012     SetTargetFPS(constants::frames_per_second);
```

```
00013
00014     GuiLoadStyle("data/bluish_open_sans.rgs");
00015
00016     scene::SceneRegistry& registry = scene::SceneRegistry::get_instance();
00017     component::SideBar sidebar;
00018
00019     bool should_close = false;
00020
00021     do {
00022         // NOTE: The order is important
00023         sidebar.interact();
00024         registry.interact();
00025
00026         BeginDrawing();
00027         {
00028             ClearBackground(
00029                 Settings::get_instance().get_color(Settings::num_color - 1));
00030
00031             // NOTE: The order is important
00032             registry.render();
00033             sidebar.render();
00034         }
00035         EndDrawing();
00036
00037         should_close = registry.should_close() || WindowShouldClose();
00038     } while (!should_close);
00039
00040     CloseWindow();
00041
00042     return 0;
00043 }
```

7.69 src/raygui_impl.cpp File Reference

```
#include "raylib.h"
#include "raygui.h"
#include "gui_file_dialog.h"
```

Include dependency graph for raygui_impl.cpp:



Macros

- #define RAYGUI_IMPLEMENTATION
- #define GUI_FILE_DIALOG_IMPLEMENTATION

7.69.1 Macro Definition Documentation

7.69.1.1 GUI_FILE_DIALOG_IMPLEMENTATION

```
#define GUI_FILE_DIALOG_IMPLEMENTATION
```

Definition at line 6 of file raygui_impl.cpp.

7.69.1.2 RAYGUI_IMPLEMENTATION

```
#define RAYGUI_IMPLEMENTATION
```

Definition at line 2 of file raygui_impl.cpp.

7.70 raygui_impl.cpp

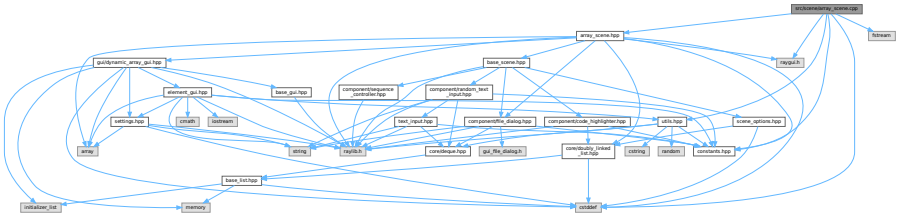
Go to the documentation of this file.

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

7.71 src/scene/array_scene.cpp File Reference

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

Include dependency graph for array_scene.cpp:



Namespaces

- namespace scene

7.72 array_scene.cpp

Go to the documentation of this file.

```
00001 #include "array_scene.hpp"
00002
00003 #include <cstddef>
00004 // #include <cstdlib>
00005 // #include <cstring>
00006 #include <fstream>
00007 // #include <iostream>
00008 // #include <limits>
00009 // #include <string>
00010
00011 #include "constants.hpp"
00012 #include "raygui.h"
00013 #include "utils.hpp"
00014
00015 namespace scene {
00016
00017 ArrayScene::ArrayScene() { m_array.reserve(scene_options.max_size); }
00018
00019 void ArrayScene::render_inputs() {
00020     int& mode = scene_options.mode_selection;
00021
00022     switch (mode) {
00023         case 0: {
00024             switch (scene_options.action_selection.at(mode)) {
00025                 case 0:
00026                     break;
00027                 case 1: {
00028                     m_text_input.render_head(options_head, head_offset);
00029                 } break;
00030                 case 2: {
00031                     m_go = (m_file_dialog.render_head(options_head,
00032                                                       head_offset) > 0);
00033                     return;
00034                 } break;
00035                 default:
00036                     utils::unreachable();
00037             }
00038         } break;
00039
00040         case 1: {
00041             m_index_input.render_head(options_head, head_offset);
00042         } break;
00043
00044         case 2: {
00045             m_index_input.render_head(options_head, head_offset);
00046             m_text_input.render_head(options_head, head_offset);
00047         } break;
00048
00049         case 3: {
00050             m_text_input.render_head(options_head, head_offset);
00051         } break;
00052
00053         case 4: {
00054             m_index_input.render_head(options_head, head_offset);
00055             m_text_input.render_head(options_head, head_offset);
00056         } break;
00057
00058         case 5: {
00059             m_index_input.render_head(options_head, head_offset);
00060         } break;
00061
00062         default:
00063             utils::unreachable();
00064     }
00065
00066     m_go |= render_go_button();
00067 }
00068
00069 void ArrayScene::render() {
00070     m_sequence_controller.inc_anim_counter();
00071
00072     int frame_idx = m_sequence_controller.get_anim_frame();
00073     auto* const frame_ptr = m_sequence.find(frame_idx);
00074     m_sequence_controller.set_progress_value(frame_idx);
00075
00076     if (frame_ptr != nullptr) {
00077         frame_ptr->data.render();
00078         m_code_highlighter.highlight(frame_idx);
00079     } else {  // end of sequence
00080         m_array.render();
00081         m_sequence_controller.set_run_all(false);
00082     }
```

```
00083
00084     m_code_highlighter.render();
00085     m_sequence_controller.render();
00086     render_options(scene_options);
00087 }
00088
00089 void ArrayScene::interact() {
00090     if (m_sequence_controller.interact()) {
00091         m_sequence_controller.reset_anim_counter();
00092         return;
00093     }
00094
00095     m_index_input.set_random_max((int)m_array.size() - 1);
00096
00097     if (m_text_input.interact() || m_index_input.interact()) {
00098         return;
00099     }
00100
00101     if (!m_go) {
00102         return;
00103     }
00104
00105     int& mode = scene_options.mode_selection;
00106
00107     switch (mode) {
00108         case 0: {
00109             switch (scene_options.action_selection.at(mode)) {
00110                 case 0: {
00111                     interact_random();
00112                 } break;
00113
00114                 case 1: {
00115                     interact_import(m_text_input.extract_values());
00116                 } break;
00117
00118                 case 2: {
00119                     interact_file_import();
00120                 } break;
00121
00122                 default:
00123                     utils::unreachable();
00124             }
00125
00126             m_code_highlighter.set_code({});
00127             m_sequence.clear();
00128             m_sequence_controller.set_max_value(0);
00129         } break;
00130
00131         case 1: {
00132             interact_access();
00133         } break;
00134
00135         case 2: {
00136             interact_update();
00137         } break;
00138
00139         case 3: {
00140             interact_search();
00141         } break;
00142
00143         case 4: {
00144             m_index_input.set_random_max((int)m_array.size());
00145             interact_insert();
00146         } break;
00147
00148         case 5: {
00149             interact_delete();
00150         } break;
00151
00152         default:
00153             utils::unreachable();
00154     }
00155
00156     m_go = false;
00157 }
00158
00159 void ArrayScene::interact_access() {
00160     auto index_container = m_index_input.extract_values();
00161     if (index_container.empty()) {
00162         return;
00163     }
00164
00165     std::size_t index = index_container.front();
00166     if (index >= m_array.size()) {
00167         return;
00168     }
00169
```

```
00170     m_code_highlighter.set_code({"return m_array[index];"});
00171
00172     m_sequence.clear();
00173
00174     m_array.set_color_index(index, 3);
00175     m_sequence.insert(m_sequence.size(), m_array);
00176     m_code_highlighter.push_into_sequence(0);
00177
00178     m_array.set_color_index(index, 0);
00179
00180     m_sequence_controller.set_max_value((int)m_sequence.size());
00181     m_sequence_controller.set_rerun();
00182 }
00183
00184 void ArrayScene::interact_random() {
00185     std::size_t size =
00186         utils::get_random(std::size_t{1}, scene_options.max_size);
00187     m_array = {};
00188
00189     for (std::size_t i = 0; i < size; ++i) {
00190         m_array.push(utils::get_random(constants::min_val, constants::max_val));
00191     }
00192
00193     m_array.reserve(max_size);
00194 }
00195
00196 void ArrayScene::interact_import(core::Deque<int> nums) {
00197     m_array = {};
00198     std::size_t i;  // NOLINT
00199
00200     for (i = 0; i < max_size && !nums.empty(); ++i) {
00201         m_array.push(nums.front());
00202         nums.pop_front();
00203     }
00204
00205     m_array.reserve(max_size);
00206 }
00207
00208 void ArrayScene::interact_update() {
00209     auto index_container = m_index_input.extract_values();
00210     if (index_container.empty()) {
00211         return;
00212     }
00213
00214     auto value_container = m_text_input.extract_values();
00215     if (value_container.empty()) {
00216         return;
00217     }
00218
00219     int index = index_container.front();
00220     int value = value_container.front();
00221
00222     if (!(0 <= index && index < m_array.size()) ||
00223         !utils::val_in_range(value)) {
00224         return;
00225     }
00226
00227     m_code_highlighter.set_code({
00228         "array[index] = value;",
00229     });
00230
00231     m_sequence.clear();
00232
00233     // initial state (before update)
00234     m_sequence.insert(m_sequence.size(), m_array);
00235     m_code_highlighter.push_into_sequence(-1);
00236
00237     // highlight
00238     m_array.set_color_index(index, 2);
00239     m_sequence.insert(m_sequence.size(), m_array);
00240     m_code_highlighter.push_into_sequence(0);
00241
00242     // update
00243     m_array[index] = value;
00244     m_array.set_color_index(index, 3);
00245     m_sequence.insert(m_sequence.size(), m_array);
00246     m_code_highlighter.push_into_sequence(0);
00247
00248     // undo highlight
00249     m_array.set_color_index(index, 0);
00250
00251     m_sequence_controller.set_max_value((int)m_sequence.size());
00252     m_sequence_controller.set_rerun();
00253 }
00254
00255 void ArrayScene::interact_file_import() {
00256     interact_import(m_file_dialog.extract_values());
```

```
00257 }
00258
00259 void ArrayScene::interact_search() {
00260     auto value_container = m_text_input.extract_values();
00261     if (value_container.empty()) {
00262         return;
00263     }
00264
00265     int value = value_container.front();
00266     if (!utils::val_in_range(value)) {
00267         return;
00268     }
00269
00270     m_code_highlighter.set_code({
00271         "for (i = 0; i < size; i++)",
00272         "    if (array[i] == value)",
00273         "        return i;",
00274         "return not_found",
00275     });
00276
00277     m_sequence.clear();
00278     m_sequence.insert(m_sequence.size(), m_array);
00279     m_code_highlighter.push_into_sequence(0);
00280
00281     bool found = false;
00282
00283     for (std::size_t i = 0; i < m_array.size(); ++i) {
00284         m_array.set_color_index(i, 3);
00285         m_sequence.insert(m_sequence.size(), m_array);
00286         m_code_highlighter.push_into_sequence(1);
00287
00288         if (m_array[i] == value) {
00289             found = true;
00290             m_array.set_color_index(i, 4);
00291             m_sequence.insert(m_sequence.size(), m_array);
00292             m_code_highlighter.push_into_sequence(2);
00293             m_array.set_color_index(i, 0);
00294             break;
00295         }
00296
00297         m_array.set_color_index(i, 0);
00298         m_sequence.insert(m_sequence.size(), m_array);
00299         m_code_highlighter.push_into_sequence(0);
00300     }
00301
00302     if (!found) {
00303         m_sequence.insert(m_sequence.size(), m_array);
00304         m_code_highlighter.push_into_sequence(3);
00305     }
00306
00307     m_sequence_controller.set_max_value((int)m_sequence.size());
00308     m_sequence_controller.set_rerun();
00309 }
00310
00311 void ArrayScene::interact_insert() {
00312     auto index_container = m_index_input.extract_values();
00313     if (index_container.empty()) {
00314         return;
00315     }
00316
00317     auto value_container = m_text_input.extract_values();
00318     if (value_container.empty()) {
00319         return;
00320     }
00321
00322     int index = index_container.front();
00323     int value = value_container.front();
00324
00325     if (m_array.size() >= max_size) {
00326         return;
00327     }
00328
00329     if (!(0 <= index && index <= m_array.size()) ||
00330         !utils::val_in_range(value)) {
00331         return;
00332     }
00333
00334     m_code_highlighter.set_code({
00335         "size++;",
00336         "for (i = size - 1; i > index; i--)",
00337         "    array[i] = array[i - 1];",
00338         "array[index] = value;",
00339     });
00340
00341     m_sequence.clear();
00342     m_sequence.insert(m_sequence.size(), m_array);
00343     m_code_highlighter.push_into_sequence(-1);
```

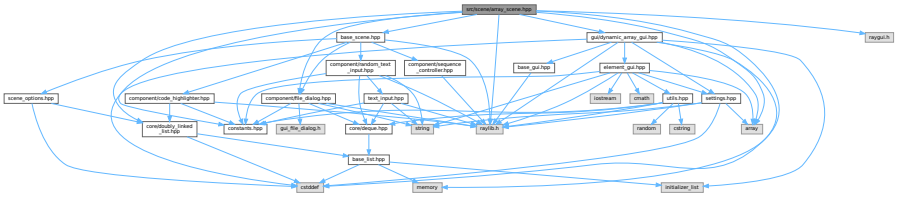
```
00344
00345     m_array.push(0);
00346     m_sequence.insert(m_sequence.size(), m_array);
00347     m_code_highlighter.push_into_sequence(0);
00348
00349     for (std::size_t i = m_array.size() - 1; i > index; --i) {
00350         m_array.set_color_index(i - 1, 2);
00351         m_sequence.insert(m_sequence.size(), m_array);
00352         m_code_highlighter.push_into_sequence(1);
00353
00354         m_array[i] = m_array[i - 1];
00355         m_array.set_color_index(i, 3);
00356         m_sequence.insert(m_sequence.size(), m_array);
00357         m_code_highlighter.push_into_sequence(2);
00358
00359         m_array.set_color_index(i - 1, 0);
00360         m_array.set_color_index(i, 0);
00361         m_sequence.insert(m_sequence.size(), m_array);
00362         m_code_highlighter.push_into_sequence(1);
00363     }
00364
00365     m_array.set_color_index(index, 2);
00366     m_sequence.insert(m_sequence.size(), m_array);
00367     m_code_highlighter.push_into_sequence(3);
00368
00369     m_array[index] = value;
00370     m_array.set_color_index(index, 3);
00371     m_sequence.insert(m_sequence.size(), m_array);
00372     m_code_highlighter.push_into_sequence(3);
00373
00374     m_array.set_color_index(index, 0);
00375     m_sequence.insert(m_sequence.size(), m_array);
00376     m_code_highlighter.push_into_sequence(-1);
00377
00378     m_sequence_controller.set_max_value((int)m_sequence.size());
00379     m_sequence_controller.set_rerun();
00380 }
00381
00382 void ArrayScene::interact_delete() {
00383     auto index_container = m_index_input.extract_values();
00384     if (index_container.empty()) {
00385         return;
00386     }
00387
00388     int index = index_container.front();
00389
00390     if (m_array.size() == 0) {
00391         return;
00392     }
00393
00394     if (!(0 <= index && index < m_array.size())) {
00395         return;
00396     }
00397
00398     m_code_highlighter.set_code({
00399         "for (i = index; i < size - 1; i++)",
00400         "    array[i] = array[i + 1];",
00401         "size--;",
00402     });
00403
00404     m_sequence.clear();
00405     m_sequence.insert(m_sequence.size(), m_array);
00406     m_code_highlighter.push_into_sequence(-1);
00407
00408     m_sequence.insert(m_sequence.size(), m_array);
00409     m_code_highlighter.push_into_sequence(0);
00410
00411     for (std::size_t i = index; i < m_array.size() - 1; ++i) {
00412         m_array.set_color_index(i, 3);
00413         m_sequence.insert(m_sequence.size(), m_array);
00414         m_code_highlighter.push_into_sequence(1);
00415
00416         m_array[i] = m_array[i + 1];
00417         m_array.set_color_index(i + 1, 2);
00418         m_sequence.insert(m_sequence.size(), m_array);
00419         m_code_highlighter.push_into_sequence(1);
00420
00421         m_array.set_color_index(i, 0);
00422         m_array.set_color_index(i + 1, 0);
00423         m_sequence.insert(m_sequence.size(), m_array);
00424         m_code_highlighter.push_into_sequence(0);
00425     }
00426
00427     m_array.set_color_index(m_array.size() - 1, 2);
00428     m_sequence.insert(m_sequence.size(), m_array);
00429     m_code_highlighter.push_into_sequence(2);
00430
```

