

RTRS

Generated by Doxygen 1.9.8



<b>1 Class Index</b>	<b>1</b>
1.1 Class List	1
<b>2 File Index</b>	<b>3</b>
2.1 File List	3
<b>3 Class Documentation</b>	<b>5</b>
3.1 Table Class Reference	5
3.1.1 Detailed Description	8
3.1.2 Constructor & Destructor Documentation	8
3.1.2.1 Table() [1/2]	8
3.1.2.2 Table() [2/2]	8
3.1.3 Member Function Documentation	9
3.1.3.1 changeMainData()	9
3.1.3.2 free()	9
3.1.3.3 getHour()	9
3.1.3.4 getMinute()	9
3.1.3.5 getName()	9
3.1.3.6 getPhoneNumber()	10
3.1.3.7 getSize()	10
3.1.3.8 getState()	10
3.1.3.9 getX()	10
3.1.3.10 getY()	10
3.1.3.11 occupy()	10
3.1.3.12 operator=()	10
3.1.3.13 reserve()	11
3.1.3.14 setAll()	11
3.1.3.15 test()	11
3.1.3.16 timeCheck()	11
3.1.4 Member Data Documentation	12
3.1.4.1 hour	12
3.1.4.2 minute	12
3.1.4.3 name	12
3.1.4.4 phoneNumber	12
3.1.4.5 size	12
3.1.4.6 state	12
3.1.4.7 x	12
3.1.4.8 y	13
3.2 TableHandler Class Reference	13
3.2.1 Detailed Description	15
3.2.2 Constructor & Destructor Documentation	16
3.2.2.1 TableHandler()	16
3.2.2.2 ~TableHandler()	16

3.2.3 Member Function Documentation	16
3.2.3.1 createNewTable()	16
3.2.3.2 deleteTable()	16
3.2.3.3 draw()	17
3.2.3.4 getActiveID()	17
3.2.3.5 getActiveIndexRef()	17
3.2.3.6 loadData()	17
3.2.3.7 loadTextures()	17
3.2.3.8 onClick()	17
3.2.3.9 saveData()	18
3.2.3.10 setActiveID()	18
3.2.3.11 test()	18
3.2.3.12 unloadTextures()	18
3.2.3.13 update()	18
3.2.4 Member Data Documentation	19
3.2.4.1 activeID	19
3.2.4.2 amount	19
3.2.4.3 saveFileName	19
3.2.4.4 tableFreeTexture	19
3.2.4.5 tableOccupiedTexture	19
3.2.4.6 tableReservedLateTexture	19
3.2.4.7 tableReservedTexture	20
3.2.4.8 tables	20
3.3 UI Class Reference	20
3.3.1 Detailed Description	22
3.3.2 Constructor & Destructor Documentation	23
3.3.2.1 UI()	23
3.3.3 Member Function Documentation	23
3.3.3.1 draw()	23
3.3.3.2 loadTextures()	23
3.3.3.3 unloadTextures()	23
3.3.4 Member Data Documentation	23
3.3.4.1 bg	23
3.3.4.2 deltaID	24
3.3.4.3 entryHourActive	24
3.3.4.4 entryHourValue	24
3.3.4.5 entryMinuteActive	24
3.3.4.6 entryMinuteValue	24
3.3.4.7 entryNameActive	24
3.3.4.8 entryNameValue	24
3.3.4.9 entryPhoneActive	24
3.3.4.10 entryPhoneValue	25

3.3.4.11 entrySizeActive . . . . .	25
3.3.4.12 entrySizeValue . . . . .	25
3.3.4.13 entryXActive . . . . .	25
3.3.4.14 entryXValue . . . . .	25
3.3.4.15 entryYActive . . . . .	25
3.3.4.16 entryYValue . . . . .	25
3.3.4.17 tableConfigWindowActive . . . . .	25
3.3.4.18 tables . . . . .	26
3.3.4.19 tableState . . . . .	26
<b>4 File Documentation</b>	<b>27</b>
4.1 include/tableHandler.h File Reference . . . . .	27
4.2 tableHandler.h . . . . .	28
4.3 include/tables.h File Reference . . . . .	29
4.4 tables.h . . . . .	30
4.5 include/ui.h File Reference . . . . .	31
4.6 ui.h . . . . .	32
4.7 src/main.cpp File Reference . . . . .	32
4.7.1 Detailed Description . . . . .	33
4.7.2 Macro Definition Documentation . . . . .	33
4.7.2.1 RAYGUI_STATIC . . . . .	33
4.7.3 Function Documentation . . . . .	34
4.7.3.1 main() . . . . .	34
4.8 src/tableHandler.cpp File Reference . . . . .	34
4.8.1 Detailed Description . . . . .	34
4.9 src/tables.cpp File Reference . . . . .	35
4.9.1 Detailed Description . . . . .	35
4.10 src/ui.cpp File Reference . . . . .	35
4.10.1 Macro Definition Documentation . . . . .	36
4.10.1.1 RAYGUI_IMPLEMENTATION . . . . .	36
4.10.1.2 RAYGUI_NO_RICONS . . . . .	36
<b>Index</b>	<b>37</b>



# Chapter 1

## Class Index

### 1.1 Class List

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

<a href="#">Table</a>	Represents a table in a restaurant with reservation and state information . . . . .	<a href="#">5</a>
<a href="#">TableHandler</a>	Manages a collection of restaurant tables and their interactions . . . . .	<a href="#">13</a>
<a href="#">UI</a>	Handles the graphical user interface for the restaurant table management system . . . . .	<a href="#">20</a>





# Chapter 2

## File Index

### 2.1 File List

Here is a list of all files with brief descriptions:

<a href="#">include/tableHandler.h</a>	27
<a href="#">include/tables.h</a>	29
<a href="#">include/ui.h</a>	31
<a href="#">src/main.cpp</a>	
Entry point for the Restaurant <a href="#">Table</a> Reservation System (RTRS)	32
<a href="#">src/tableHandler.cpp</a>	
Manages a collection of <a href="#">Table</a> objects, including their creation, deletion, serialization, and rendering	34
<a href="#">src/tables.cpp</a>	
Implementation of the <a href="#">Table</a> class for managing individual restaurant tables and their reservations	35
<a href="#">src/ui.cpp</a>	35



## Chapter 3

# Class Documentation

### 3.1 Table Class Reference

Represents a table in a restaurant with reservation and state information.

```
#include <tables.h>
```