```
00431     m_array.pop();
00432     m_sequence.insert(m_sequence.size(), m_array);
00433     m_code_highlighter.push_into_sequence(-1);
00434 
00435     m_sequence_controller.set_max_value((int)m_sequence.size());
00436     m_sequence_controller.set_rerun();
00437 }
00438 
00439 }  // namespace scene
```

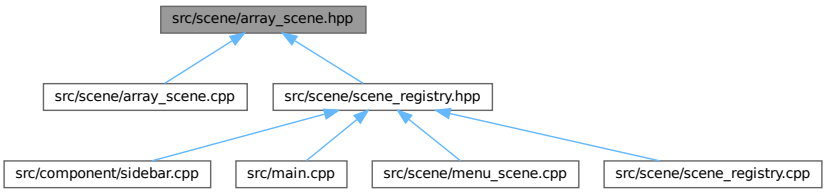
7.73 src/scene/array_scene.hpp File Reference

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

Include dependency graph for array_scene.hpp:



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



Classes

- class scene::ArrayScene

  The array scene.

Namespaces

- namespace scene

7.74 array_scene.hpp

Go to the documentation of this file.

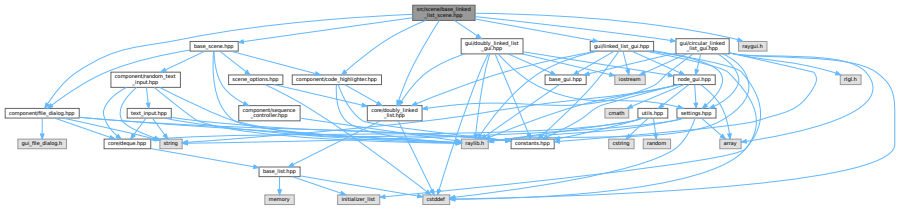
```
00001 #ifndef SCENE_ARRAY_SCENE_HPP_
00002 #define SCENE_ARRAY_SCENE_HPP_
00003
00004 #include <array>
00005 #include <cstddef>
00006
00007 #include "base_scene.hpp"
00008 #include "component/file_dialog.hpp"
00009 #include "constants.hpp"
00010 #include "core/doubly_linked_list.hpp"
00011 #include "gui/dynamic_array_gui.hpp"
00012 #include "raygui.h"
00013 #include "raylib.h"
00014
00015 namespace scene {
00016
00021 class ArrayScene : public internal::BaseScene {
00022 public:
00026     ArrayScene();
00027
00031     void render() override;
00032
00036     void interact() override;
00037
00038 private:
00042     static constexpr std::size_t max_size = 8;
00043
00047     internal::SceneOptions scene_options{
00048         // max_size
00049         max_size,
00050
00051         // mode_labels
00052         "Mode: Create;"
00053         "Mode: Access;"
00054         "Mode: Update;"
00055         "Mode: Search;"
00056         "Mode: Insert;"
00057         "Mode: Delete",
00058
00059         // mode_selection
00060         0,
00061
00062         // action_labels
00063         {
00064             // Mode: Create
00065             "Action: Random;Action: Input;Action: File",
00066
00067             // Mode: Access
00068             "",
00069
00070             // Mode: Update
00071             "",
00072
00073             // Mode: Search
00074             "",
00075
00076             // Mode: Insert
00077             "",
00078
00079             // Mode: Delete
00080             "",
00081         },
00082
00083         // action_selection
00084         core::DoublyLinkedList<int>{0, 0, 0, 0, 0},
00085     };
00086
00087     using internal::BaseScene::button_size;
00088     using internal::BaseScene::head_offset;
00089     using internal::BaseScene::options_head;
00090
00094     gui::GuiDynamicArray<int> m_array{};
00095
00099     core::DoublyLinkedList<gui::GuiDynamicArray<int>> m_sequence;
00100
00104     bool m_go{};
00105
00106     using internal::BaseScene::m_file_dialog;
00107     using internal::BaseScene::m_index_input;
00108     using internal::BaseScene::m_sequence_controller;
00109     using internal::BaseScene::m_text_input;
00110
```

```
00111     using internal::BaseScene::render_go_button;
00112     using internal::BaseScene::render_options;
00113
00117     void render_inputs() override;
00118
00122     void interact_random();
00123
00127     void interact_import(core::Deque<int> nums);
00128
00132     void interact_file_import();
00133
00137     void interact_update();
00138
00142     void interact_search();
00143
00147     void interact_insert();
00148
00152     void interact_delete();
00153
00157     void interact_access();
00158 };
00159
00160 }  // namespace scene
00161
00162 #endif  // SCENE_ARRAY_SCENE_HPP_
```

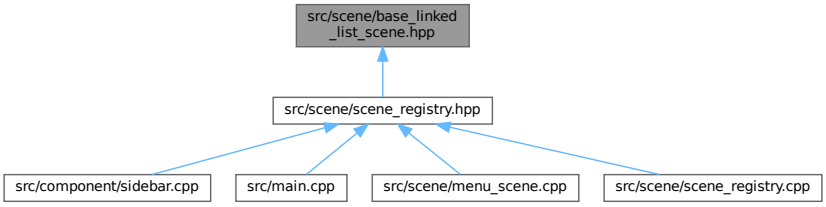
7.75 src/scene/base_linked_list_scene.hpp File Reference

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

Include dependency graph for base_linked_list_scene.hpp:



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



Classes

- class scene::BaseLinkedListScene< Con >

  The base linked list scene.

Namespaces

- namespace scene

Typedefs

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

7.76 base_linked_list_scene.hpp

Go to the documentation of this file.

```
00001 #ifndef SCENE_BASE_LINKED_LIST_SCENE_HPP_
00002 #define SCENE_BASE_LINKED_LIST_SCENE_HPP_
00003
00004 #include "base_scene.hpp"
00005 #include "component/code_highlighter.hpp"
00006 #include "component/file_dialog.hpp"
00007 #include "core/doubly_linked_list.hpp"
00008 #include "gui/circular_linked_list_gui.hpp"
00009 #include "gui/doubly_linked_list_gui.hpp"
00010 #include "gui/linked_list_gui.hpp"
00011 #include "raygui.h"
00012
00013 namespace scene {
00014
00020 template<typename Con>
00021 class BaseLinkedListScene : public internal::BaseScene {
00022 public:
00026     void render() override;
00027
00031     void interact() override;
00032
00033 private:
00037     internal::SceneOptions scene_options{
00038         // max_size
00039         8,  // NOLINT
00040
00041         // mode_labels
00042         "Mode: Create;"
00043         "Mode: Add;"
00044         "Mode: Delete;"
00045         "Mode: Update;"
00046         "Mode: Search",
00047
00048         // mode_selection
00049         0,
00050
00051         // action_labels
00052         {
00053             // Mode: Create
00054             "Action: Random;Action: Input;Action: File",
00055             // Mode: Add
00056             "",
00057             // Mode: Delete
00058             "",
00059             // Mode: Update
00060             "",
00061             // Mode: Search
00062             "",
00063         },
00064
00065         // action_selection
00066         core::DoublyLinkedList<int>{0, 0, 0, 0, 0},
```

```
00067     };
00068
00069     using internal::BaseScene::button_size;
00070     using internal::BaseScene::head_offset;
00071     using internal::BaseScene::options_head;
00072
00076     Con m_list{
00077         gui::GuiNode<int>{1},
00078         gui::GuiNode<int>{2},
00079         gui::GuiNode<int>{3},
00080     };
00081
00085     core::DoublyLinkedList<Con> m_sequence;
00086
00090     bool m_go{};
00091
00092     using internal::BaseScene::m_code_highlighter;
00093     using internal::BaseScene::m_file_dialog;
00094     using internal::BaseScene::m_index_input;
00095     using internal::BaseScene::m_sequence_controller;
00096     using internal::BaseScene::m_text_input;
00097
00098     using internal::BaseScene::render_go_button;
00099     using internal::BaseScene::render_options;
00100
00104     void render_inputs() override;
00105
00109     void interact_random();
00110
00114     void interact_import(core::Deque<int> nums);
00115
00119     void interact_file_import();
00120
00124     void interact_add();
00125
00129     void interact_add_head(int value);
00130
00134     void interact_add_tail(int value);
00135
00139     void interact_add_middle(int index, int value);
00140
00144     void interact_delete();
00145
00149     void interact_delete_head();
00150
00154     void interact_delete_tail();
00155
00159     void interact_delete_middle(int index);
00160
00164     void interact_update();
00165
00169     void interact_search();
00170 };
00171
00172 using LinkedListScene = BaseLinkedListScene<gui::GuiLinkedList<int>>;
00173 using DoublyLinkedListScene =
00174     BaseLinkedListScene<gui::GuiDoublyLinkedList<int>>;
00175 using CircularLinkedListScene =
00176     BaseLinkedListScene<gui::GuiCircularLinkedList<int>>;
00177
00178 template<typename Con>
00179 void BaseLinkedListScene<Con>::render_inputs() {
00180     int& mode = scene_options.mode_selection;
00181
00182     switch (mode) {
00183         case 0: {
00184             switch (scene_options.action_selection.at(mode)) {
00185                 case 0:
00186                     break;
00187                 case 1: {
00188                     m_text_input.render_head(options_head, head_offset);
00189                 } break;
00190                 case 2: {
00191                     m_go = (m_file_dialog.render_head(options_head,
00192                                                       head_offset) > 0);
00193                     return;
00194                 } break;
00195                 default:
00196                     utils::unreachable();
00197             }
00198         } break;
00199
00200         case 1: {
00201             m_index_input.render_head(options_head, head_offset);
00202             m_text_input.render_head(options_head, head_offset);
00203         } break;
00204
```

```
00205         case 2: {
00206             m_index_input.render_head(options_head, head_offset);
00207         } break;
00208 
00209         case 3: {
00210             m_index_input.render_head(options_head, head_offset);
00211             m_text_input.render_head(options_head, head_offset);
00212         } break;
00213 
00214         case 4: {
00215             m_text_input.render_head(options_head, head_offset);
00216         } break;
00217 
00218         default:
00219             utils::unreachable();
00220     }
00221 
00222     m_go |= render_go_button();
00223 }
00224 
00225 template<typename Con>
00226 void BaseLinkedListScene<Con>::render() {
00227     m_sequence_controller.inc_anim_counter();
00228 
00229     int frame_idx = m_sequence_controller.get_anim_frame();
00230     auto* const frame_ptr = m_sequence.find(frame_idx);
00231     m_sequence_controller.set_progress_value(frame_idx);
00232 
00233     if (frame_ptr != nullptr) {
00234         frame_ptr->data.render();
00235         m_code_highlighter.highlight(frame_idx);
00236     } else {  // end of sequence
00237         m_list.render();
00238         m_sequence_controller.set_run_all(false);
00239     }
00240 
00241     m_code_highlighter.render();
00242     m_sequence_controller.render();
00243     render_options(scene_options);
00244 }
00245 
00246 template<typename Con>
00247 void BaseLinkedListScene<Con>::interact() {
00248     if (m_sequence_controller.interact()) {
00249         m_sequence_controller.reset_anim_counter();
00250         return;
00251     }
00252 
00253     m_index_input.set_random_max((int)m_list.size() - 1);
00254 
00255     if (m_text_input.interact() || m_index_input.interact()) {
00256         return;
00257     }
00258 
00259     if (!m_go) {
00260         return;
00261     }
00262 
00263     int& mode = scene_options.mode_selection;
00264 
00265     switch (mode) {
00266         case 0: {
00267             switch (scene_options.action_selection.at(mode)) {
00268                 case 0: {
00269                     interact_random();
00270                 } break;
00271 
00272                 case 1: {
00273                     interact_import(m_text_input.extract_values());
00274                 } break;
00275 
00276                 case 2: {
00277                     interact_file_import();
00278                 } break;
00279 
00280                 default:
00281                     utils::unreachable();
00282             }
00283             m_code_highlighter.set_code({});
00284             m_sequence.clear();
00285             m_sequence_controller.set_max_value(0);
00286         } break;
00287 
00288         case 1: {
00289             m_index_input.set_random_max((int)m_list.size());
00290             interact_add();
00291         } break;
```

```
00292
00293         case 2: {
00294             interact_delete();
00295         } break;
00296
00297         case 3: {
00298             interact_update();
00299         } break;
00300
00301         case 4: {
00302             interact_search();
00303         } break;
00304
00305         default:
00306             utils::unreachable();
00307     }
00308
00309     m_go = false;
00310 }
00311
00312 template<typename Con>
00313 void BaseLinkedListScene<Con>::interact_random() {
00314     std::size_t size =
00315         utils::get_random(std::size_t{1}, scene_options.max_size);
00316     m_list = Con();
00317
00318     for (auto i = 0; i < size; ++i) {
00319         m_list.insert(
00320             i, utils::get_random(constants::min_val, constants::max_val));
00321     }
00322     m_list.init_label();
00323 }
00324
00325 template<typename Con>
00326 void BaseLinkedListScene<Con>::interact_import(core::Deque<int> nums) {
00327     m_sequence.clear();
00328     m_list = Con();
00329
00330     while (!nums.empty()) {
00331         if (utils::val_in_range(nums.front())) {
00332             m_list.insert(m_list.size(), nums.front());
00333         }
00334         nums.pop_front();
00335     }
00336     m_list.init_label();
00337 }
00338
00339 template<typename Con>
00340 void BaseLinkedListScene<Con>::interact_file_import() {
00341     interact_import(m_file_dialog.extract_values());
00342 }
00343
00344 template<typename Con>
00345 void BaseLinkedListScene<Con>::interact_add() {
00346     auto index_container = m_index_input.extract_values();
00347     if (index_container.empty()) {
00348         return;
00349     }
00350
00351     auto value_container = m_text_input.extract_values();
00352     if (value_container.empty()) {
00353         return;
00354     }
00355
00356     int index = index_container.front();
00357     int value = value_container.front();
00358
00359     if (!(0 <= index && index <= m_list.size())) {
00360         return;
00361     }
00362
00363     if (!utils::val_in_range(value)) {
00364         return;
00365     }
00366
00367     m_sequence.clear();
00368     m_sequence.insert(m_sequence.size(), m_list);
00369
00370     if (index == 0) {
00371         interact_add_head(value);
00372     } else if (index == m_list.size()) {
00373         interact_add_tail(value);
00374     } else {
00375         interact_add_middle(index, value);
00376     }
00377
00378     m_sequence_controller.set_max_value((int)m_sequence.size());
```

```
00379     m_sequence_controller.set_rerun();
00380 }
00381
00382 template<typename Con>
00383 void BaseLinkedListScene<Con>::interact_add_head(int value) {
00384     m_code_highlighter.set_code({
00385         "Node* node = new Node(value);",
00386         "node->next = head;",
00387         "head = next;",
00388     });
00389     m_code_highlighter.push_into_sequence(-1);
00390
00391     m_list.insert(0, value);
00392
00393     m_list.at(0).set_color_index(6);
00394     m_list.at(0).set_label("node");
00395     m_sequence.insert(m_sequence.size(), m_list);
00396     m_code_highlighter.push_into_sequence(0);
00397
00398     if (m_list.size() > 1) {
00399         m_list.at(1).set_color_index(4);
00400     }
00401
00402     m_sequence.insert(m_sequence.size(), m_list);
00403     m_code_highlighter.push_into_sequence(1);
00404
00405     if (m_list.size() > 1) {
00406         m_list.at(1).set_color_index(0);
00407         m_list.at(1).set_label("");
00408     }
00409
00410     m_list.at(0).set_color_index(4);
00411     m_list.at(0).set_label("head");
00412     m_sequence.insert(m_sequence.size(), m_list);
00413     m_code_highlighter.push_into_sequence(2);
00414
00415     m_list.at(0).set_color_index(0);
00416 }
00417
00418 template<typename Con>
00419 void BaseLinkedListScene<Con>::interact_add_tail(int value) {
00420     m_code_highlighter.set_code({
00421         "Node* node = new Node(value);",
00422         "tail->next = node;",
00423         "tail = tail->next;",
00424     });
00425     m_code_highlighter.push_into_sequence(-1);
00426
00427     std::size_t size = m_list.size();
00428
00429     m_list.insert(size, value);
00430     m_list.at(size).set_color_index(6);
00431     m_sequence.insert(m_sequence.size(), m_list);
00432     m_code_highlighter.push_into_sequence(0);
00433
00434     m_list.at(size - 1).set_color_index(4);
00435     m_sequence.insert(m_sequence.size(), m_list);
00436     m_code_highlighter.push_into_sequence(1);
00437
00438     m_list.at(size - 1).set_color_index(0);
00439     m_list.at(size - 1).set_label("");
00440     m_list.at(size).set_color_index(4);
00441     m_list.at(size).set_label("tail");
00442     m_sequence.insert(m_sequence.size(), m_list);
00443     m_code_highlighter.push_into_sequence(2);
00444
00445     m_list.at(size).set_color_index(0);
00446 }
00447
00448 template<typename Con>
00449 void BaseLinkedListScene<Con>::interact_add_middle(int index, int value) {
00450     m_code_highlighter.set_code({
00451         "Node* pre = head;",
00452         "for (i = 0; i < index - 1; ++i)",
00453         "    pre = pre->next;",
00454         "",
00455         "Node* nxt = pre->next;",
00456         "Node* node = new Node(value);",
00457         "node->next = nxt;",
00458         "pre->next = node;",
00459     });
00460     m_code_highlighter.push_into_sequence(-1);
00461
00462     m_list.at(0).set_color_index(4);
00463     m_list.at(0).set_label("head/pre");
00464     m_sequence.insert(m_sequence.size(), m_list);
00465     m_code_highlighter.push_into_sequence(0);
```

```
00466
00467     // search until index - 1
00468     for (int i = 0; i < index - 1; ++i) {
00469         m_list.at(i).set_color_index(2);
00470         m_sequence.insert(m_sequence.size(), m_list);
00471         m_code_highlighter.push_into_sequence(1);
00472
00473         m_list.at(i).set_color_index(0);
00474         m_list.at(i).set_label(i == 0 ? "head" : "");
00475         m_list.at(i + 1).set_color_index(2);
00476         m_list.at(i + 1).set_label("pre");
00477         m_sequence.insert(m_sequence.size(), m_list);
00478         m_code_highlighter.push_into_sequence(2);
00479     }
00480
00481     m_sequence.insert(m_sequence.size(), m_list);
00482     m_code_highlighter.push_into_sequence(1);
00483
00484     // reaching index - 1
00485     // cur
00486     m_list.at(index - 1).set_color_index(2);
00487     m_sequence.insert(m_sequence.size(), m_list);
00488     m_code_highlighter.push_into_sequence(3);
00489
00490     // cur->next
00491     m_list.at(index).set_color_index(7);
00492     m_list.at(index).set_label(index + 1 == m_list.size() ? "tail/nxt" : "nxt");
00493     m_sequence.insert(m_sequence.size(), m_list);
00494     m_code_highlighter.push_into_sequence(4);
00495
00496     // insert between cur and cur->next
00497     m_list.insert(index, value);
00498     m_list.at(index).set_color_index(6);
00499     m_list.at(index).set_label("node");
00500     m_sequence.insert(m_sequence.size(), m_list);
00501     m_code_highlighter.push_into_sequence(5);
00502
00503     m_list.at(index - 1).set_color_index(2);
00504     m_list.at(index + 1).set_color_index(0);
00505     m_sequence.insert(m_sequence.size(), m_list);
00506     m_code_highlighter.push_into_sequence(6);
00507
00508     m_list.at(index - 1).set_color_index(0);
00509     m_list.at(index + 1).set_color_index(7);
00510     m_list.init_label();
00511     m_sequence.insert(m_sequence.size(), m_list);
00512     m_code_highlighter.push_into_sequence(7);
00513
00514     // done
00515     m_list.at(index - 1).set_color_index(0);
00516     m_list.at(index - 1).set_label("");
00517     m_list.at(index).set_color_index(0);
00518     m_list.at(index).set_label("");
00519     m_list.at(index + 1).set_color_index(0);
00520     m_list.at(index + 1).set_label("");
00521     m_list.init_label();
00522 }
00523
00524 template<typename Con>
00525 void BaseLinkedListScene<Con>::interact_delete() {
00526     if (m_list.empty()) {
00527         return;
00528     }
00529
00530     auto index_container = m_index_input.extract_values();
00531     if (index_container.empty()) {
00532         return;
00533     }
00534
00535     int index = index_container.front();
00536
00537     if (!(0 <= index && index < m_list.size())) {
00538         return;
00539     }
00540
00541     m_sequence.clear();
00542     m_sequence.insert(m_sequence.size(), m_list);
00543
00544     if (index == 0) {
00545         interact_delete_head();
00546     } else if (index + 1 == m_list.size()) {
00547         interact_delete_tail();
00548     } else {
00549         interact_delete_middle(index);
00550     }
00551
00552     m_sequence_controller.set_max_value((int)m_sequence.size());
```

```
00553     m_sequence_controller.set_rerun();
00554 }
00555
00556 template<typename Con>
00557 void BaseLinkedListScene<Con>::interact_delete_head() {
00558     m_code_highlighter.set_code({
00559         "Node* temp = head;",
00560         "head = head->next;",
00561         "delete temp;",
00562     });
00563     m_code_highlighter.push_into_sequence(-1);
00564
00565     m_list.at(0).set_color_index(4);
00566     m_sequence.insert(m_sequence.size(), m_list);
00567     m_code_highlighter.push_into_sequence(0);
00568
00569     m_list.at(0).set_color_index(5);
00570     m_list.at(0).set_label("");
00571     if (m_list.size() > 1) {
00572         m_list.at(1).set_color_index(4);
00573         m_list.at(1).set_label("head");
00574     }
00575     m_sequence.insert(m_sequence.size(), m_list);
00576     m_code_highlighter.push_into_sequence(1);
00577
00578     m_list.remove(0);
00579     m_sequence.insert(m_sequence.size(), m_list);
00580     m_code_highlighter.push_into_sequence(2);
00581
00582     if (m_list.size() > 0) {
00583         m_list.at(0).set_color_index(0);
00584     }
00585 }
00586
00587 template<typename Con>
00588 void BaseLinkedListScene<Con>::interact_delete_tail() {
00589     m_code_highlighter.set_code({
00590         "Node* pre = head;",
00591         "Node* nxt = pre->next;",
00592         "while (nxt->next != nullptr)",
00593         "    pre = pre->next, nxt = nxt->next;",
00594         "",
00595         "delete nxt;",
00596         "tail = pre;",
00597     });
00598     m_code_highlighter.push_into_sequence(-1);
00599
00600     m_list.at(0).set_color_index(2);
00601     m_list.at(0).set_label("head/pre");
00602     m_sequence.insert(m_sequence.size(), m_list);
00603     m_code_highlighter.push_into_sequence(0);
00604
00605     m_list.at(1).set_color_index(3);
00606     if (m_list.size() == 2) {
00607         m_list.at(1).set_label("tail/nxt");
00608     } else {
00609         m_list.at(1).set_label("nxt");
00610     }
00611     m_sequence.insert(m_sequence.size(), m_list);
00612     m_code_highlighter.push_into_sequence(1);
00613
00614     int idx = 0;
00615     for (; idx + 2 < m_list.size(); ++idx) {
00616         m_sequence.insert(m_sequence.size(), m_list);
00617         m_code_highlighter.push_into_sequence(2);
00618
00619         m_list.at(idx).set_color_index(0);
00620         if (idx == 0) {
00621             m_list.at(idx).set_label("head");
00622         } else {
00623             m_list.at(idx).set_label("");
00624         }
00625
00626         m_list.at(idx + 1).set_color_index(2);
00627         m_list.at(idx + 1).set_label("pre");
00628         m_list.at(idx + 2).set_color_index(3);
00629         if (idx + 3 == m_list.size()) {
00630             m_list.at(idx + 2).set_label("tail/nxt");
00631         } else {
00632             m_list.at(idx + 2).set_label("nxt");
00633         }
00634
00635         m_sequence.insert(m_sequence.size(), m_list);
00636         m_code_highlighter.push_into_sequence(3);
00637     }
00638
00639     m_sequence.insert(m_sequence.size(), m_list);
```

```
00640     m_code_highlighter.push_into_sequence(2);
00641
00642     m_list.at(idx).set_color_index(2);
00643     m_list.at(idx).set_label("pre");
00644     m_list.at(idx + 1).set_color_index(3);
00645     m_list.at(idx + 1).set_label("tail/nxt");
00646     m_sequence.insert(m_sequence.size(), m_list);
00647     m_code_highlighter.push_into_sequence(4);
00648
00649     m_list.remove(idx + 1);
00650     m_list.at(idx).set_label("tail/pre");
00651     m_sequence.insert(m_sequence.size(), m_list);
00652     m_code_highlighter.push_into_sequence(5);
00653
00654     m_list.at(idx).set_color_index(4);
00655     m_list.init_label();
00656     m_sequence.insert(m_sequence.size(), m_list);
00657     m_code_highlighter.push_into_sequence(6);
00658
00659     m_list.at(idx).set_color_index(0);
00660 }
00661
00662 template<typename Con>
00663 void BaseLinkedListScene<Con>::interact_delete_middle(int index) {
00664     m_code_highlighter.set_code({
00665         "Node* pre = head;",
00666         "for (i = 0; i < index - 1; i++)",
00667         "    pre = pre->next;",
00668         "",
00669         "Node* node = pre->next;",
00670         "Node* nxt = node->next;",
00671         "delete node;",
00672         "pre->next = nxt;",
00673     });
00674     m_code_highlighter.push_into_sequence(-1);
00675
00676     m_list.at(0).set_color_index(4);
00677     m_list.at(0).set_label("head/pre");
00678     m_sequence.insert(m_sequence.size(), m_list);
00679     m_code_highlighter.push_into_sequence(0);
00680
00681     int idx = 0;
00682     for (; idx + 1 < index; ++idx) {
00683         m_list.at(idx).set_color_index(2);
00684         m_sequence.insert(m_sequence.size(), m_list);
00685         m_code_highlighter.push_into_sequence(1);
00686
00687         m_list.at(idx).set_color_index(0);
00688         m_list.at(idx).set_label("");
00689         m_list.at(idx + 1).set_color_index(2);
00690         m_list.init_label();
00691         m_list.at(idx + 1).set_label("pre");
00692         m_sequence.insert(m_sequence.size(), m_list);
00693         m_code_highlighter.push_into_sequence(2);
00694     }
00695
00696     m_list.at(idx).set_color_index(2);
00697     m_list.at(idx).set_label("pre");
00698     m_sequence.insert(m_sequence.size(), m_list);
00699     m_code_highlighter.push_into_sequence(3);
00700
00701     m_list.at(idx + 1).set_color_index(5);
00702     m_list.at(idx + 1).set_label("node");
00703     m_sequence.insert(m_sequence.size(), m_list);
00704     m_code_highlighter.push_into_sequence(4);
00705
00706     m_list.at(idx + 2).set_color_index(3);
00707     if (idx + 3 == m_list.size()) {
00708         m_list.at(idx + 2).set_label("tail/nxt");
00709     } else {
00710         m_list.at(idx + 2).set_label("nxt");
00711     }
00712     m_sequence.insert(m_sequence.size(), m_list);
00713     m_code_highlighter.push_into_sequence(5);
00714
00715     m_list.remove(idx + 1);
00716     m_sequence.insert(m_sequence.size(), m_list);
00717     m_code_highlighter.push_into_sequence(6);
00718
00719     m_list.at(idx + 1).set_color_index(7);
00720     m_sequence.insert(m_sequence.size(), m_list);
00721     m_code_highlighter.push_into_sequence(7);
00722
00723     m_list.at(idx).set_color_index(0);
00724     m_list.at(idx).set_label("");
00725     m_list.at(idx + 1).set_color_index(0);
00726     m_list.at(idx + 1).set_label("");
```

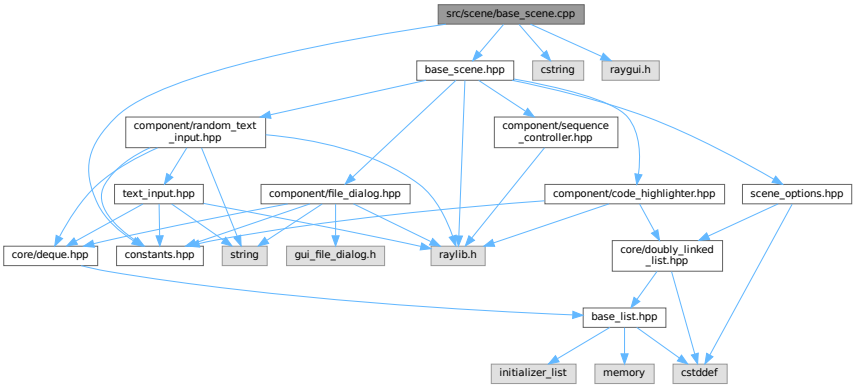
```
00727 }
00728
00729 template<typename Con>
00730 void BaseLinkedListScene<Con>::interact_update() {
00731     auto index_container = m_index_input.extract_values();
00732     if (index_container.empty()) {
00733         return;
00734     }
00735
00736     auto value_container = m_text_input.extract_values();
00737     if (value_container.empty()) {
00738         return;
00739     }
00740
00741     int index = index_container.front();
00742     int value = value_container.front();
00743
00744     if (!(0 <= index && index < m_list.size())) {
00745         return;
00746     }
00747
00748     m_code_highlighter.set_code({
00749         "Node* node = head;",
00750         "for (i = 0; i < index; i++)",
00751         "    node = node->next;",
00752         "",
00753         "node->value = value;",
00754     });
00755
00756     m_sequence.clear();
00757     m_sequence.insert(m_sequence.size(), m_list);
00758     m_code_highlighter.push_into_sequence(-1);
00759
00760     m_list.at(0).set_color_index(4);
00761     m_list.at(0).set_label("head/node");
00762     m_sequence.insert(m_sequence.size(), m_list);
00763     m_code_highlighter.push_into_sequence(0);
00764
00765     for (int i = 0; i < index; ++i) {
00766         m_list.at(i).set_color_index(2);
00767         m_sequence.insert(m_sequence.size(), m_list);
00768         m_code_highlighter.push_into_sequence(1);
00769
00770         m_list.at(i).set_color_index(0);
00771         m_list.at(i).set_label(i == 0 ? "head" : "");
00772         m_list.at(i + 1).set_color_index(2);
00773         m_list.at(i + 1).set_label(i + 2 == m_list.size() ? "tail/node"
00774                                                           : "node");
00775         m_sequence.insert(m_sequence.size(), m_list);
00776         m_code_highlighter.push_into_sequence(2);
00777     }
00778
00779     m_sequence.insert(m_sequence.size(), m_list);
00780     m_code_highlighter.push_into_sequence(1);
00781     m_sequence.insert(m_sequence.size(), m_list);
00782     m_code_highlighter.push_into_sequence(3);
00783
00784     m_list.at(index).set_color_index(3);
00785     m_list.at(index).set_value(value);
00786     m_sequence.insert(m_sequence.size(), m_list);
00787     m_code_highlighter.push_into_sequence(4);
00788
00789     m_list.at(index).set_color_index(0);
00790     m_list.at(index).set_label("");
00791     m_list.init_label();
00792
00793     m_sequence_controller.set_max_value((int)m_sequence.size());
00794     m_sequence_controller.set_rerun();
00795 }
00796
00797 template<typename Con>
00798 void BaseLinkedListScene<Con>::interact_search() {
00799     auto value_container = m_text_input.extract_values();
00800     if (value_container.empty()) {
00801         return;
00802     }
00803
00804     int value = value_container.front();
00805     if (!utils::val_in_range(value)) {
00806         return;
00807     }
00808
00809     m_code_highlighter.set_code({
00810         "Node* node = head;",
00811         "while (node != nullptr) {",
00812         "    if (node->value == value)",
00813         "        return node;",
```

```
00814         "    node = node->next;",
00815         "}",
00816         "return not_found",
00817     });
00818
00819     m_sequence.clear();
00820     m_sequence.insert(m_sequence.size(), m_list);
00821     m_code_highlighter.push_into_sequence(-1);
00822
00823     m_list.at(0).set_color_index(4);
00824     m_list.at(0).set_label("head/node");
00825     m_sequence.insert(m_sequence.size(), m_list);
00826     m_code_highlighter.push_into_sequence(0);
00827
00828     std::size_t idx = 0;
00829
00830     while (idx < m_list.size()) {
00831         m_list.at(idx).set_color_index(2);
00832         m_sequence.insert(m_sequence.size(), m_list);
00833         m_code_highlighter.push_into_sequence(1);
00834
00835         m_sequence.insert(m_sequence.size(), m_list);
00836         m_code_highlighter.push_into_sequence(2);
00837         if (m_list.at(idx).get_value() == value) {
00838             m_list.at(idx).set_color_index(3);
00839             m_sequence.insert(m_sequence.size(), m_list);
00840             m_code_highlighter.push_into_sequence(3);
00841             m_list.at(idx).set_color_index(0);
00842             m_list.at(idx).set_label(idx + 1 == m_list.size() ? "tail" : "");
00843             break;
00844         }
00845
00846         m_list.at(idx).set_color_index(0);
00847         m_list.at(idx).set_label("");
00848         m_list.init_label();
00849         ++idx;
00850         if (idx < m_list.size()) {
00851             m_list.at(idx).set_color_index(2);
00852             m_list.at(idx).set_label(idx + 1 == m_list.size() ? "tail/node"
00853                                                               : "node");
00854         }
00855         m_sequence.insert(m_sequence.size(), m_list);
00856         m_code_highlighter.push_into_sequence(4);
00857     }
00858
00859     if (idx >= m_list.size()) {
00860         m_sequence.insert(m_sequence.size(), m_list);
00861         m_code_highlighter.push_into_sequence(1);
00862
00863         m_sequence.insert(m_sequence.size(), m_list);
00864         m_code_highlighter.push_into_sequence(5);
00865
00866         m_sequence.insert(m_sequence.size(), m_list);
00867         m_code_highlighter.push_into_sequence(6);
00868     }
00869
00870     m_sequence_controller.set_max_value((int)m_sequence.size());
00871     m_sequence_controller.set_rerun();
00872 }
00873
00874 }  // namespace scene
00875
00876 #endif  // SCENE_BASE_LINKED_LIST_SCENE_HPP_
```

7.77 src/scene/base_scene.cpp File Reference