Collaboration diagram for Table:

Table
<ul style="list-style-type: none"> <li>- int x</li> <li>- int y</li> <li>- short int state</li> <li>- short int size</li> <li>- std::string name</li> <li>- std::string phoneNumber</li> <li>- short int hour</li> <li>- short int minute</li> </ul>
<ul style="list-style-type: none"> <li>+ Table()</li> <li>+ Table(int x, int y, short int state, short int size, const std::string &amp;name, const std::string &amp;phoneNumber, short int hour, short int minute)</li> <li>+ Table &amp; operator=(Table obj)</li> <li>+ void setAll(int x, int y, short int state, short int size, const std::string &amp;name, const std::string &amp;phoneNumber, short int hour, short int minute)</li> <li>+ void changeMainData(int newX, int newY, short int newSize)</li> <li>+ void reserve(const std::string &amp;clientName, const std::string &amp;clientPhoneNumber, short int reservationHour, short int reservationMinute)</li> <li>+ void occupy()</li> <li>+ void free()</li> <li>+ void timeCheck(short int currentHour, short int currentMinute)</li> <li>+ void test()</li> <li>+ short int getState()</li> <li>+ int getX()</li> <li>+ int getY()</li> <li>+ short int getSize()</li> <li>+ std::string getName()</li> <li>+ std::string getPhoneNumber()</li> <li>+ short int getHour()</li> <li>+ short int getMinute()</li> </ul>

## Public Member Functions

- [Table](#) ()  
*Default constructor.*
- [Table](#) (int [x](#), int [y](#), short int [state](#), short int [size](#), const std::string &[name](#), const std::string &[phoneNumber](#), short int [hour](#), short int [minute](#))  
*Parameterized constructor to initialize a table with full data.*

- **Table & operator=** (Table obj)  
*Assignment operator.*
- void **setAll** (int x, int y, short int state, short int size, const std::string &name, const std::string &phoneNumber, short int hour, short int minute)  
*Set all table data at once.*
- void **changeMainData** (int newX, int newY, short int newSize)  
*Change the table's position and size.*
- void **reserve** (const std::string &clientName, const std::string &clientPhoneNumber, short int reservationHour, short int reservationMinute)  
*Make a reservation for the table.*
- void **occupy** ()  
*Mark the table as occupied.*
- void **free** ()  
*Free the table, removing reservation and occupancy state.*
- void **timeCheck** (short int currentHour, short int currentMinute)  
*Check if a reservation is late based on the current time.*
- void **test** ()  
*Test function (used for debug or placeholder).*
- short int **getState** ()  
*Get the current state of the table.*
- int **getX** ()  
*Get the X-coordinate of the table.*
- int **getY** ()  
*Get the Y-coordinate of the table.*
- short int **getSize** ()  
*Get the maximum number of people allowed at the table.*
- std::string **getName** ()  
*Get the name of the reservation holder.*
- std::string **getPhoneNumber** ()  
*Get the phone number of the reservation holder.*
- short int **getHour** ()  
*Get the hour of the reservation.*
- short int **getMinute** ()  
*Get the minute of the reservation.*

### Private Attributes

- int x  
*X-coordinate of the table on the layout.*
- int y  
*Y-coordinate of the table on the layout.*
- short int state  
*Current state of the table: free, occupied, reserved, or late reservation.*
- short int size  
*Maximum number of people the table can accommodate.*
- std::string name  
*Name of the person who reserved the table.*
- std::string phoneNumber  
*Phone number of the person who reserved the table.*
- short int hour  
*Reservation hour (0–23)*
- short int minute  
*Reservation minute (0–59)*

### 3.1.1 Detailed Description

Represents a table in a restaurant with reservation and state information.

The [Table](#) class encapsulates properties related to a restaurant table, including its position, state (e.g., free, reserved, occupied), and reservation details.

### 3.1.2 Constructor & Destructor Documentation

#### 3.1.2.1 [Table\(\)](#) [1/2]

```
Table::Table ( )
```

Default constructor.

Default constructor for a [Table](#).

Initializes the table with default values indicating it's unused or unconfigured.

#### 3.1.2.2 [Table\(\)](#) [2/2]

```
Table::Table (
    int x,
    int y,
    short int state,
    short int size,
    const std::string & name,
    const std::string & phoneNumber,
    short int hour,
    short int minute )
```

Parameterized constructor to initialize a table with full data.

Parameterized constructor for a [Table](#).

#### Parameters

<i>x</i>	X-coordinate.
<i>y</i>	Y-coordinate.
<i>state</i>	<a href="#">Table</a> state.
<i>size</i>	Number of people the table supports.
<i>name</i>	Reservation name.
<i>phoneNumber</i>	Reservation phone number.
<i>hour</i>	Reservation hour.
<i>minute</i>	Reservation minute.
<i>x</i>	<a href="#">Table</a> 's X position.
<i>y</i>	<a href="#">Table</a> 's Y position.
<i>state</i>	Initial state (0: free, 1: reserved, etc.).
<i>size</i>	Number of people the table can seat.
<i>name</i>	Reservation holder's name.
<i>phoneNumber</i>	Reservation holder's phone number.
<i>hour</i>	Reservation hour.
<i>minute</i>	Reservation minute.

### 3.1.3 Member Function Documentation

#### 3.1.3.1 `changeMainData()`

```
void Table::changeMainData (
    int newX,
    int newY,
    short int newSize )
```

Change the table's position and size.

Changes only the main physical data of the table (position and size).

#### 3.1.3.2 `free()`

```
void Table::free ( )
```

Free the table, removing reservation and occupancy state.

Frees the table and clears any reservation details.

#### 3.1.3.3 `getHour()`

```
short int Table::getHour ( )
```

Get the hour of the reservation.

Returns the reservation hour.

#### 3.1.3.4 `getMinute()`

```
short int Table::getMinute ( )
```

Get the minute of the reservation.

Returns the reservation minute.

#### 3.1.3.5 `getName()`

```
std::string Table::getName ( )
```

Get the name of the reservation holder.

Returns the name on the reservation.

### 3.1.3.6 getPhoneNumber()

```
std::string Table::getPhoneNumber ( )
```

Get the phone number of the reservation holder.

Returns the reservation phone number.

### 3.1.3.7 getSize()

```
short int Table::getSize ( )
```

Get the maximum number of people allowed at the table.

Returns the size (seating capacity) of the table.