```
#include "base_scene.hpp"
#include <cstring>
#include "constants.hpp"
#include "raygui.h"
```

Include dependency graph for base_scene.cpp:



Namespaces

- namespace scene
- namespace scene::internal

7.78 base_scene.cpp

Go to the documentation of this file.

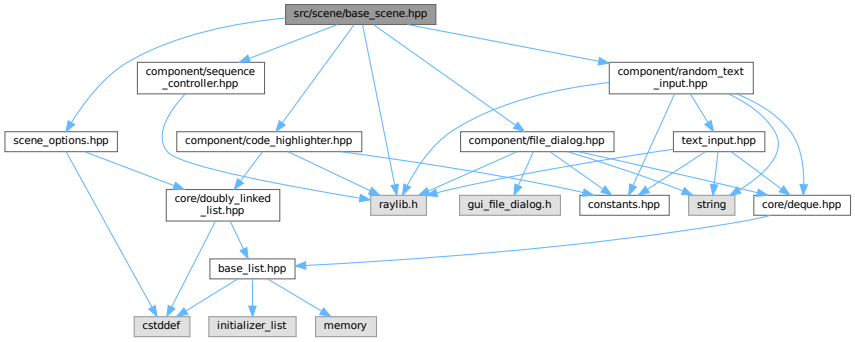
```
00001 #include "base_scene.hpp"
00002
00003 #include <cstring>
00004
00005 #include "constants.hpp"
00006 #include "raygui.h"
00007
00008 namespace scene::internal {
00009
00010 bool BaseScene::render_go_button() const {
00011     Rectangle shape{options_head, constants::scene_height - button_size.y,
00012                     button_size.y, button_size.y};
00013     return GuiButton(shape, "Go");
00014 }
00015
00016 void BaseScene::render_options(SceneOptions& scene_config) {
00017     (m_edit_mode || m_edit_action) ? GuiLock() : GuiUnlock();
00018
00019     options_head = 2 * constants::sidebar_width;
00020
00021     Rectangle mode_button_shape{options_head,
00022                                 constants::scene_height - button_size.y,
00023                                 button_size.x, button_size.y};
00024
00025     options_head += (button_size.x + head_offset);
00026
00027     int& mode = scene_config.mode_selection;
00028
00029     if (GuiDropupBox(mode_button_shape, scene_config.mode_labels, &mode,
00030                      m_edit_mode)) {
00031         m_edit_mode ^= 1;
00032     }
00033
00034     if (std::strlen(scene_config.action_labels.at(mode)) != 0) {
00035         Rectangle action_button_shape{options_head,
00036                                       constants::scene_height - button_size.y,
00037                                       button_size.x, button_size.y};
00038
00039         options_head += (button_size.x + head_offset);
00040
00041         int& action_selection = scene_config.action_selection.at(mode);
00042
```

```
00043         if (GuiDropupBox(action_button_shape,
00044                          scene_config.action_labels.at(mode), &action_selection,
00045                          m_edit_action)) {
00046             m_edit_action ^= 1;
00047         }
00048
00049         // scene_config.action_selection.at(mode) = GuiComboBox(
00050         //     action_button_shape, scene_config.action_labels.at(mode),
00051         //     scene_config.action_selection.at(mode));
00052     }
00053
00054     render_inputs();
00055 }
00056
00057 }  // namespace scene::internal
```

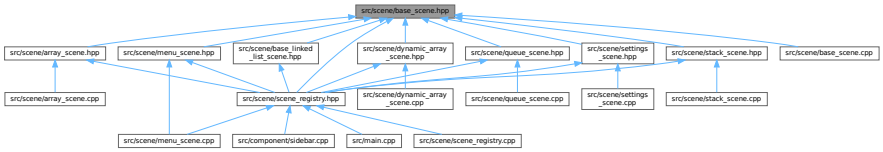
7.79 src/scene/base_scene.hpp File Reference

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

Include dependency graph for base_scene.hpp:



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



Classes

- class scene::internal::BaseScene

  The base scene class.

Namespaces

- namespace scene
- namespace scene::internal

7.80 base_scene.hpp

Go to the documentation of this file.

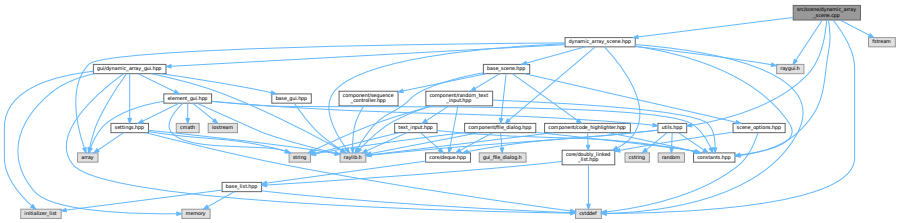
```
00001 #ifndef SCENE_BASE_SCENE_HPP_
00002 #define SCENE_BASE_SCENE_HPP_
00003
00004 #include "component/code_highlighter.hpp"
00005 #include "component/file_dialog.hpp"
00006 #include "component/random_text_input.hpp"
00007 #include "component/sequence_controller.hpp"
00008 #include "raylib.h"
00009 #include "scene_options.hpp"
00010
00011 namespace scene::internal {
00012
00017 class BaseScene {
00018 public:
00022     BaseScene() = default;
00023
00027     BaseScene(const BaseScene&) = delete;
00028
00032     BaseScene(BaseScene&&) = delete;
00033
00037     BaseScene& operator=(const BaseScene&) = delete;
00038
00042     BaseScene& operator=(BaseScene&&) = delete;
00043
00047     virtual ~BaseScene() = default;
00048
00052     virtual void render() {}
00053
00057     virtual void interact() {}
00058
00059 protected:
00063     static constexpr Vector2 button_size{200, 50};
00064
00068     static constexpr int head_offset = 20;
00069
00073     float options_head{};
00074
00080     virtual bool render_go_button() const;
00081
00086     virtual void render_options(SceneOptions& scene_config);
00087
00091     virtual void render_inputs() {}
00092
00096     component::RandomTextInput m_text_input{"value"};
00097
00101     component::RandomTextInput m_index_input{"index"};
00102
00106     component::FileDialog m_file_dialog;
00107
00111     component::SequenceController m_sequence_controller;
00112
00116     component::CodeHighlighter m_code_highlighter;
00117
00121     bool m_edit_mode{};
00122
00126     bool m_edit_action{};
00127 };
00128
00129 }  // namespace scene::internal
00130
00131 #endif  // SCENE_BASE_SCENE_HPP_
```

7.81 src/scene/dynamic_array_scene.cpp File Reference

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

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

Include dependency graph for dynamic_array_scene.cpp:



Namespaces

- namespace scene

7.82 dynamic_array_scene.cpp

Go to the documentation of this file.

```
00001 #include "dynamic_array_scene.hpp"
00002
00003 #include <cstddef>
00004 // #include <cstdlib>
00005 // #include <cstring>
00006 #include <fstream>
00007 // #include <iostream>
00008 // #include <limits>
00009 // #include <string>
00010
00011 #include "constants.hpp"
00012 #include "raygui.h"
00013 #include "utils.hpp"
00014
00015 namespace scene {
00016
00017 void DynamicArrayScene::render_inputs() {
00018     int& mode = scene_options.mode_selection;
00019
00020     switch (mode) {
00021         case 0: {
00022             switch (scene_options.action_selection.at(mode)) {
00023                 case 0:
00024                     break;
00025                 case 1: {
00026                     m_text_input.render_head(options_head, head_offset);
00027                 } break;
00028                 case 2: {
00029                     m_go = (m_file_dialog.render_head(options_head,
00030                                                       head_offset) > 0);
00031                     return;
00032                 } break;
00033                 default:
00034                     utils::unreachable();
00035             }
00036         } break;
00037
00038         case 1: {
00039             m_index_input.render_head(options_head, head_offset);
00040         } break;
00041
00042         case 2: {
00043             switch (scene_options.action_selection.at(mode)) {
00044                 case 0: {
00045                     m_text_input.render_head(options_head, head_offset);
00046                 } break;
00047                 case 1:
```

```
00048                     break;
00049                 default:
00050                     utils::unreachable();
00051             }
00052         } break;
00053
00054         case 3: {
00055             m_index_input.render_head(options_head, head_offset);
00056             m_text_input.render_head(options_head, head_offset);
00057         } break;
00058
00059         case 4: {
00060             m_text_input.render_head(options_head, head_offset);
00061         } break;
00062
00063         case 5: {
00064             m_index_input.render_head(options_head, head_offset);
00065             m_text_input.render_head(options_head, head_offset);
00066         } break;
00067
00068         case 6: {
00069             m_index_input.render_head(options_head, head_offset);
00070         } break;
00071
00072         default:
00073             utils::unreachable();
00074     }
00075
00076     m_go |= render_go_button();
00077 }
00078
00079 void DynamicArrayScene::render() {
00080     m_sequence_controller.inc_anim_counter();
00081
00082     int frame_idx = m_sequence_controller.get_anim_frame();
00083     auto* const frame_ptr = m_sequence.find(frame_idx);
00084     m_sequence_controller.set_progress_value(frame_idx);
00085
00086     if (frame_ptr != nullptr) {
00087         frame_ptr->data.render();
00088         m_code_highlighter.highlight(frame_idx);
00089     } else {  // end of sequence
00090         m_array.render();
00091         m_sequence_controller.set_run_all(false);
00092     }
00093
00094     m_code_highlighter.render();
00095     m_sequence_controller.render();
00096     render_options(scene_options);
00097 }
00098
00099 void DynamicArrayScene::interact() {
00100     if (m_sequence_controller.interact()) {
00101         m_sequence_controller.reset_anim_counter();
00102         return;
00103     }
00104
00105     m_index_input.set_random_max((int)m_array.size() - 1);
00106
00107     if (m_text_input.interact() || m_index_input.interact()) {
00108         return;
00109     }
00110
00111     if (!m_go) {
00112         return;
00113     }
00114
00115     int& mode = scene_options.mode_selection;
00116
00117     switch (mode) {
00118         case 0: {
00119             switch (scene_options.action_selection.at(mode)) {
00120                 case 0: {
00121                     interact_random();
00122                 } break;
00123
00124                 case 1: {
00125                     interact_import(m_text_input.extract_values());
00126                 } break;
00127
00128                 case 2: {
00129                     interact_file_import();
00130                 } break;
00131
00132                 default:
00133                     utils::unreachable();
00134             }
```

```
00135
00136             m_code_highlighter.set_code({});
00137             m_sequence.clear();
00138             m_sequence_controller.set_max_value(0);
00139         } break;
00140
00141         case 1: {
00142             interact_access();
00143         } break;
00144
00145         case 2: {
00146             switch (scene_options.action_selection.at(mode)) {
00147                 case 0: {
00148                     interact_reserve();
00149                 } break;
00150
00151                 case 1: {
00152                     interact_shrink();
00153                 } break;
00154
00155                 default:
00156                     utils::unreachable();
00157             }
00158         } break;
00159
00160         case 3: {
00161             interact_update();
00162         } break;
00163
00164         case 4: {
00165             interact_search();
00166         } break;
00167
00168         case 5: {
00169             m_index_input.set_random_max((int)m_array.size());
00170             interact_insert();
00171         } break;
00172
00173         case 6: {
00174             interact_delete();
00175         } break;
00176
00177         default:
00178             utils::unreachable();
00179     }
00180
00181     m_go = false;
00182 }
00183
00184 void DynamicArrayScene::interact_access() {
00185     auto index_container = m_index_input.extract_values();
00186     if (index_container.empty()) {
00187         return;
00188     }
00189
00190     std::size_t index = index_container.front();
00191     if (index >= m_array.size()) {
00192         return;
00193     }
00194
00195     m_code_highlighter.set_code({"return m_array[index];"});
00196
00197     m_sequence.clear();
00198
00199     m_array.set_color_index(index, 3);
00200     m_sequence.insert(m_sequence.size(), m_array);
00201     m_code_highlighter.push_into_sequence(0);
00202
00203     m_array.set_color_index(index, 0);
00204
00205     m_sequence_controller.set_max_value((int)m_sequence.size());
00206     m_sequence_controller.set_rerun();
00207 }
00208
00209 void DynamicArrayScene::interact_reserve() {
00210     auto value_container = m_text_input.extract_values();
00211     if (value_container.empty()) {
00212         return;
00213     }
00214
00215     std::size_t size = value_container.front();
00216     m_array.reserve(size);
00217 }
00218
00219 void DynamicArrayScene::interact_shrink() { m_array.shrink_to_fit(); }
00220
00221 void DynamicArrayScene::interact_random() {
```

```
00222     std::size_t size =
00223         utils::get_random(std::size_t{1}, scene_options.max_size);
00224     m_array = {};
00225
00226     for (std::size_t i = 0; i < size; ++i) {
00227         m_array.push(utils::get_random(constants::min_val, constants::max_val));
00228     }
00229
00230     m_array.shrink_to_fit();
00231 }
00232
00233 void DynamicArrayScene::interact_import(core::Deque<int> nums) {
00234     m_array = {};
00235     std::size_t i;  // NOLINT
00236
00237     for (i = 0; i < max_size && !nums.empty(); ++i) {
00238         m_array.push(nums.front());
00239         nums.pop_front();
00240     }
00241
00242     m_array.shrink_to_fit();
00243 }
00244
00245 void DynamicArrayScene::interact_update() {
00246     auto index_container = m_index_input.extract_values();
00247     if (index_container.empty()) {
00248         return;
00249     }
00250
00251     auto value_container = m_text_input.extract_values();
00252     if (value_container.empty()) {
00253         return;
00254     }
00255
00256     int index = index_container.front();
00257     int value = value_container.front();
00258
00259     if (!(0 <= index && index < m_array.size()) ||
00260         !utils::val_in_range(value)) {
00261         return;
00262     }
00263
00264     m_code_highlighter.set_code({
00265         "array[index] = value;",
00266     });
00267
00268     m_sequence.clear();
00269
00270     // initial state (before update)
00271     m_sequence.insert(m_sequence.size(), m_array);
00272     m_code_highlighter.push_into_sequence(-1);
00273
00274     // highlight
00275     m_array.set_color_index(index, 2);
00276     m_sequence.insert(m_sequence.size(), m_array);
00277     m_code_highlighter.push_into_sequence(0);
00278
00279     // update
00280     m_array[index] = value;
00281     m_array.set_color_index(index, 3);
00282     m_sequence.insert(m_sequence.size(), m_array);
00283     m_code_highlighter.push_into_sequence(0);
00284
00285     // undo highlight
00286     m_array.set_color_index(index, 0);
00287
00288     m_sequence_controller.set_max_value((int)m_sequence.size());
00289     m_sequence_controller.set_rerun();
00290 }
00291
00292 void DynamicArrayScene::interact_file_import() {
00293     interact_import(m_file_dialog.extract_values());
00294 }
00295
00296 void DynamicArrayScene::interact_search() {
00297     auto value_container = m_text_input.extract_values();
00298     if (value_container.empty()) {
00299         return;
00300     }
00301
00302     int value = value_container.front();
00303     if (!utils::val_in_range(value)) {
00304         return;
00305     }
00306
00307     m_code_highlighter.set_code({
00308         "for (i = 0; i < size; i++)",
```