### 3.1.3.8 getState()

```
short int Table::getState ( )
```

Get the current state of the table.

Returns the current state of the table.

### 3.1.3.9 getX()

```
int Table::getX ( )
```

Get the X-coordinate of the table.

Returns the X position of the table.

### 3.1.3.10 getY()

```
int Table::getY ( )
```

Get the Y-coordinate of the table.

Returns the Y position of the table.

### 3.1.3.11 occupy()

```
void Table::occupy ( )
```

Mark the table as occupied.

Marks the table as currently occupied.

### 3.1.3.12 operator=()

```
Table & Table::operator= (
    Table obj )
```

Assignment operator.

Copy assignment operator using copy-swap idiom.



**Parameters**

<i>obj</i>	<a href="#">Table</a> to copy from.
------------	-------------------------------------

**Returns**

Reference to this table.

**3.1.3.13 reserve()**

```
void Table::reserve (
    const std::string & clientName,
    const std::string & clientPhoneNumber,
    short int reservationHour,
    short int reservationMinute )
```

Make a reservation for the table.

Marks the table as reserved and stores reservation details.

**3.1.3.14 setAll()**

```
void Table::setAll (
    int x,
    int y,
    short int state,
    short int size,
    const std::string & name,
    const std::string & phoneNumber,
    short int hour,
    short int minute )
```

Set all table data at once.

Sets all properties of the table at once.

**3.1.3.15 test()**

```
void Table::test ( )
```

Test function (used for debug or placeholder).

Debug method to print test output.

**3.1.3.16 timeCheck()**

```
void Table::timeCheck (
    short int currentHour,
    short int currentMinute )
```

Check if a reservation is late based on the current time.

Checks if a reserved table is late compared to the current time. Changes state to 2 if late.

### 3.1.4 Member Data Documentation

#### 3.1.4.1 hour

```
short int Table::hour [private]
```

Reservation hour (0–23)

#### 3.1.4.2 minute

```
short int Table::minute [private]
```

Reservation minute (0–59)

#### 3.1.4.3 name

```
std::string Table::name [private]
```

Name of the person who reserved the table.

#### 3.1.4.4 phoneNumber

```
std::string Table::phoneNumber [private]
```

Phone number of the person who reserved the table.

#### 3.1.4.5 size

```
short int Table::size [private]
```

Maximum number of people the table can accommodate.

#### 3.1.4.6 state

```
short int Table::state [private]
```

Current state of the table: free, occupied, reserved, or late reservation.

#### 3.1.4.7 x

```
int Table::x [private]
```

X-coordinate of the table on the layout.

### 3.1.4.8 y

```
int Table::y [private]
```

Y-coordinate of the table on the layout.

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

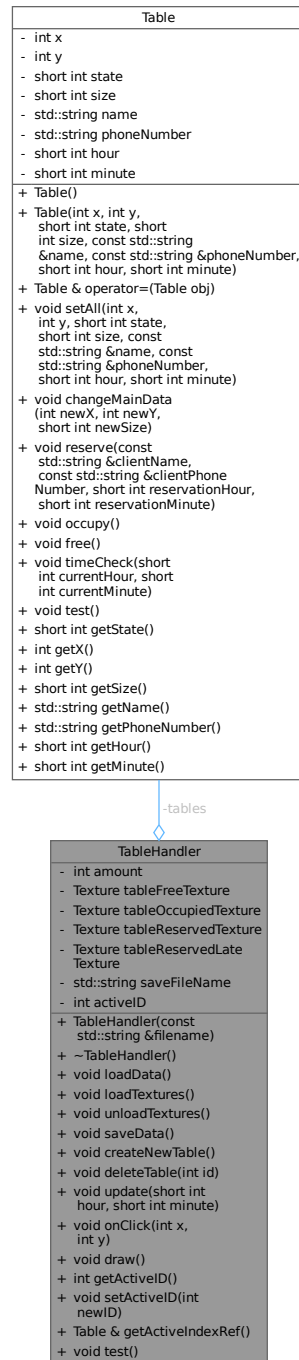
- [include/tables.h](#)
- [src/tables.cpp](#)

## 3.2 TableHandler Class Reference

Manages a collection of restaurant tables and their interactions.

```
#include <tableHandler.h>
```

Collaboration diagram for TableHandler:



## Public Member Functions

- [TableHandler](#) (const std::string &filename)  
*Constructor that initializes the handler with a file name for data persistence.*
- [~TableHandler](#) ()  
*Destructor to clean up resources.*
- void [loadData](#) ()

- Loads table data from the save file.*
  - void `loadTextures` ()
- Loads textures required for table visualization.*
  - void `unloadTextures` ()
- Unloads previously loaded textures.*
  - void `saveData` ()
- Saves current table data to the save file.*
  - void `createNewTable` ()
- Adds a new table to the collection.*
  - void `deleteTable` (int id)
- Deletes a table by its ID.*
  - void `update` (short int hour, short int minute)
- Updates the state of all tables based on the current time.*
  - void `onClick` (int x, int y)
- Handles mouse click interactions on the table layout.*
  - void `draw` ()
- Renders all tables and their states.*
  - int `getActiveID` ()
- Gets the ID of the currently active table.*
  - void `setActiveID` (int newID)
- Sets the currently active table ID.*
  - `Table & getActiveIndexRef` ()
- Gets a reference to the currently active table.*
  - void `test` ()
- Test function (for debug or development use).*

### Private Attributes

- `Table * tables`  
*Dynamic array of tables.*
- int `amount`  
*Number of tables currently managed.*
- Texture `tableFreeTexture`  
*Texture used to represent a free table.*
- Texture `tableOccupiedTexture`  
*Texture used to represent an occupied table.*
- Texture `tableReservedTexture`  
*Texture used to represent a reserved table.*
- Texture `tableReservedLateTexture`  
*Texture used to represent a reserved but late table.*
- std::string `saveFileName`  
*File path for saving/loading table data.*
- int `activeID`  
*ID of the currently selected/active table.*

### 3.2.1 Detailed Description

Manages a collection of restaurant tables and their interactions.

This class handles dynamic creation, deletion, updating, saving/loading, and rendering of table objects in a restaurant setting.

## 3.2.2 Constructor & Destructor Documentation

### 3.2.2.1 TableHandler()

```
TableHandler::TableHandler (
    const std::string & filename )
```

Constructor that initializes the handler with a file name for data persistence.

Constructs a [TableHandler](#) with a specified filename for saving/loading.

#### Parameters

<i>filename</i>	The file name used for loading and saving table data.
-----------------	---

### 3.2.2.2 ~TableHandler()

```
TableHandler::~~TableHandler ( )
```

Destructor to clean up resources.

Destructor that deallocates the table array if it's been initialized.

## 3.2.3 Member Function Documentation

### 3.2.3.1 createNewTable()

```
void TableHandler::createNewTable ( )
```

Adds a new table to the collection.

Adds a new [Table](#) to the dynamic table array.

### 3.2.3.2 deleteTable()

```
void TableHandler::deleteTable (
    int id )
```

Deletes a table by its ID.

Deletes a table by index.

#### Parameters

<i>id</i>	The ID/index of the table to delete.
<i>id</i>	Index of the table to delete.

### 3.2.3.3 draw()

```
void TableHandler::draw ( )
```

Renders all tables and their states.

Draws all tables using appropriate textures based on their state.

### 3.2.3.4 getActiveID()

```
int TableHandler::getActiveID ( )
```

Gets the ID of the currently active table.

Gets the index of the currently active table.

#### Returns

The active table's ID.

### 3.2.3.5 getActiveIndexRef()

```
Table & TableHandler::getActiveIndexRef ( )
```

Gets a reference to the currently active table.

Returns a reference to the currently active [Table](#). Returns the first table if no active ID is set.

#### Returns

Reference to the active table.

### 3.2.3.6 loadData()

```
void TableHandler::loadData ( )
```

Loads table data from the save file.

### 3.2.3.7 loadTextures()

```
void TableHandler::loadTextures ( )
```

Loads textures required for table visualization.

Loads table textures used for rendering.

### 3.2.3.8 onClick()

```
void TableHandler::onClick (
    int x,
    int y )
```

Handles mouse click interactions on the table layout.

Detects table selection based on mouse click coordinates.

**Parameters**

<i>x</i>	The X-coordinate of the click.
<i>y</i>	The Y-coordinate of the click.
<i>x</i>	Mouse X coordinate.
<i>y</i>	Mouse Y coordinate.

**3.2.3.9 saveData()**

```
void TableHandler::saveData ( )
```

Saves current table data to the save file.

**3.2.3.10 setActiveID()**

```
void TableHandler::setActiveID (
    int newID )
```

Sets the currently active table ID.

Sets the currently active table index.

**Parameters**

<i>newID</i>	The new active table ID.
--------------	--------------------------

**3.2.3.11 test()**

```
void TableHandler::test ( )
```

Test function (for debug or development use).

Prints all table data to standard output for debugging.

**3.2.3.12 unloadTextures()**

```
void TableHandler::unloadTextures ( )
```

Unloads previously loaded textures.

Unloads textures to free GPU memory.

**3.2.3.13 update()**

```
void TableHandler::update (
    short int hour,
    short int minute )
```

Updates the state of all tables based on the current time.



## Parameters

<i>hour</i>	The current hour.
<i>minute</i>	The current minute.
<i>hour</i>	Current hour.
<i>minute</i>	Current minute.

## 3.2.4 Member Data Documentation

### 3.2.4.1 activeID

```
int TableHandler::activeID [private]
```

ID of the currently selected/active table.

### 3.2.4.2 amount

```
int TableHandler::amount [private]
```

Number of tables currently managed.

### 3.2.4.3 saveFileName

```
std::string TableHandler::saveFileName [private]
```

File path for saving/loading table data.

### 3.2.4.4 tableFreeTexture

```
Texture TableHandler::tableFreeTexture [private]
```

Texture used to represent a free table.

### 3.2.4.5 tableOccupiedTexture

```
Texture TableHandler::tableOccupiedTexture [private]
```

Texture used to represent an occupied table.

### 3.2.4.6 tableReservedLateTexture

```
Texture TableHandler::tableReservedLateTexture [private]
```

Texture used to represent a reserved but late table.

### 3.2.4.7 tableReservedTexture

```
Texture TableHandler::tableReservedTexture [private]
```

Texture used to represent a reserved table.

### 3.2.4.8 tables

```
Table* TableHandler::tables [private]
```

Dynamic array of tables.

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

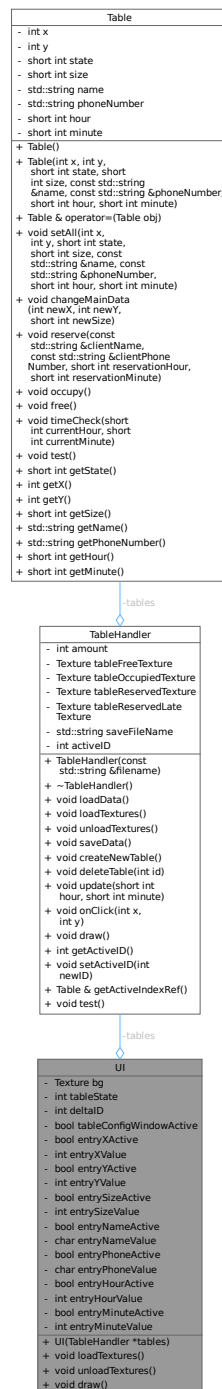
- [include/tableHandler.h](#)
- [src/tableHandler.cpp](#)

## 3.3 UI Class Reference

Handles the graphical user interface for the restaurant table management system.

```
#include <ui.h>
```

Collaboration diagram for UI:



## Public Member Functions

- **UI (TableHandler \*tables)**  
Constructs the **UI** object with a reference to the **TableHandler**.
- **void loadTextures ()**  
Loads textures required for the **UI**.
- **void unloadTextures ()**

- Unloads [UI](#) textures.
- void [draw](#) ()  
Renders the entire [UI](#) including background, table states, and input fields.