```
00309         "    if (array[i] == value)",
00310         "        return i;",
00311         "return not_found",
00312     });
00313
00314     m_sequence.clear();
00315     m_sequence.insert(m_sequence.size(), m_array);
00316     m_code_highlighter.push_into_sequence(0);
00317
00318     bool found = false;
00319
00320     for (std::size_t i = 0; i < m_array.size(); ++i) {
00321         m_array.set_color_index(i, 3);
00322         m_sequence.insert(m_sequence.size(), m_array);
00323         m_code_highlighter.push_into_sequence(1);
00324
00325         if (m_array[i] == value) {
00326             found = true;
00327             m_array.set_color_index(i, 4);
00328             m_sequence.insert(m_sequence.size(), m_array);
00329             m_code_highlighter.push_into_sequence(2);
00330             m_array.set_color_index(i, 0);
00331             break;
00332         }
00333
00334         m_array.set_color_index(i, 0);
00335         m_sequence.insert(m_sequence.size(), m_array);
00336         m_code_highlighter.push_into_sequence(0);
00337     }
00338
00339     if (!found) {
00340         m_sequence.insert(m_sequence.size(), m_array);
00341         m_code_highlighter.push_into_sequence(3);
00342     }
00343
00344     m_sequence_controller.set_max_value((int)m_sequence.size());
00345     m_sequence_controller.set_rerun();
00346 }
00347
00348 void DynamicArrayScene::interact_insert() {
00349     auto index_container = m_index_input.extract_values();
00350     if (index_container.empty()) {
00351         return;
00352     }
00353
00354     auto value_container = m_text_input.extract_values();
00355     if (value_container.empty()) {
00356         return;
00357     }
00358
00359     int index = index_container.front();
00360     int value = value_container.front();
00361
00362     if (m_array.size() >= max_size) {
00363         return;
00364     }
00365
00366     if (!(0 <= index && index <= m_array.size()) ||
00367         !utils::val_in_range(value)) {
00368         return;
00369     }
00370
00371     m_code_highlighter.set_code({
00372         "if (size == capacity)",
00373         "    capacity = max(capacity * 2, 1);",
00374         "size++;",
00375         "for (i = size - 1; i > index; i--)",
00376         "    array[i] = array[i - 1];",
00377         "array[index] = value;",
00378     });
00379
00380     m_sequence.clear();
00381     m_sequence.insert(m_sequence.size(), m_array);
00382     m_code_highlighter.push_into_sequence(-1);
00383
00384     m_sequence.insert(m_sequence.size(), m_array);
00385     m_code_highlighter.push_into_sequence(0);
00386
00387     if (m_array.size() == m_array.capacity()) {
00388         m_array.reserve(m_array.size() * 2);
00389         m_sequence.insert(m_sequence.size(), m_array);
00390         m_code_highlighter.push_into_sequence(1);
00391     }
00392
00393     m_array.push(0);
00394     m_sequence.insert(m_sequence.size(), m_array);
00395     m_code_highlighter.push_into_sequence(2);
```

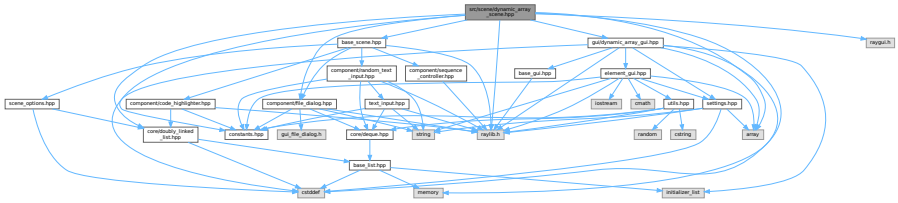
```
00396
00397     for (std::size_t i = m_array.size() - 1; i > index; --i) {
00398         m_array.set_color_index(i - 1, 2);
00399         m_sequence.insert(m_sequence.size(), m_array);
00400         m_code_highlighter.push_into_sequence(3);
00401
00402         m_array[i] = m_array[i - 1];
00403         m_array.set_color_index(i, 3);
00404         m_sequence.insert(m_sequence.size(), m_array);
00405         m_code_highlighter.push_into_sequence(4);
00406
00407         m_array.set_color_index(i - 1, 0);
00408         m_array.set_color_index(i, 0);
00409         m_sequence.insert(m_sequence.size(), m_array);
00410         m_code_highlighter.push_into_sequence(3);
00411     }
00412
00413     m_array.set_color_index(index, 2);
00414     m_sequence.insert(m_sequence.size(), m_array);
00415     m_code_highlighter.push_into_sequence(5);
00416
00417     m_array[index] = value;
00418     m_array.set_color_index(index, 3);
00419     m_sequence.insert(m_sequence.size(), m_array);
00420     m_code_highlighter.push_into_sequence(5);
00421
00422     m_array.set_color_index(index, 0);
00423     m_sequence.insert(m_sequence.size(), m_array);
00424     m_code_highlighter.push_into_sequence(-1);
00425
00426     m_sequence_controller.set_max_value((int)m_sequence.size());
00427     m_sequence_controller.set_rerun();
00428 }
00429
00430 void DynamicArrayScene::interact_delete() {
00431     auto index_container = m_index_input.extract_values();
00432     if (index_container.empty()) {
00433         return;
00434     }
00435
00436     int index = index_container.front();
00437
00438     if (m_array.size() == 0) {
00439         return;
00440     }
00441
00442     if (!(0 <= index && index < m_array.size())) {
00443         return;
00444     }
00445
00446     m_code_highlighter.set_code({
00447         "for (i = index; i < size - 1; i++)",
00448         "    array[i] = array[i + 1];",
00449         "size--;",
00450     });
00451
00452     m_sequence.clear();
00453     m_sequence.insert(m_sequence.size(), m_array);
00454     m_code_highlighter.push_into_sequence(-1);
00455
00456     m_sequence.insert(m_sequence.size(), m_array);
00457     m_code_highlighter.push_into_sequence(0);
00458
00459     for (std::size_t i = index; i < m_array.size() - 1; ++i) {
00460         m_array.set_color_index(i, 3);
00461         m_sequence.insert(m_sequence.size(), m_array);
00462         m_code_highlighter.push_into_sequence(1);
00463
00464         m_array[i] = m_array[i + 1];
00465         m_array.set_color_index(i + 1, 2);
00466         m_sequence.insert(m_sequence.size(), m_array);
00467         m_code_highlighter.push_into_sequence(1);
00468
00469         m_array.set_color_index(i, 0);
00470         m_array.set_color_index(i + 1, 0);
00471         m_sequence.insert(m_sequence.size(), m_array);
00472         m_code_highlighter.push_into_sequence(0);
00473     }
00474
00475     m_array.set_color_index(m_array.size() - 1, 2);
00476     m_sequence.insert(m_sequence.size(), m_array);
00477     m_code_highlighter.push_into_sequence(2);
00478
00479     m_array.pop();
00480     m_sequence.insert(m_sequence.size(), m_array);
00481     m_code_highlighter.push_into_sequence(-1);
00482
```

```
00483     m_sequence_controller.set_max_value((int)m_sequence.size());
00484     m_sequence_controller.set_rerun();
00485 }
00486
00487 }  // namespace scene
```

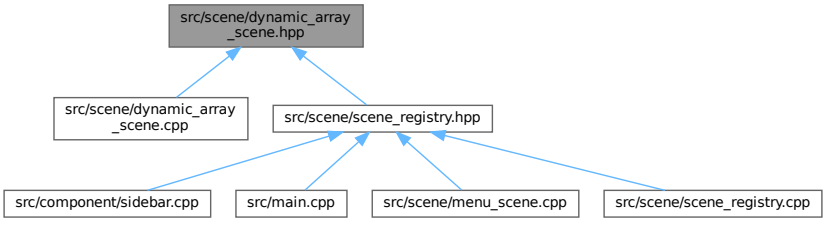
7.83 src/scene/dynamic_array_scene.hpp File Reference

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

Include dependency graph for dynamic_array_scene.hpp:



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



Classes

- class scene::DynamicArrayScene

  The dynamic array scene.

Namespaces

- namespace scene

7.84 dynamic_array_scene.hpp

Go to the documentation of this file.

```
00001 #ifndef SCENE_DYNAMIC_ARRAY_SCENE_HPP_
00002 #define SCENE_DYNAMIC_ARRAY_SCENE_HPP_
00003
00004 #include <array>
00005 #include <cstddef>
00006
00007 #include "base_scene.hpp"
00008 #include "component/file_dialog.hpp"
00009 #include "constants.hpp"
00010 #include "core/doubly_linked_list.hpp"
00011 #include "gui/dynamic_array_gui.hpp"
00012 #include "raygui.h"
00013 #include "raylib.h"
00014
00015 namespace scene {
00016
00021 class DynamicArrayScene : public internal::BaseScene {
00022 public:
00026     void render() override;
00027
00031     void interact() override;
00032
00033 private:
00037     static constexpr std::size_t max_size = 8;
00038
00042     internal::SceneOptions scene_options{
00043         // max_size
00044         max_size,
00045
00046         // mode_labels
00047         "Mode: Create;"
00048         "Mode: Access;"
00049         "Mode: Allocate;"
00050         "Mode: Update;"
00051         "Mode: Search;"
00052         "Mode: Insert;"
00053         "Mode: Delete",
00054
00055         // mode_selection
00056         0,
00057
00058         // action_labels
00059         {
00060             // Mode: Create
00061             "Action: Random;Action: Input;Action: File",
00062
00063             // Mode: Access
00064             "",
00065
00066             // Mode: Allocate
00067             "Action: Reserve;Action: Shrink",
00068
00069             // Mode: Update
00070             "",
00071
00072             // Mode: Search
00073             "",
00074
00075             // Mode: Insert
00076             "",
00077
00078             // Mode: Delete
00079             "",
00080         },
00081
00082         // action_selection
00083         core::DoublyLinkedList<int>{0, 0, 0, 0, 0, 0},
00084     };
00085
00086     using internal::BaseScene::button_size;
00087     using internal::BaseScene::head_offset;
00088     using internal::BaseScene::options_head;
00089
00093     gui::GuiDynamicArray<int> m_array{};
00094
00098     core::DoublyLinkedList<gui::GuiDynamicArray<int>> m_sequence;
00099
00103     bool m_go{};
00104
00105     using internal::BaseScene::m_file_dialog;
00106     using internal::BaseScene::m_index_input;
00107     using internal::BaseScene::m_sequence_controller;
```

```
00108     using internal::BaseScene::m_text_input;
00109
00110     using internal::BaseScene::render_go_button;
00111     using internal::BaseScene::render_options;
00112
00116     void render_inputs() override;
00117
00121     void interact_random();
00122
00126     void interact_import(core::Deque<int> nums);
00127
00131     void interact_file_import();
00132
00136     void interact_update();
00137
00141     void interact_search();
00142
00146     void interact_insert();
00147
00151     void interact_delete();
00152
00156     void interact_reserve();
00157
00161     void interact_shrink();
00162
00166     void interact_access();
00167 };
00168
00169 }  // namespace scene
00170
00171 #endif  // SCENE_DYNAMIC_ARRAY_SCENE_HPP_
```

7.85 src/scene/menu_scene.cpp File Reference

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

Include dependency graph for menu_scene.cpp:

Namespaces

- namespace scene

7.86 menu_scene.cpp

Go to the documentation of this file.

```
00001 #include "menu_scene.hpp"
00002
00003 #include <iostream>
00004
00005 #include "constants.hpp"
00006 #include "raygui.h"
```

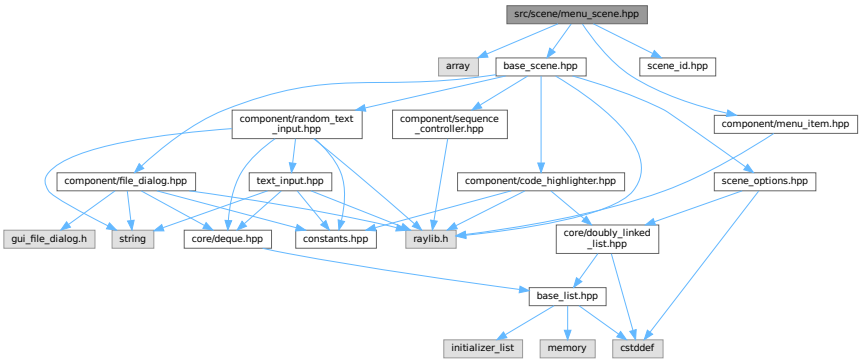
```
00007 #include "raylib.h"
00008 #include "scene_registry.hpp"
00009 #include "settings.hpp"
00010 #include "utils.hpp"
00011
00012 namespace scene {
00013
00014 MenuScene::MenuScene() {
00015     constexpr int block_width = component::MenuItem::block_width;
00016     constexpr int block_height = component::MenuItem::block_height;
00017     constexpr int button_width = component::MenuItem::button_width;
00018     constexpr int button_height = component::MenuItem::button_height;
00019     constexpr int gap = 20;
00020
00021     constexpr int first_row_y =
00022         constants::scene_height / 16.0F * 5 - block_height / 2.0F;
00023
00024     // first row
00025     {
00026         constexpr int row_width =
00027             3 * component::MenuItem::block_width + 2 * gap;
00028         constexpr int row_x = constants::scene_width / 2.0F - row_width / 2.0F;
00029         constexpr int row_y = first_row_y;
00030
00031         for (auto i = 0; i < 3; ++i) {
00032             m_menu_items[i] =
00033                 component::MenuItem(labels[i], row_x + i * (block_width + gap),
00034                                     row_y, img_paths[i]);
00035         }
00036     }
00037
00038     // second row
00039     {
00040         constexpr int row_width = 4 * block_width + 3 * gap;
00041         constexpr int row_x = constants::scene_width / 2.0F - row_width / 2.0F;
00042         constexpr int row_y = first_row_y + block_height + gap;
00043
00044         for (auto i = 3; i < 7; ++i) {
00045             m_menu_items[i] = component::MenuItem(
00046                 labels[i], row_x + (i - 3) * (block_width + gap), row_y,
00047                 img_paths[i]);
00048         }
00049     }
00050 }
00051
00052 void MenuScene::render() {
00053     const Color text_color = utils::adaptive_text_color(
00054         Settings::get_instance().get_color(Settings::num_color - 1));
00055
00056     // Menu text
00057     constexpr int menu_font_size = 60;
00058     constexpr int menu_font_spacing = 5;
00059
00060     constexpr const char* menu_text = "CS162 - VisuAlgo.net clone in C++";
00061
00062     const Vector2 menu_text_size =
00063         utils::MeasureText(menu_text, menu_font_size, menu_font_spacing);
00064
00065     const Vector2 menu_text_pos{
00066         constants::scene_width / 2.0F - menu_text_size.x / 2,
00067         constants::scene_height / 16.0F - menu_text_size.y / 2};
00068
00069     utils::DrawText(menu_text, menu_text_pos, text_color, menu_font_size,
00070                     menu_font_spacing);
00071
00072     // Sub text
00073     constexpr int sub_font_size = 30;
00074     constexpr int sub_font_spacing = 2;
00075
00076     constexpr const char* sub_text = "By Quang-Truong Nguyen (@jalsol)";
00077
00078     const Vector2 sub_text_size =
00079         utils::MeasureText(sub_text, sub_font_size, sub_font_spacing);
00080
00081     const Vector2 sub_text_pos{
00082         constants::scene_width / 2.0F - sub_text_size.x / 2,
00083         menu_text_pos.y + menu_text_size.y / 2 + sub_text_size.y};
00084
00085     utils::DrawText(sub_text, sub_text_pos, text_color, sub_font_size,
00086                     sub_font_spacing);
00087
00088     // Button
00089     constexpr int block_width = 300;
00090     constexpr int block_height = 200;
00091     constexpr int button_width = block_width;
00092     constexpr int button_height = 50;
00093     constexpr int gap = 20;
```

```
00094     constexpr int first_row_y =
00095         constants::scene_height / 16.0F * 5 - block_height / 2.0F;
00096
00097     for (auto i = 0; i < 7; ++i) {
00098         m_menu_items[i].render();
00099     }
00100
00101     const Rectangle quit_button_shape{
00102         constants::scene_width / 2.0F - 128,
00103         constants::scene_height / 16.0F * 15 - block_height / 2.0F, 256, 64};
00104
00105     m_quit = GuiButton(quit_button_shape, "Quit");
00106
00107     // Bottom text
00108     constexpr int bot_font_size = 20;
00109     constexpr int bot_font_spacing = 2;
00110
00111     constexpr const char* bot_text =
00112         "(pls read the src code, i tried so hard for this)";
00113
00114     const Vector2 bot_text_size =
00115         utils::MeasureText(bot_text, bot_font_size, bot_font_spacing);
00116
00117     const Vector2 bot_text_pos{
00118         constants::scene_width / 2.0F - bot_text_size.x / 2,
00119         constants::scene_height - 1.5F * bot_text_size.y};
00120
00121     utils::DrawText(bot_text, bot_text_pos, text_color, bot_font_size,
00122                     bot_font_spacing);
00123 }
00124
00125 void MenuScene::interact() {
00126     scene::SceneRegistry& registry = scene::SceneRegistry::get_instance();
00127
00128     if (m_quit) {
00129         registry.close_window();
00130         return;
00131     }
00132
00133     for (auto i = 0; i < 7; ++i) {
00134         if (m_menu_items[i].clicked()) {
00135             m_next_scene = SceneId(i);
00136             m_start = true;
00137         }
00138     }
00139
00140     for (auto i = 0; i < 7; ++i) {
00141         m_menu_items[i].reset();
00142     }
00143
00144     if (m_start) {
00145         registry.set_scene(m_next_scene);
00146         m_start = false;
00147     }
00148 }
00149
00150 }  // namespace scene
```

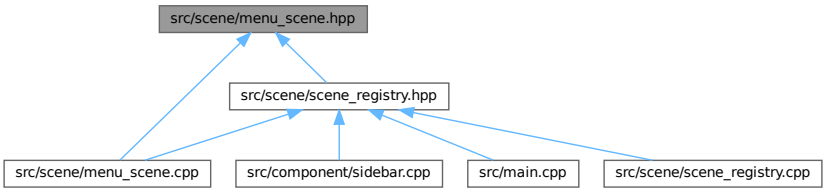
7.87 src/scene/menu_scene.hpp File Reference

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

Include dependency graph for menu_scene.hpp:



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



Classes

- class scene::MenuScene

  The menu scene.

Namespaces

- namespace scene

7.88 menu_scene.hpp

Go to the documentation of this file.

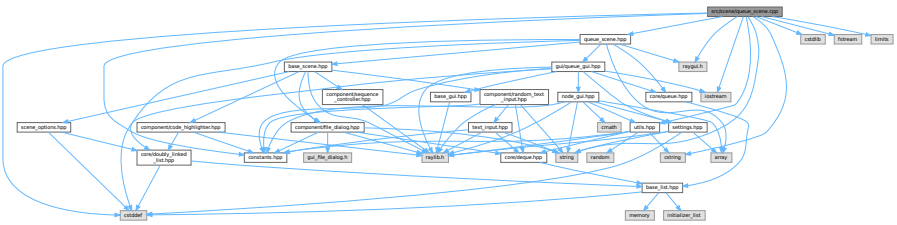
```
00001 #ifndef SCENE_MENU_SCENE_HPP_
00002 #define SCENE_MENU_SCENE_HPP_
00003
00004 #include <array>
00005
00006 #include "base_scene.hpp"
00007 #include "component/menu_item.hpp"
00008 #include "scene_id.hpp"
00009
00010 namespace scene {
00011
00016 class MenuScene : public internal::BaseScene {
00017 public:
00021     MenuScene();
```

```
00022
00026     void render() override;
00027
00031     void interact() override;
00032
00033 private:
00037     bool m_start{};
00038
00042     bool m_quit{};
00043
00047     SceneId m_next_scene{};
00048
00052     static constexpr std::array<const char*, 7> labels = {{
00053         "Array",
00054         "Dynamic Array",
00055         "Linked List",
00056         "Doubly Linked List",
00057         "Circular Linked List",
00058         "Stack",
00059         "Queue",
00060     }};
00061
00065     static constexpr std::array<const char*, 7> img_paths = {{
00066         "data/preview/array.png",
00067         "data/preview/dynamic_array.png",
00068         "data/preview/linked_list.png",
00069         "data/preview/doubly_linked_list.png",
00070         "data/preview/circular_linked_list.png",
00071         "data/preview/stack.png",
00072         "data/preview/queue.png",
00073     }};
00074
00078     std::array<component::MenuItem, 7> m_menu_items{};
00079 };
00080
00081 }  // namespace scene
00082
00083 #endif  // SCENE_MENU_SCENE_HPP_
```

7.89 src/scene/queue_scene.cpp File Reference

```
#include "queue_scene.hpp"
#include <cstddef>
#include <cstdlib>
#include <cstring>
#include <fstream>
#include <iostream>
#include <limits>
#include <string>
#include "constants.hpp"
#include "raygui.h"
#include "utils.hpp"
```

Include dependency graph for queue_scene.cpp:



Namespaces

- namespace scene

7.90 queue_scene.cpp

Go to the documentation of this file.

```
00001 #include "queue_scene.hpp"
00002
00003 #include <cstddef>
00004 #include <cstdlib>
00005 #include <cstring>
00006 #include <fstream>
00007 #include <iostream>
00008 #include <limits>
00009 #include <string>
00010
00011 #include "constants.hpp"
00012 #include "raygui.h"
00013 #include "utils.hpp"
00014
00015 namespace scene {
00016
00017 void QueueScene::render_inputs() {
00018     int& mode = scene_options.mode_selection;
00019
00020     switch (mode) {
00021         case 0: {
00022             switch (scene_options.action_selection.at(mode)) {
00023                 case 0:
00024                     break;
00025                 case 1: {
00026                     m_text_input.render_head(options_head, head_offset);
00027                 } break;
00028                 case 2: {
00029                     m_go = (m_file_dialog.render_head(options_head,
00030                                                       head_offset) > 0);
00031                     return;
00032                 } break;
00033                 default:
00034                     utils::unreachable();
00035             }
00036         } break;
00037
00038         case 1: {
00039             m_text_input.render_head(options_head, head_offset);
00040         } break;
00041
00042         case 2:
00043             break;
00044         default:
00045             utils::unreachable();
00046     }
00047
00048     m_go |= render_go_button();
00049 }
00050
00051 void QueueScene::render() {
00052     m_sequence_controller.inc_anim_counter();
00053
00054     int frame_idx = m_sequence_controller.get_anim_frame();
00055     auto* const frame_ptr = m_sequence.find(frame_idx);
00056     m_sequence_controller.set_progress_value(frame_idx);
00057
00058     if (frame_ptr != nullptr) {
00059         frame_ptr->data.render();
00060         m_code_highlighter.highlight(frame_idx);
00061     } else {  // end of sequence
00062         m_queue.render();
00063         m_sequence_controller.set_run_all(false);
00064     }
00065
00066     m_code_highlighter.render();
00067     m_sequence_controller.render();
00068     render_options(scene_options);
00069 }
00070
00071 void QueueScene::interact() {
00072     if (m_sequence_controller.interact()) {
00073         m_sequence_controller.reset_anim_counter();
00074         return;
00075     }
00076
00077     m_index_input.set_random_max((int)m_queue.size() - 1);
00078
00079     if (m_text_input.interact() || m_index_input.interact()) {
00080         return;
00081     }
00082
```

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

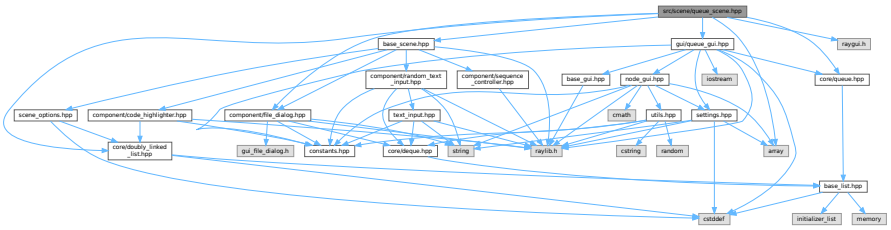
```
00170         "tail = tail->next;",
00171     });
00172
00173     m_sequence.clear();
00174     m_sequence.insert(m_sequence.size(), m_queue);
00175     m_code_highlighter.push_into_sequence(-1);
00176
00177     m_queue.push(value);
00178     m_queue.back().set_color_index(6);
00179     m_sequence.insert(m_sequence.size(), m_queue);
00180     m_code_highlighter.push_into_sequence(0);
00181
00182     m_queue.pop_back();
00183     if (!m_queue.empty()) {
00184         m_queue.back().set_color_index(4);
00185     }
00186     m_queue.push(value);
00187     m_queue.back().set_color_index(6);
00188     m_sequence.insert(m_sequence.size(), m_queue);
00189     m_code_highlighter.push_into_sequence(1);
00190
00191     m_queue.pop_back();
00192     if (!m_queue.empty()) {
00193         m_queue.back().set_color_index(0);
00194         m_queue.back().set_label("");
00195     }
00196     m_queue.push(value);
00197     m_queue.back().set_color_index(3);
00198     m_queue.init_label();
00199     m_sequence.insert(m_sequence.size(), m_queue);
00200     m_code_highlighter.push_into_sequence(2);
00201
00202     m_queue.back().set_color_index(0);
00203
00204     m_sequence_controller.set_max_value((int)m_sequence.size());
00205     m_sequence_controller.set_rerun();
00206 }
00207
00208 void QueueScene::interact_pop() {
00209     if (m_queue.empty()) {
00210         return;
00211     }
00212
00213     m_code_highlighter.set_code({
00214         "Node* temp = head;",
00215         "head = head->next;",
00216         "delete temp;",
00217     });
00218
00219     m_sequence.clear();
00220     m_sequence.insert(m_sequence.size(), m_queue);
00221     m_code_highlighter.push_into_sequence(-1);
00222
00223     m_queue.front().set_color_index(5);
00224     m_sequence.insert(m_sequence.size(), m_queue);
00225     m_code_highlighter.push_into_sequence(0);
00226
00227     auto old_front = m_queue.front();
00228     m_queue.pop();
00229
00230     if (!m_queue.empty()) {
00231         m_queue.front().set_color_index(3);
00232         if (m_queue.size() == 1) {
00233             m_queue.front().set_label("head/tail");
00234         } else {
00235             m_queue.front().set_label("head");
00236         }
00237     }
00238
00239     m_queue.push_front(old_front.get_value());
00240     m_queue.front().set_color_index(5);
00241     m_sequence.insert(m_sequence.size(), m_queue);
00242     m_code_highlighter.push_into_sequence(1);
00243
00244     m_queue.pop();
00245     m_queue.init_label();
00246     m_sequence.insert(m_sequence.size(), m_queue);
00247     m_code_highlighter.push_into_sequence(2);
00248
00249     if (!m_queue.empty()) {
00250         m_queue.front().set_color_index(0);
00251     }
00252
00253     m_sequence_controller.set_max_value((int)m_sequence.size());
00254     m_sequence_controller.set_rerun();
00255 }
00256
```

```
00257 void QueueScene::interact_clear() {
00258     m_queue = gui::GuiQueue<int>();
00259     m_sequence.clear();
00260     m_code_highlighter.clear();
00261     m_sequence_controller.set_max_value(0);
00262 }
00263
00264 }  // namespace scene
```

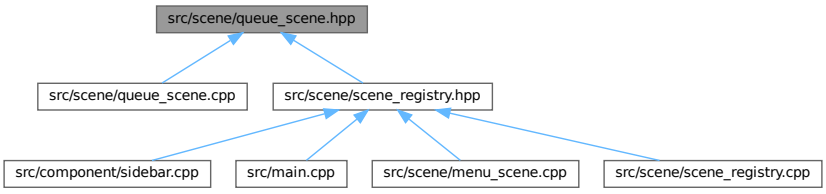
7.91 src/scene/queue_scene.hpp File Reference

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

Include dependency graph for queue_scene.hpp:



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



Classes

- class scene::QueueScene

  The queue scene.

Namespaces

- namespace scene

7.92 queue_scene.hpp

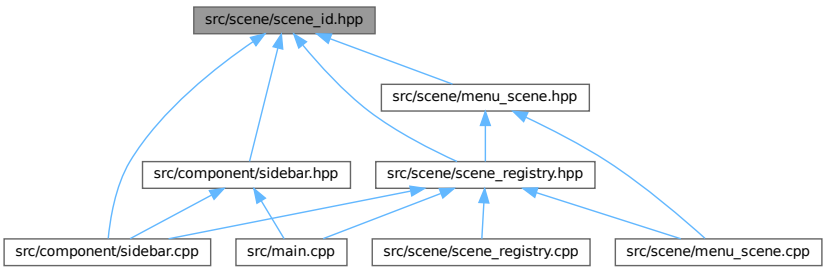
Go to the documentation of this file.

```
00001 #ifndef SCENE_QUEUE_SCENE_HPP_
00002 #define SCENE_QUEUE_SCENE_HPP_
00003
00004 #include <array>
00005
00006 #include "base_scene.hpp"
00007 #include "component/file_dialog.hpp"
00008 #include "core/doubly_linked_list.hpp"
00009 #include "core/queue.hpp"
00010 #include "gui/queue_gui.hpp"
00011 #include "raygui.h"
00012
00013 namespace scene {
00014
00019 class QueueScene : public internal::BaseScene {
00020 public:
00024     void render() override;
00025
00029     void interact() override;
00030
00031 private:
00035     internal::SceneOptions scene_options{
00036         // max_size
00037         8,  // NOLINT
00038
00039         // mode_labels
00040         "Mode: Create;"
00041         "Mode: Push;"
00042         "Mode: Pop;"
00043         "Mode: Clear",
00044
00045         // mode_selection
00046         0,
00047
00048         // action_labels
00049         {
00050             // Mode: Create
00051             "Action: Random;"
00052             "Action: Input;"
00053             "Action: File",
00054
00055             // Mode: Push
00056             "",
00057
00058             // Mode: Pop
00059             "",
00060
00061             // Mode: Clear
00062             "",
00063         },
00064
00065         // action_selection
00066         core::DoublyLinkedList<int>{0, 0, 0, 0},
00067     };
00068
00069     using internal::BaseScene::button_size;
00070     using internal::BaseScene::head_offset;
00071     using internal::BaseScene::options_head;
00072
00076     gui::GuiQueue<int> m_queue{
00077         gui::GuiNode<int>{1},
00078         gui::GuiNode<int>{2},
00079         gui::GuiNode<int>{3},
00080     };
00081
00085     core::DoublyLinkedList<gui::GuiQueue<int>> m_sequence;
00086
00090     bool m_go{};
00091     using internal::BaseScene::m_code_highlighter;
00092     using internal::BaseScene::m_file_dialog;
00093     using internal::BaseScene::m_sequence_controller;
00094     using internal::BaseScene::m_text_input;
00095
00096     using internal::BaseScene::render_go_button;
00097     using internal::BaseScene::render_options;
00098
00102     void render_inputs() override;
00103
00107     void interact_random();
00108
00112     void interact_import(core::Deque<int> nums);
00113
```

```
00117     void interact_file_import();
00118
00122     void interact_push();
00123
00127     void interact_pop();
00128
00132     void interact_clear();
00133 };
00134
00135 }  // namespace scene
00136
00137 #endif  // SCENE_QUEUE_SCENE_HPP_
```

7.93 src/scene/scene_id.hpp File Reference

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



Namespaces

- namespace scene

Enumerations

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

  The scene ID.

7.94 scene_id.hpp

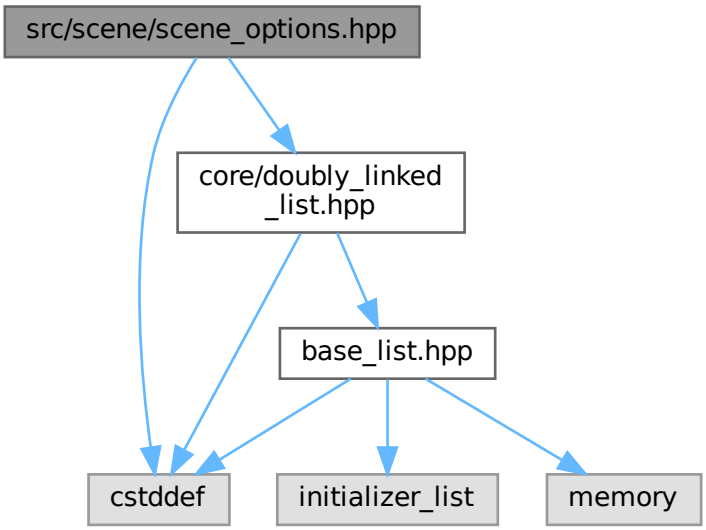
Go to the documentation of this file.