### Private Attributes

- Texture [bg](#)  
Background texture.
- [TableHandler](#) \* [tables](#)  
Pointer to the table handler.
- int [tableState](#)  
Current state of the selected table.
- int [deltaID](#)  
Used for table selection changes.
- bool [tableConfigWindowActive](#)  
Indicates if the table configuration window is active.
- bool [entryXActive](#)  
Is the X position input active.
- int [entryXValue](#)  
Value of the X position input.
- bool [entryYActive](#)  
Is the Y position input active.
- int [entryYValue](#)  
Value of the Y position input.
- bool [entrySizeActive](#)  
Is the size input active.
- int [entrySizeValue](#)  
Value of the size input.
- bool [entryNameActive](#)  
Is the name input active.
- char [entryNameValue](#) [128] = ""  
Value of the name input.
- bool [entryPhoneActive](#)  
Is the phone input active.
- char [entryPhoneValue](#) [128] = ""  
Value of the phone number input.
- bool [entryHourActive](#)  
Is the hour input active.
- int [entryHourValue](#)  
Value of the hour input.
- bool [entryMinuteActive](#)  
Is the minute input active.
- int [entryMinuteValue](#)  
Value of the minute input.

### 3.3.1 Detailed Description

Handles the graphical user interface for the restaurant table management system.

The [UI](#) class is responsible for rendering the background, interacting with the user, and providing input fields for configuring table data and reservations.

### 3.3.2 Constructor & Destructor Documentation

#### 3.3.2.1 UI()

```
UI::UI (
    TableHandler * tables )
```

Constructs the [UI](#) object with a reference to the [TableHandler](#).

Constructs the [UI](#) and initializes default values.

##### Parameters

<i>tables</i>	Pointer to the <a href="#">TableHandler</a> managing all tables.
---------------	--

### 3.3.3 Member Function Documentation

#### 3.3.3.1 draw()

```
void UI::draw ( )
```

Renders the entire [UI](#) including background, table states, and input fields.

Renders the entire [UI](#) frame, including buttons and table configuration window.

Displays the background, handles GUI input, and allows table creation, modification, deletion, and reservation.

#### 3.3.3.2 loadTextures()

```
void UI::loadTextures ( )
```

Loads textures required for the [UI](#).

Loads background texture and table textures.

#### 3.3.3.3 unloadTextures()

```
void UI::unloadTextures ( )
```

Unloads [UI](#) textures.

Unloads background and table textures to free memory.

### 3.3.4 Member Data Documentation

#### 3.3.4.1 bg

```
Texture UI::bg [private]
```

Background texture.

#### 3.3.4.2 `deltaID`

```
int UI::deltaID [private]
```

Used for table selection changes.

#### 3.3.4.3 `entryHourActive`

```
bool UI::entryHourActive [private]
```

Is the hour input active.

#### 3.3.4.4 `entryHourValue`

```
int UI::entryHourValue [private]
```

Value of the hour input.

#### 3.3.4.5 `entryMinuteActive`

```
bool UI::entryMinuteActive [private]
```

Is the minute input active.

#### 3.3.4.6 `entryMinuteValue`

```
int UI::entryMinuteValue [private]
```

Value of the minute input.

#### 3.3.4.7 `entryNameActive`

```
bool UI::entryNameActive [private]
```

Is the name input active.

#### 3.3.4.8 `entryNameValue`

```
char UI::entryNameValue[128] = "" [private]
```

Value of the name input.

#### 3.3.4.9 `entryPhoneActive`

```
bool UI::entryPhoneActive [private]
```

Is the phone input active.

#### 3.3.4.10 entryPhoneValue

```
char UI::entryPhoneValue[128] = "" [private]
```

Value of the phone number input.

#### 3.3.4.11 entrySizeActive

```
bool UI::entrySizeActive [private]
```

Is the size input active.

#### 3.3.4.12 entrySizeValue

```
int UI::entrySizeValue [private]
```

Value of the size input.

#### 3.3.4.13 entryXActive

```
bool UI::entryXActive [private]
```

Is the X position input active.

#### 3.3.4.14 entryXValue

```
int UI::entryXValue [private]
```

Value of the X position input.

#### 3.3.4.15 entryYActive

```
bool UI::entryYActive [private]
```

Is the Y position input active.

#### 3.3.4.16 entryYValue

```
int UI::entryYValue [private]
```

Value of the Y position input.

#### 3.3.4.17 tableConfigWindowActive

```
bool UI::tableConfigWindowActive [private]
```

Indicates if the table configuration window is active.

#### 3.3.4.18 tables

```
TableHandler* UI::tables [private]
```

Pointer to the table handler.

#### 3.3.4.19 tableState

```
int UI::tableState [private]
```

Current state of the selected table.

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

- [include/ui.h](#)
- [src/ui.cpp](#)



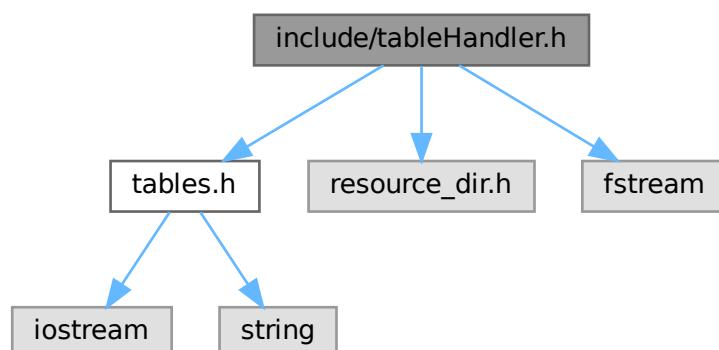
## Chapter 4

# File Documentation

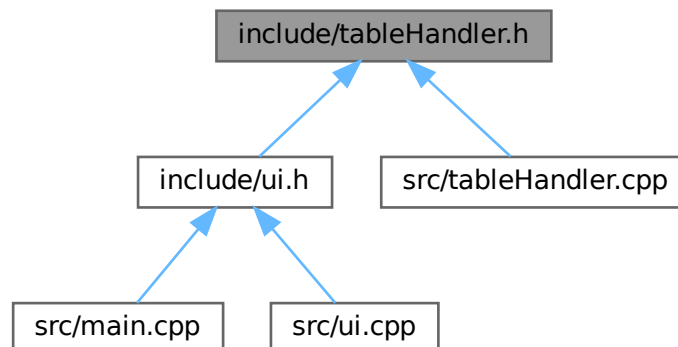
### 4.1 include/tableHandler.h File Reference

```
#include "tables.h"  
#include "resource_dir.h"  
#include <fstream>
```

Include dependency graph for tableHandler.h:



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



## Classes

- class [TableHandler](#)  
*Manages a collection of restaurant tables and their interactions.*

## 4.2 tableHandler.h

[Go to the documentation of this file.](#)