```
00001 #ifndef SCENE_SCENE_ID_HPP_
00002 #define SCENE_SCENE_ID_HPP_
00003
00004 namespace scene {
00005
00010 enum SceneId {
00011     Array,
00012     DynamicArray,
00013     LinkedList,
00014     DoublyLinkedList,
00015     CircularLinkedList,
00016     Stack,
00017     Queue,
00018     Menu,
00019     Settings,
00020 };
00021
00022 }  // namespace scene
00023
00024 #endif  // SCENE_SCENE_ID_HPP_
```

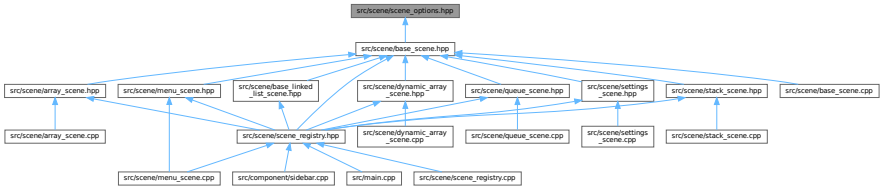
7.95 src/scene/scene_options.hpp File Reference

```
#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

  The scene options.

Namespaces

- namespace scene
- namespace scene::internal

7.96 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
00014 struct SceneOptions {
00018     const std::size_t max_size{};
00019
00023     const char* mode_labels{};
00024
00028     int mode_selection{};
00029
00033     core::DoublyLinkedList<const char*> action_labels;
00034
00038     core::DoublyLinkedList<int> action_selection;
00039 };
00040
00041 }  // namespace scene::internal
00042
00043 #endif  // SCENE_SCENE_OPTIONS_HPP_
```

7.97 src/scene/scene_registry.cpp File Reference

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

Namespaces

- namespace scene

7.98 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(SceneId scene_type) {
00013     m_current_scene = scene_type;
00014     scene_ptr = m_registry.at(scene_type).get();
00015 }
00016
00017 SceneId SceneRegistry::get_scene() const { return m_current_scene; }
00018
00019 void SceneRegistry::render() { scene_ptr->render(); }
```

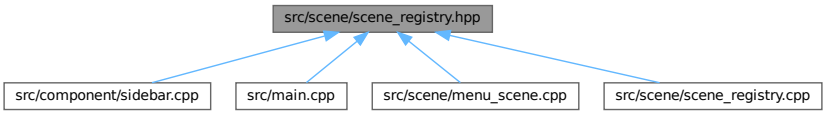
```
00020
00021 void SceneRegistry::interact() { scene_ptr->interact(); }
00022
00023 bool SceneRegistry::should_close() const { return m_should_close; }
00024
00025 void SceneRegistry::close_window() { m_should_close = true; }
00026
00027 }  // namespace scene
```

7.99 src/scene/scene_registry.hpp File Reference

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

Include dependency graph for scene_registry.hpp:

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



Classes

- class scene::SceneRegistry

  The scene registry.

Namespaces

- namespace scene

7.100 scene_registry.hpp

Go to the documentation of this file.

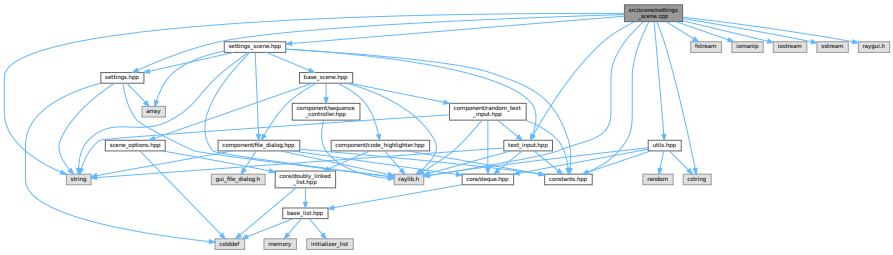
```
00001 #ifndef SCENE_SCENE_REGISTRY_HPP_
00002 #define SCENE_SCENE_REGISTRY_HPP_
00003
00004 #include <array>
00005 #include <memory>
00006
00007 #include "array_scene.hpp"
00008 #include "base_linked_list_scene.hpp"
00009 #include "base_scene.hpp"
00010 #include "dynamic_array_scene.hpp"
00011 #include "menu_scene.hpp"
00012 #include "queue_scene.hpp"
00013 #include "scene_id.hpp"
00014 #include "settings_scene.hpp"
00015 #include "stack_scene.hpp"
00016
00017 namespace scene {
00018
00022 class SceneRegistry {
00023 public:
00027     SceneRegistry(const SceneRegistry&) = delete;
00028
00032     SceneRegistry(SceneRegistry&&) = delete;
00033
00037     SceneRegistry& operator=(const SceneRegistry&) = delete;
00038
00042     SceneRegistry& operator=(SceneRegistry&&) = delete;
00043
00047     ~SceneRegistry() = default;
00048
00053     static SceneRegistry& get_instance();
00054
00059     void set_scene(SceneId scene_type);
00060
00065     SceneId get_scene() const;
00066
00070     void render();
00071
00075     void interact();
00076
00082     bool should_close() const;
00083
00087     void close_window();
00088
00089 private:
00093     internal::BaseScene* scene_ptr{};
00094
00098     SceneRegistry();
00099
00103     bool m_should_close{};
00104
00108     SceneId m_current_scene{};
00109
00113     const std::array<const std::unique_ptr<internal::BaseScene>, 9> m_registry{{
00114         std::make_unique<ArrayScene>(),
00115         std::make_unique<DynamicArrayScene>(),
00116         std::make_unique<LinkedListScene>(),
00117         std::make_unique<DoublyLinkedListScene>(),
00118         std::make_unique<CircularLinkedListScene>(),
00119         std::make_unique<StackScene>(),
00120         std::make_unique<QueueScene>(),
00121         std::make_unique<MenuScene>(),
00122         std::make_unique<SettingsScene>(),
00123     }};
00124 };
00125
00126 }  // namespace scene
00127
00128 #endif  // SCENE_SCENE_REGISTRY_HPP_
```

7.101 src/scene/settings_scene.cpp File Reference

```
#include "settings_scene.hpp"
#include <cstring>
#include <fstream>
```

```
#include <iomanip>
#include <iostream>
#include <sstream>
#include <string>
#include "component/text_input.hpp"
#include "constants.hpp"
#include "raygui.h"
#include "raylib.h"
#include "settings.hpp"
#include "utils.hpp"
```

Include dependency graph for settings_scene.cpp:



Namespaces

- namespace scene

7.102 settings_scene.cpp

Go to the documentation of this file.

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

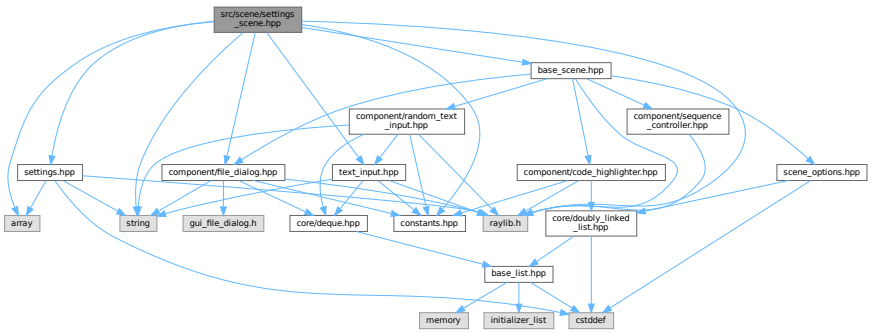
```
00037
00038     unsigned hex_value;
00039     for (auto i = 0; i < Settings::num_color; ++i) {
00040         file_in.read(reinterpret_cast<char*>(&hex_value), sizeof(hex_value));
00041         settings.get_color(i) = GetColor(hex_value);
00042     }
00043
00044     set_buffer();
00045 }
00046
00047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

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

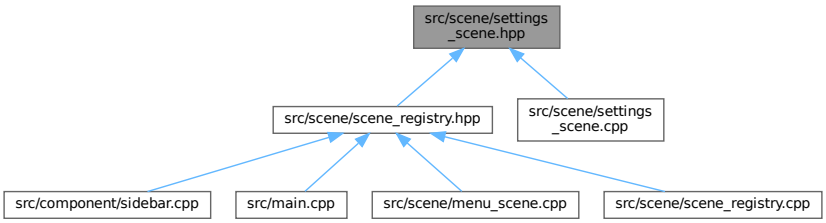
7.103 src/scene/settings_scene.hpp File Reference

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

Include dependency graph for settings_scene.hpp:



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



Classes

- class scene::SettingsScene

  The settings scene.

Namespaces

- namespace scene

7.104 settings_scene.hpp

Go to the documentation of this file.

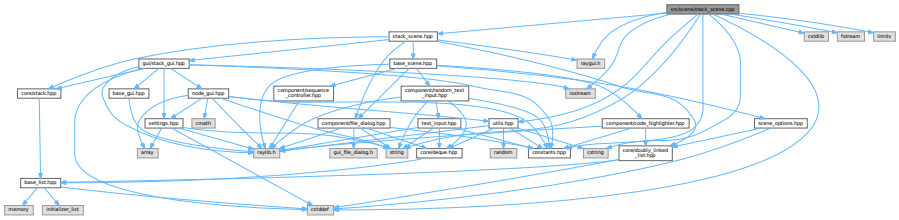
```
00001 #ifndef SCENE_SETTINGS_SCENE_HPP_
00002 #define SCENE_SETTINGS_SCENE_HPP_
00003
00004 #include <array>
00005 #include <constants.hpp>
00006 #include <string>
00007
00008 #include "base_scene.hpp"
00009 #include "component/file_dialog.hpp"
00010 #include "component/text_input.hpp"
00011 #include "raylib.h"
00012 #include "settings.hpp"
00013
00014 namespace scene {
00015
00020 class SettingsScene : public internal::BaseScene {
00021 public:
00025     SettingsScene();
00026
00030     void render() override;
00031
00035     void interact() override;
00036
00037 private:
00041     static constexpr Vector2 head_pos{400, 70};
00042
00046     std::array<component::TextInput, Settings::num_color> m_inputs{};
00047
00051     int m_selected{};
00052
00056     component::FileDialog m_open_file;
00057
00061     component::FileDialog m_save_file{3, "Save file...", "Save file"};
00062
00066     int m_open{};
00067
00071     int m_save{};
00072
00076     void set_buffer();
00077
```

```
00081     void set_color();
00082
00086     void open_from_file(const std::string& path);
00087 };
00088
00089 }  // namespace scene
00090
00091 #endif  // SCENE_SETTINGS_SCENE_HPP_
```

7.105 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

7.106 stack_scene.cpp

Go to the documentation of this file.

```
00001 #include "stack_scene.hpp"
00002
00003 #include <cstddef>
00004 #include <cstdlib>
00005 #include <cstring>
00006 #include <fstream>
00007 #include <iostream>
00008 #include <limits>
00009 #include <string>
00010
00011 #include "constants.hpp"
00012 #include "raygui.h"
00013 #include "utils.hpp"
00014
00015 namespace scene {
00016
00017 void StackScene::render() {
00018     m_sequence_controller.inc_anim_counter();
00019
00020     int frame_idx = m_sequence_controller.get_anim_frame();
```

```
00021     auto* const frame_ptr = m_sequence.find(frame_idx);
00022     m_sequence_controller.set_progress_value(frame_idx);
00023
00024     if (frame_ptr != nullptr) {
00025         frame_ptr->data.render();
00026         m_code_highlighter.highlight(frame_idx);
00027     } else {  // end of sequence
00028         m_stack.render();
00029         m_sequence_controller.set_run_all(false);
00030     }
00031
00032     m_code_highlighter.render();
00033     m_sequence_controller.render();
00034     render_options(scene_options);
00035 }
00036
00037 void StackScene::render_inputs() {
00038     int& mode = scene_options.mode_selection;
00039
00040     switch (mode) {
00041         case 0: {
00042             switch (scene_options.action_selection.at(mode)) {
00043                 case 0:
00044                     break;
00045                 case 1: {
00046                     m_text_input.render_head(options_head, head_offset);
00047                 } break;
00048                 case 2: {
00049                     m_go = (m_file_dialog.render_head(options_head,
00050                                                       head_offset) > 0);
00051                     return;
00052                 } break;
00053                 default:
00054                     utils::unreachable();
00055             }
00056         } break;
00057
00058         case 1: {
00059             m_text_input.render_head(options_head, head_offset);
00060         } break;
00061
00062         case 2:
00063         case 3:
00064             break;
00065         default:
00066             utils::unreachable();
00067     }
00068
00069     m_go |= render_go_button();
00070 }
00071
00072 void StackScene::interact() {
00073     if (m_sequence_controller.interact()) {
00074         m_sequence_controller.reset_anim_counter();
00075         return;
00076     }
00077
00078     m_index_input.set_random_max((int)m_stack.size() - 1);
00079     if (m_text_input.interact() || m_index_input.interact()) {
00080         return;
00081     }
00082
00083     if (!m_go) {
00084         return;
00085     }
00086
00087     int& mode = scene_options.mode_selection;
00088
00089     switch (mode) {
00090         case 0: {
00091             switch (scene_options.action_selection.at(mode)) {
00092                 case 0: {
00093                     interact_random();
00094                 } break;
00095
00096                 case 1: {
00097                     interact_import(m_text_input.extract_values());
00098                 } break;
00099
00100                 case 2: {
00101                     interact_file_import();
00102                 } break;
00103
00104                 default:
00105                     utils::unreachable();
00106             }
00107             m_code_highlighter.set_code({});
```

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

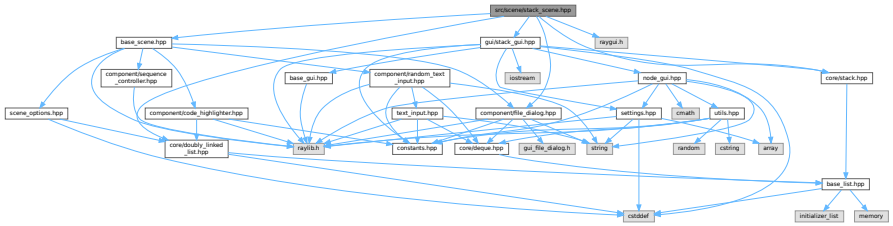
```
00195     }
00196     m_stack.push(value);
00197     m_stack.top().set_color_index(3);
00198     m_stack.init_label();
00199     m_sequence.insert(m_sequence.size(), m_stack);
00200     m_code_highlighter.push_into_sequence(2);
00201
00202     m_stack.top().set_color_index(0);
00203
00204     m_sequence_controller.set_max_value((int)m_sequence.size());
00205     m_sequence_controller.set_rerun();
00206 }
00207
00208 void StackScene::interact_pop() {
00209     if (m_stack.empty()) {
00210         return;
00211     }
00212
00213     m_code_highlighter.set_code({
00214         "Node* temp = head;",
00215         "head = head->next;",
00216         "delete temp;",
00217     });
00218
00219     m_sequence.clear();
00220     m_sequence.insert(m_sequence.size(), m_stack);
00221     m_code_highlighter.push_into_sequence(-1);
00222
00223     m_stack.top().set_color_index(5);
00224     m_sequence.insert(m_sequence.size(), m_stack);
00225     m_code_highlighter.push_into_sequence(0);
00226
00227     auto old_top = m_stack.top();
00228     m_stack.pop();
00229
00230     if (!m_stack.empty()) {
00231         m_stack.top().set_color_index(3);
00232         m_stack.top().set_label("head");
00233     }
00234
00235     m_stack.push(old_top.get_value());
00236     m_stack.top().set_color_index(5);
00237     m_sequence.insert(m_sequence.size(), m_stack);
00238     m_code_highlighter.push_into_sequence(1);
00239
00240     m_stack.pop();
00241     m_sequence.insert(m_sequence.size(), m_stack);
00242     m_code_highlighter.push_into_sequence(2);
00243
00244     if (!m_stack.empty()) {
00245         m_stack.top().set_color_index(0);
00246     }
00247
00248     m_sequence_controller.set_max_value((int)m_sequence.size());
00249     m_sequence_controller.set_rerun();
00250 }
00251
00252 void StackScene::interact_file_import() {
00253     interact_import(m_file_dialog.extract_values());
00254 }
00255
00256 void StackScene::interact_clear() {
00257     m_stack = gui::GuiStack<int>();
00258     m_sequence.clear();
00259     m_code_highlighter.set_code({});
00260     m_sequence_controller.set_max_value(0);
00261 }
00262
00263 }  // namespace scene
```

7.107 src/scene/stack_scene.hpp File Reference

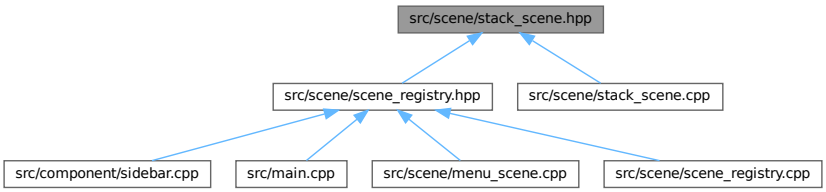
```
#include "base_scene.hpp"
#include "component/file_dialog.hpp"
#include "core/doubly_linked_list.hpp"
#include "core/stack.hpp"
#include "gui/stack_gui.hpp"
```

```
#include "raygui.h"
```

Include dependency graph for stack_scene.hpp:



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



Classes

- class scene::StackScene

  The stack scene.

Namespaces

- namespace scene

7.108 stack_scene.hpp

Go to the documentation of this file.