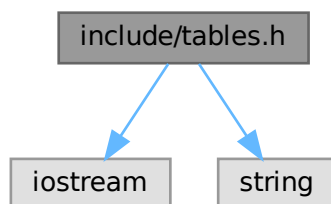
```

00001 #pragma once
00002
00003 #include "tables.h"
00004 #include "resource_dir.h"
00005 #include <fstream>
00006
00014 class TableHandler
00015 {
00016 private:
00017     Table* tables;
00018     int amount;
00019     Texture tableFreeTexture;
00020     Texture tableOccupiedTexture;
00021     Texture tableReservedTexture;
00022     Texture tableReservedLateTexture;
00023     std::string saveFileName;
00024     int activeID;
00025
00026 public:
00031     TableHandler(const std::string& filename);
00032
00036     ~TableHandler();
00037
00041     void loadData();
00042
00046     void loadTextures();
00047
00051     void unloadTextures();
00052
00056     void saveData();
00057
00061     void createNewTable();
00062
00067     void deleteTable(int id);
00068
00074     void update(short int hour, short int minute);
00075
  
```

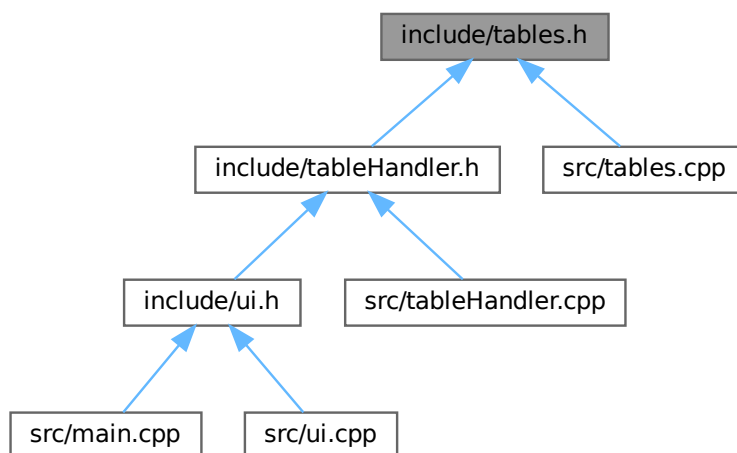
```
00081     void onClick(int x, int y);
00082
00086     void draw();
00087
00092     int getActiveID();
00093
00098     void setActiveID(int newID);
00099
00104     Table& getActiveIndexRef();
00105
00109     void test();
00110 };
00111
```

### 4.3 include/tables.h File Reference

```
#include <iostream>
#include <string>
Include dependency graph for tables.h:
```



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



## Classes

- class [Table](#)

*Represents a table in a restaurant with reservation and state information.*

## 4.4 tables.h

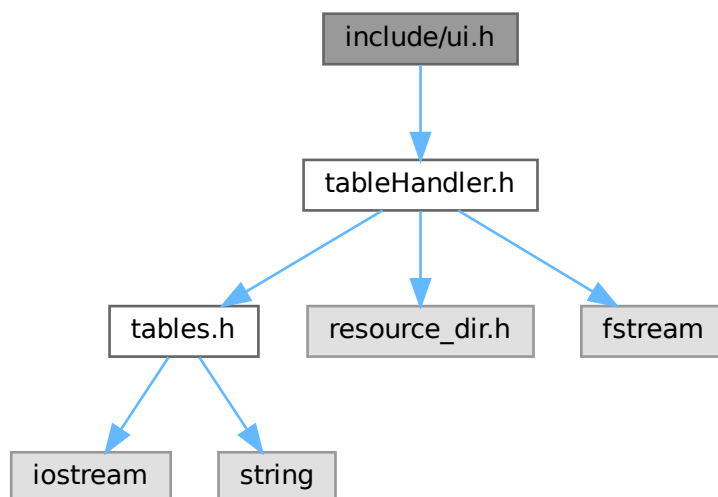
[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include <iostream>
00004 #include <string>
00005
00013 class Table
00014 {
00015 private:
00016     // GUI data
00017     int x;
00018     int y;
00019
00020     // Table properties
00021     short int state;
00022     short int size;
00023
00024     // Reservation data
00025     std::string name;
00026     std::string phoneNumber;
00027     short int hour;
00028     short int minute;
00029
00030 public:
00034     Table();
00035
00047     Table(int x, int y, short int state, short int size,
00048           const std::string& name, const std::string& phoneNumber,
00049           short int hour, short int minute);
00050
00056     Table& operator=(Table obj);
00057
00061     void setAll(int x, int y, short int state, short int size,
00062                const std::string& name, const std::string& phoneNumber,
00063                short int hour, short int minute);
00064
00068     void changeMainData(int newX, int newY, short int newSize);
00069
00073     void reserve(const std::string& clientName, const std::string& clientPhoneNumber,
00074                 short int reservationHour, short int reservationMinute);
00075
00079     void occupy();
00080
00084     void free();
00085
00089     void timeCheck(short int currentHour, short int currentMinute);
00090
00094     void test();
00095
00096     // Getters
00097
00101     short int getState();
00102
00106     int getX();
00107
00111     int getY();
00112
00116     short int getSize();
00117
00121     std::string getName();
00122
00126     std::string getPhoneNumber();
00127
00131     short int getHour();
00132
00136     short int getMinute();
00137 };
00138
```

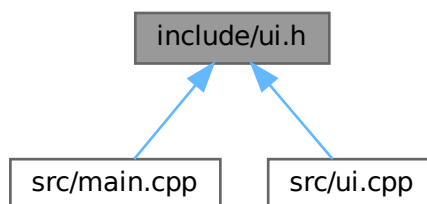
## 4.5 include/ui.h File Reference

```
#include "tableHandler.h"
```

Include dependency graph for ui.h:



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



### Classes

- class [UI](#)

*Handles the graphical user interface for the restaurant table management system.*

## 4.6 ui.h

[Go to the documentation of this file.](#)