```
00001 #ifndef SCENE_STACK_SCENE_HPP_
00002 #define SCENE_STACK_SCENE_HPP_
00003
00004 #include "base_scene.hpp"
00005 #include "component/file_dialog.hpp"
00006 #include "core/doubly_linked_list.hpp"
00007 #include "core/stack.hpp"
00008 #include "gui/stack_gui.hpp"
00009 #include "raygui.h"
00010
00011 namespace scene {
00012
00017 class StackScene : public internal::BaseScene {
00018 public:
00022     void render() override;
00023
00027     void interact() override;
00028
00029 private:
00033     internal::SceneOptions scene_options{
```

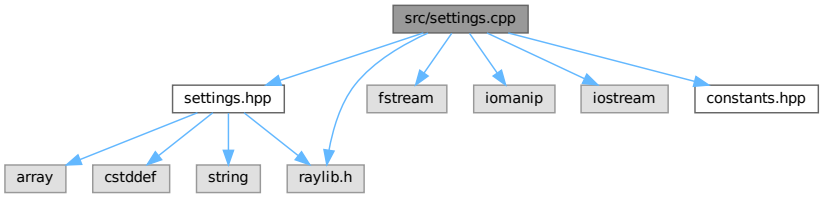
```
00034         // max_size
00035         8,  // NOLINT
00036
00037         // mode_labels
00038         "Mode: Create;"
00039         "Mode: Push;"
00040         "Mode: Pop;"
00041         "Mode: Clear",
00042
00043         // mode_selection
00044         0,
00045
00046         // action_labels
00047         {
00048             // Mode: Create
00049             "Action: Random;"
00050             "Action: Input;"
00051             "Action: File",
00052
00053             // Mode: Push
00054             "",
00055
00056             // Mode: Pop
00057             "",
00058
00059             // Mode: Clear
00060             "",
00061         },
00062
00063         // action_selection
00064         core::DoublyLinkedList<int>{0, 0, 0, 0},
00065     };
00066
00067     using internal::BaseScene::button_size;
00068     using internal::BaseScene::head_offset;
00069     using internal::BaseScene::options_head;
00070
00074     gui::GuiStack<int> m_stack{
00075         gui::GuiNode<int>{1},
00076         gui::GuiNode<int>{2},
00077         gui::GuiNode<int>{3},
00078     };
00079
00083     core::DoublyLinkedList<gui::GuiStack<int>> m_sequence;
00084
00088     bool m_go{};
00089     using internal::BaseScene::m_code_highlighter;
00090     using internal::BaseScene::m_file_dialog;
00091     using internal::BaseScene::m_sequence_controller;
00092     using internal::BaseScene::m_text_input;
00093
00094     using internal::BaseScene::render_go_button;
00095     using internal::BaseScene::render_options;
00096
00100     void render_inputs() override;
00101
00105     void interact_random();
00106
00110     void interact_import(core::Deque<int> nums);
00111
00115     void interact_push();
00116
00120     void interact_pop();
00121
00125     void interact_file_import();
00126
00130     void interact_clear();
00131 };
00132
00133 }  // namespace scene
00134
00135 #endif  // SCENE_STACK_SCENE_HPP_
```

7.109 src/settings.cpp File Reference

```
#include "settings.hpp"
#include <fstream>
#include <iomanip>
#include <iostream>
```

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

Include dependency graph for settings.cpp:



7.110 settings.cpp

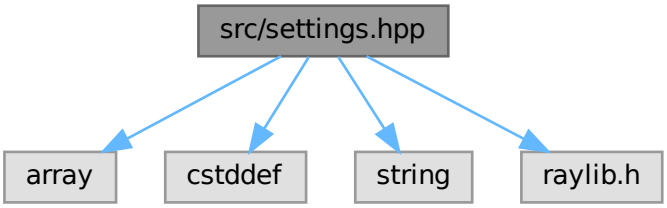
Go to the documentation of this file.

```
00001 #include "settings.hpp"
00002
00003 #include <fstream>
00004 #include <iomanip>
00005 #include <iostream>
00006
00007 #include "constants.hpp"
00008 #include "raylib.h"
00009
00010 Settings& Settings::get_instance() {
00011     static Settings settings;
00012     return settings;
00013 }
00014
00015 void Settings::save_to_file(const std::string& path) {
00016     std::ofstream file_out(path, std::ios::binary);
00017
00018     for (auto i = 0; i < num_color; ++i) {
00019         unsigned value = ColorToInt(m_colors.at(i));
00020         file_out.write(reinterpret_cast<const char*>(&value), sizeof(value));
00021     }
00022 }
00023
00024 Settings::~Settings() { save_to_file(constants::default_color_path); }
00025
00026 Color& Settings::get_color(std::size_t index) { return m_colors.at(index); }
00027
00028 Color Settings::get_color(std::size_t index) const {
00029     return m_colors.at(index);
00030 }
```

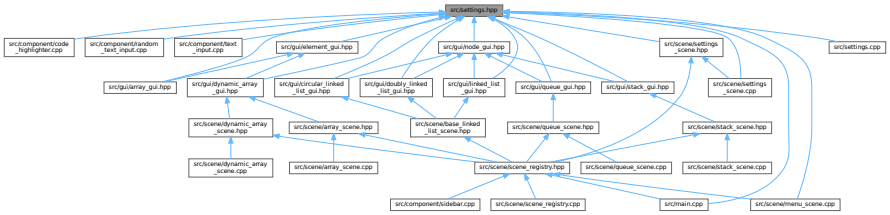
7.111 src/settings.hpp File Reference

```
#include <array>
#include <cstddef>
#include <string>
#include "raylib.h"
```

Include dependency graph for settings.hpp:



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



Classes

- class Settings

  The settings.

7.112 settings.hpp

Go to the documentation of this file.

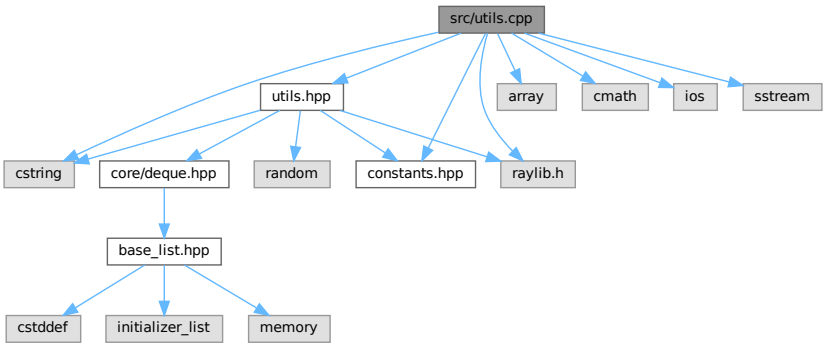
```
00001 #ifndef SETTINGS_HPP_
00002 #define SETTINGS_HPP_
00003
00004 #include <array>
00005 #include <cstddef>
00006 #include <string>
00007
00008 #include "raylib.h"
00009
00013 class Settings {
00014 public:
00018     static constexpr int num_color = 9;
00019
00023     static constexpr std::array<unsigned, num_color> default_color{{
00024         0x00000000,
00025         0x82828200,
00026         0xffa10000,
00027         0x00e43000,
00028         0x873cbe00,
00029         0xe6293700,
00030         0x0079f100,
00031         0xff6dc200,
00032         0xf5f5f500,
00033     }};
```

```
00034
00038     Settings(const Settings&) = delete;
00039
00043     Settings(Settings&&) = delete;
00044
00048     Settings& operator=(const Settings&) = delete;
00049
00053     Settings& operator=(Settings&&) = delete;
00054
00058     ~Settings();
00059
00064     static Settings& get_instance();
00065
00071     Color& get_color(std::size_t index);
00072
00078     Color get_color(std::size_t index) const;
00079
00084     void save_to_file(const std::string& path);
00085
00086 private:
00090     Settings() = default;
00091
00095     std::array<Color, num_color> m_colors{};
00096 };
00097
00098 #endif  // SETTINGS_HPP_
```

7.113 src/utils.cpp File Reference

```
#include "utils.hpp"
#include <array>
#include <cmath>
#include <cstring>
#include <ios>
#include <sstream>
#include "constants.hpp"
#include "raylib.h"
```

Include dependency graph for utils.cpp:



Namespaces

- namespace utils

  The utility functions.

Functions

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

  Draws text with custom font size and spacing.
- Vector2 utils::MeasureText (const char *text, float font_size, float spacing)

  Measures the text with custom font size and spacing.
- core::Deque< int > utils::str_extract_data (char str[constants::text_buffer_size])

  Extracts integers from a string separated by commas.
- bool utils::val_in_range (int num)

  Checks if a value is in range [min_val, max_val].
- void utils::unreachable ()

  Tells the compiler that this branch is unreachable.
- char * utils::strtok (char *str, const char *delim, char **save_ptr)

  Splits a string into tokens.
- Color utils::color_from_hex (const std::string &hex)

  Converts a hex string to a color.
- Color utils::adaptive_text_color (Color bg_color)

  Returns the color of the text based on the background color.

7.114 utils.cpp

Go to the documentation of this file.

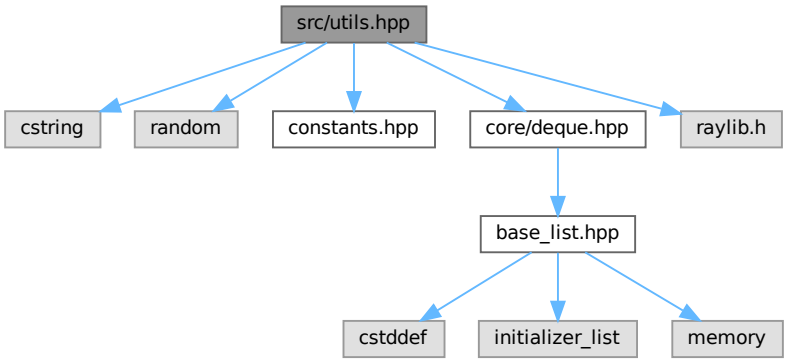
```
00001 #include "utils.hpp"
00002
00003 #include <array>
00004 #include <cmath>
00005 #include <cstring>
00006 #include <ios>
00007 #include <sstream>
00008
00009 #include "constants.hpp"
00010 #include "raylib.h"
00011
00012 namespace utils {
00013
00014 void DrawText(const char* text, Vector2 pos, Color color, float font_size,
00015               float spacing) {
00016     static Font font = LoadFontEx("data/open_sans.ttf",
00017                                   constants::default_font_size, nullptr, 0);
00018
00019     Vector2 pos_vec{static_cast<float>(pos.x), static_cast<float>(pos.y)};
00020     DrawTextEx(font, text, pos_vec, font_size, spacing, color);
00021 }
00022
00023 Vector2 MeasureText(const char* text, float font_size, float spacing) {
00024     static Font font = LoadFontEx("data/open_sans.ttf",
00025                                   constants::default_font_size, nullptr, 0);
00026
00027     return MeasureTextEx(font, text, font_size, spacing);
00028 }
00029
00030 core::Deque<int> str_extract_data(
00031     char str[constants::text_buffer_size]) {  // NOLINT
00032     char str_copy[constants::text_buffer_size];
00033     strncpy(str_copy, str, constants::text_buffer_size);
00034
00035     char* save_ptr = nullptr;
00036     char* token = utils::strtok(str_copy, ",", &save_ptr);
00037
00038     if (token == nullptr) {
00039         return {};
00040     }
00041
00042     core::Deque<int> ret;
00043
00044     constexpr int base = 10;
00045     int num = static_cast<int>(std::strtol(token, nullptr, base));
00046     ret.push_back(num);
00047
```

```
00048     while (true) {
00049         token = utils::strtok(nullptr, ",", &save_ptr);
00050         if (token == nullptr) {
00051             break;
00052         }
00053
00054         num = static_cast<int>(std::strtol(token, nullptr, base));
00055         ret.push_back(num);
00056     }
00057
00058     return ret;
00059 }
00060
00061 bool val_in_range(int num) {
00062     return constants::min_val <= num && num <= constants::max_val;
00063 }
00064
00065 void unreachable() {
00066 #if defined(_MSC_VER)
00067     __assume(0);
00068 #else
00069     __builtin_unreachable();
00070 #endif
00071 }
00072
00073 char* strtok(char* str, const char* delim, char** save_ptr) {
00074     return
00075 #if defined(_MSC_VER)
00076         strtok_s(str, delim, save_ptr);
00077 #else
00078         strtok_r(str, delim, save_ptr);
00079 #endif
00080 }
00081
00082 Color color_from_hex(const std::string& hex) {
00083     std::stringstream stream(hex + "ff");
00084     unsigned int value;
00085     stream » std::hex » value;
00086     return GetColor(value);
00087 }
00088
00089 // https://stackoverflow.com/a/3943023
00090 Color adaptive_text_color(Color bg_color) {
00091     constexpr std::array<float, 3> threshold{{0.2126, 0.7152, 0.0722}};
00092     const std::array<int, 3> colors = {{bg_color.r, bg_color.g, bg_color.b}};
00093     float sum = 0;
00094
00095     for (auto i = 0; i < 3; ++i) {
00096         float value = (float)colors.at(i) / 255.0F;
00097         if (value <= 0.04045) {
00098             value /= 12.92;
00099         } else {
00100             value = std::pow(((value + 0.055) / 1.055), 2.4);
00101         }
00102
00103         sum += value;
00104     }
00105
00106     return (sum > 0.179) ? BLACK : WHITE;
00107 }
00108
00109 }  // namespace utils
```

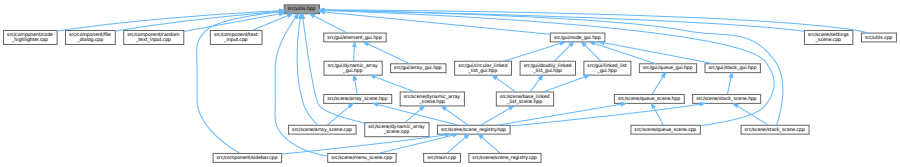
7.115 src/utils.hpp File Reference

```
#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

  The utility functions.

Functions

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

  Draws text with custom font size and spacing.
- Vector2 utils::MeasureText (const char *text, float font_size, float spacing)

  Measures the text with custom font size and spacing.
- template<typename T >
  T utils::get_random (T low, T high)

  Get a random number in the range [low, high].
- core::Deque< int > utils::str_extract_data (char str[constants::text_buffer_size])

  Extracts integers from a string separated by commas.
- bool utils::val_in_range (int num)

  Checks if a value is in range [min_val, max_val].
- void utils::unreachable ()

  Tells the compiler that this branch is unreachable.
- char * utils::strtok (char *str, const char *delim, char **save_ptr)

  Splits a string into tokens.
- Color utils::color_from_hex (const std::string &hex)

  Converts a hex string to a color.
- Color utils::adaptive_text_color (Color bg_color)

  Returns the color of the text based on the background color.

7.116 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
00015 namespace utils {
00016
00026 void DrawText(const char* text, Vector2 pos, Color color, float font_size,
00027               float spacing);
00028
00037 Vector2 MeasureText(const char* text, float font_size, float spacing);
00038
00047 template<typename T>
00048 T get_random(T low, T high) {
00049     if (low > high) {
00050         return low;
00051     }
00052
00053     static std::random_device ran_dev;
00054     static std::mt19937 prng(ran_dev());
00055     std::uniform_int_distribution<T> dist{low, high};
00056     return dist(prng);
00057 }
00058
00065 core::Deque<int> str_extract_data(
00066     char str[constants::text_buffer_size]);  // NOLINT
00067
00075 bool val_in_range(int num);
00076
00080 void unreachable();
00081
00090 char* strtok(char* str, const char* delim, char** save_ptr);
00091
00096 Color color_from_hex(const std::string& hex);
00097
00104 Color adaptive_text_color(Color bg_color);
00105
00106 }  // namespace utils
00107
00108 #endif  // UTILS_HPP_
```

Index