```

00001 #pragma once
00002
00003 #include "tableHandler.h"
00004
00012 class UI
00013 {
00014 private:
00015     Texture bg;
00016     TableHandler* tables;
00017     int tableState;
00018     int deltaID;
00019     bool tableConfigWindowActive;
00020
00021     // Entry field state and values
00022     bool entryXActive;
00023     int entryXValue;
00024
00025     bool entryYActive;
00026     int entryYValue;
00027
00028     bool entrySizeActive;
00029     int entrySizeValue;
00030
00031     bool entryNameActive;
00032     char entryNameValue[128] = "";
00033
00034     bool entryPhoneActive;
00035     char entryPhoneValue[128] = "";
00036
00037     bool entryHourActive;
00038     int entryHourValue;
00039
00040     bool entryMinuteActive;
00041     int entryMinuteValue;
00042
00043 public:
00048     UI(TableHandler* tables);
00049
00053     void loadTextures();
00054
00058     void unloadTextures();
00059
00063     void draw();
00064 };
00065

```

## 4.7 src/main.cpp File Reference

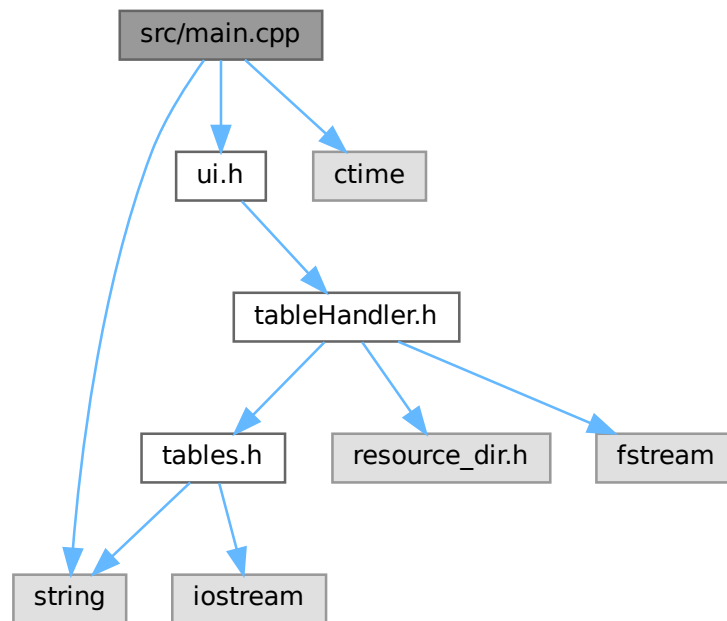
Entry point for the Restaurant [Table](#) Reservation System (RTRS).

```

#include <string>
#include "ui.h"
#include <ctime>

```

Include dependency graph for main.cpp:



## Macros

- `#define` [RAYGUI\\_STATIC](#)

## Functions

- `int` [main](#) ()  
*Application entry point.*

### 4.7.1 Detailed Description

Entry point for the Restaurant [Table](#) Reservation System (RTRS).

Initializes the window and core systems, loads data, and enters the main update/render loop. Delegates functionality to [TableHandler](#) and [UI](#) classes.

### 4.7.2 Macro Definition Documentation

#### 4.7.2.1 RAYGUI\_STATIC

```
#define RAYGUI_STATIC
```

### 4.7.3 Function Documentation

#### 4.7.3.1 main()

```
int main ( )
```

Application entry point.

Initializes the graphical window, loads saved table data, updates the table states over time, and handles rendering and input. Uses the [TableHandler](#) and [UI](#) classes to manage state and draw the interface.

#### Returns

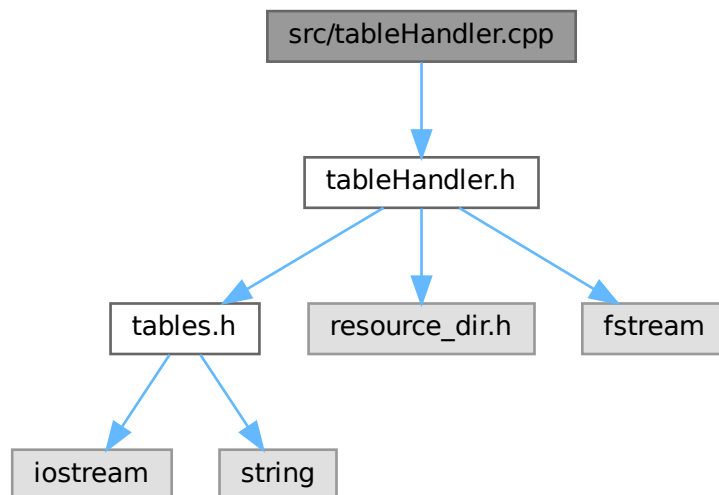
Exit code (0 if successful).

## 4.8 src/tableHandler.cpp File Reference

Manages a collection of [Table](#) objects, including their creation, deletion, serialization, and rendering.

```
#include "tableHandler.h"
```

Include dependency graph for tableHandler.cpp:



#### 4.8.1 Detailed Description

Manages a collection of [Table](#) objects, including their creation, deletion, serialization, and rendering.

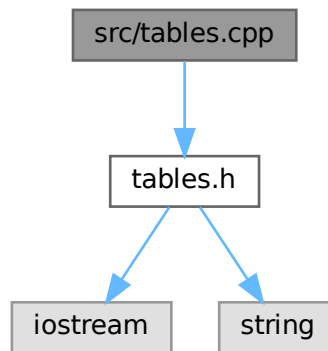


## 4.9 src/tables.cpp File Reference

Implementation of the [Table](#) class for managing individual restaurant tables and their reservations.

```
#include "tables.h"
```

Include dependency graph for tables.cpp:



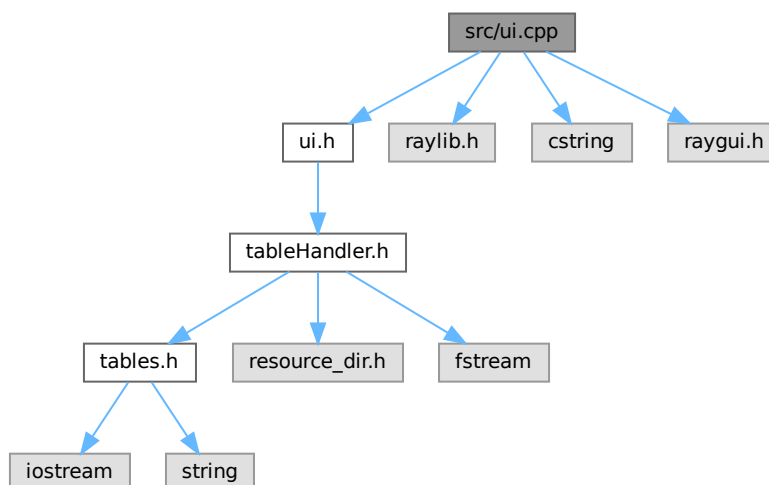
### 4.9.1 Detailed Description

Implementation of the [Table](#) class for managing individual restaurant tables and their reservations.

## 4.10 src/ui.cpp File Reference

```
#include "ui.h"  
#include "raylib.h"  
#include <cstring>  
#include "raygui.h"
```

Include dependency graph for ui.cpp:



## Macros

- `#define RAYGUI_NO_RICONS`
- `#define RAYGUI_IMPLEMENTATION`

## 4.10.1 Macro Definition Documentation

### 4.10.1.1 RAYGUI\_IMPLEMENTATION

```
#define RAYGUI_IMPLEMENTATION
```

### 4.10.1.2 RAYGUI\_NO\_RICONS

```
#define RAYGUI_NO_RICONS
```

# Index

- ~TableHandler
  - TableHandler, [16](#)
- activeID
  - TableHandler, [19](#)
- amount
  - TableHandler, [19](#)
- bg
  - UI, [23](#)
- changeMainData
  - Table, [9](#)
- createNewTable
  - TableHandler, [16](#)
- deleteTable
  - TableHandler, [16](#)
- deltaID
  - UI, [23](#)
- draw
  - TableHandler, [16](#)
  - UI, [23](#)
- entryHourActive
  - UI, [24](#)
- entryHourValue
  - UI, [24](#)
- entryMinuteActive
  - UI, [24](#)
- entryMinuteValue
  - UI, [24](#)
- entryNameActive
  - UI, [24](#)
- entryNameValue
  - UI, [24](#)
- entryPhoneActive
  - UI, [24](#)
- entryPhoneValue
  - UI, [24](#)
- entrySizeActive
  - UI, [25](#)
- entrySizeValue
  - UI, [25](#)
- entryXActive
  - UI, [25](#)
- entryXValue
  - UI, [25](#)
- entryYActive
  - UI, [25](#)
- entryYValue

- UI, [25](#)
- free
  - Table, [9](#)
- getActiveID
  - TableHandler, [17](#)
- getActiveIndexRef
  - TableHandler, [17](#)
- getHour
  - Table, [9](#)
- getMinute
  - Table, [9](#)
- getName
  - Table, [9](#)
- getPhoneNumber
  - Table, [9](#)
- getSize
  - Table, [10](#)
- getState
  - Table, [10](#)
- getX
  - Table, [10](#)
- getY
  - Table, [10](#)
- hour
  - Table, [12](#)
- include/tableHandler.h, [27](#), [28](#)
- include/tables.h, [29](#), [30](#)
- include/ui.h, [31](#), [32](#)
- loadData
  - TableHandler, [17](#)
- loadTextures
  - TableHandler, [17](#)
  - UI, [23](#)
- main
  - main.cpp, [34](#)
- main.cpp
  - main, [34](#)
  - RAYGUI\_STATIC, [33](#)
- minute
  - Table, [12](#)
- name
  - Table, [12](#)
- occupy

- Table, 10
- onClick
  - TableHandler, 17
- operator=
  - Table, 10
- phoneNumber
  - Table, 12
- RAYGUI\_IMPLEMENTATION
  - ui.cpp, 36
- RAYGUI\_NO\_RICONS
  - ui.cpp, 36
- RAYGUI\_STATIC
  - main.cpp, 33
- reserve
  - Table, 11
- saveData
  - TableHandler, 18
- saveFileName
  - TableHandler, 19
- setActiveID
  - TableHandler, 18
- setAll
  - Table, 11
- size
  - Table, 12
- src/main.cpp, 32
- src/tableHandler.cpp, 34
- src/tables.cpp, 35
- src/ui.cpp, 35
- state
  - Table, 12
- Table, 5
  - changeMainData, 9
  - free, 9
  - getHour, 9
  - getMinute, 9
  - getName, 9
  - getPhoneNumber, 9
  - getSize, 10
  - getState, 10
  - getX, 10
  - getY, 10
  - hour, 12
  - minute, 12
  - name, 12
  - occupy, 10
  - operator=, 10
  - phoneNumber, 12
  - reserve, 11
  - setAll, 11
  - size, 12
  - state, 12
  - Table, 8
  - test, 11
  - timeCheck, 11
  - x, 12
  - y, 12
- tableConfigWindowActive
  - UI, 25
- tableFreeTexture
  - TableHandler, 19
- TableHandler, 13
  - ~TableHandler, 16
  - activeID, 19
  - amount, 19
  - createNewTable, 16
  - deleteTable, 16
  - draw, 16
  - getActiveID, 17
  - getActiveIndexRef, 17
  - loadData, 17
  - loadTextures, 17
  - onClick, 17
  - saveData, 18
  - saveFileName, 19
  - setActiveID, 18
  - tableFreeTexture, 19
  - TableHandler, 16
  - tableOccupiedTexture, 19
  - tableReservedLateTexture, 19
  - tableReservedTexture, 19
  - tables, 20
  - test, 18
  - unloadTextures, 18
  - update, 18
- tableOccupiedTexture
  - TableHandler, 19
- tableReservedLateTexture
  - TableHandler, 19
- tableReservedTexture
  - TableHandler, 19
- tables
  - TableHandler, 20
  - UI, 25
- tableState
  - UI, 26
- test
  - Table, 11
  - TableHandler, 18
- timeCheck
  - Table, 11
- UI, 20
  - bg, 23
  - deltaID, 23
  - draw, 23
  - entryHourActive, 24
  - entryHourValue, 24
  - entryMinuteActive, 24
  - entryMinuteValue, 24
  - entryNameActive, 24
  - entryNameValue, 24
  - entryPhoneActive, 24
  - entryPhoneValue, 24

- entrySizeActive, [25](#)
- entrySizeValue, [25](#)
- entryXActive, [25](#)
- entryXValue, [25](#)
- entryYActive, [25](#)
- entryYValue, [25](#)
- loadTextures, [23](#)
- tableConfigWindowActive, [25](#)
- tables, [25](#)
- tableState, [26](#)
- UI, [23](#)
- unloadTextures, [23](#)
- ui.cpp
  - RAYGUI\_IMPLEMENTATION, [36](#)
  - RAYGUI\_NO\_RICONS, [36](#)
- unloadTextures
  - TableHandler, [18](#)
  - UI, [23](#)
- update
  - TableHandler, [18](#)
- x
  - Table, [12](#)
- y
  - Table, [12](#)