RTRS

Generated by Doxygen 1.9.8

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

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:

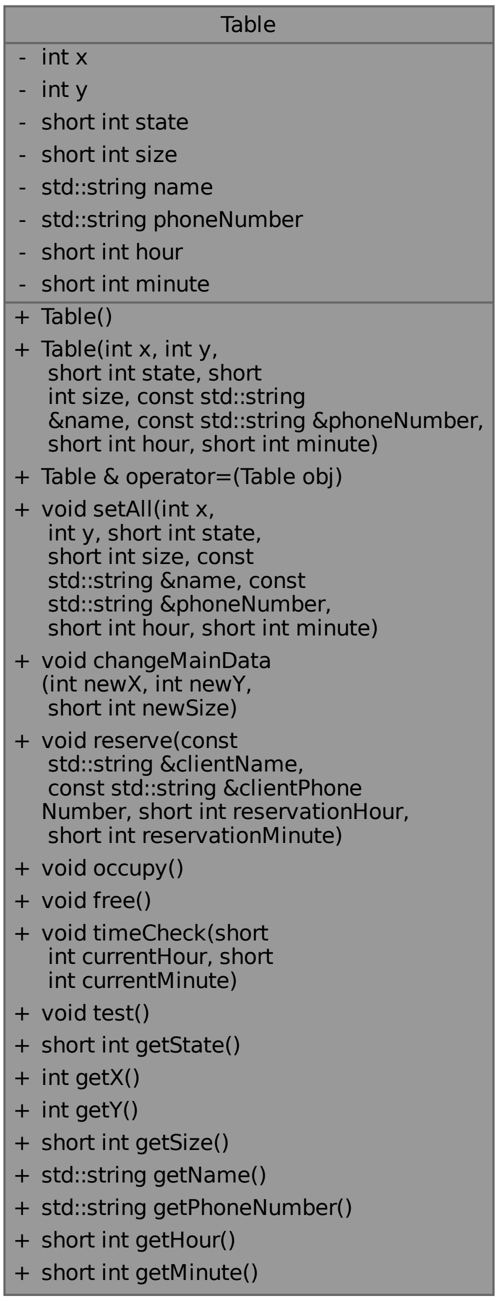


Table
- int x
- int y
- short int state
- short int size
- std::string name
- std::string phoneNumber
- short int hour
- short int minute
+ Table()
+ Table(int x, int y, short int state, short int size, const std::string &name, const std::string &phoneNumber, short int hour, short int minute)
+ Table & operator=(Table obj)
+ 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)
+ void changeMainData (int newX, int newY, short int newSize)
+ void reserve(const std::string &clientName, const std::string &clientPhone Number, short int reservationHour, short int reservationMinute)
+ void occupy()
+ void free()
+ void timeCheck(short int currentHour, short int currentMinute)
+ void test()
+ short int getState()
+ int getX()
+ int getY()
+ short int getSize()
+ std::string getName()
+ std::string getPhoneNumber()
+ short int getHour()
+ short int getMinute()

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

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

obj	Table 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:

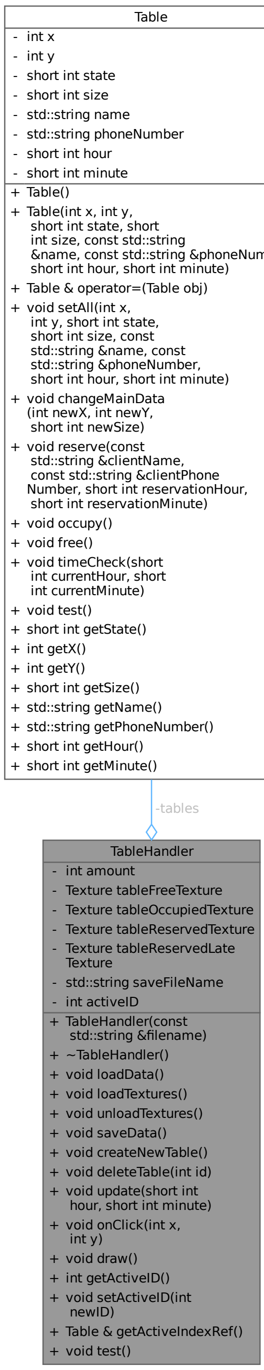


Table
- int x
- int y
- short int state
- short int size
- std::string name
- std::string phoneNumber
- short int hour
- short int minute
+ Table()
+ Table(int x, int y, short int state, short int size, const std::string &name, const std::string &phoneNumber, short int hour, short int minute)
+ Table & operator=(Table obj)
+ 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)
+ void changeMainData (int newX, int newY, short int newSize)
+ void reserve(const std::string &clientName, const std::string &clientPhone Number, short int reservationHour, short int reservationMinute)
+ void occupy()
+ void free()
+ void timeCheck(short int currentHour, short int currentMinute)
+ void test()
+ short int getState()
+ int getX()
+ int getY()
+ short int getSize()
+ std::string getName()
+ std::string getPhoneNumber()
+ short int getHour()
+ short int getMinute()
-tables
TableHandler
- int amount
- Texture tableFreeTexture
- Texture tableOccupiedTexture
- Texture tableReservedTexture
- Texture tableReservedLate Texture
- std::string saveFileName
- int activeID
+ TableHandler(const std::string &filename)
+ ~TableHandler()
+ void loadData()
+ void loadTextures()
+ void unloadTextures()
+ void saveData()
+ void createNewTable()
+ void deleteTable(int id)
+ void update(short int hour, short int minute)
+ void onClick(int x, int y)
+ void draw()
+ int getActiveID()
+ void setActiveID(int newID)
+ Table & getActiveIndexRef()
+ void test()

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

filename	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

id	The ID/index of the table to delete.
id	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

x	The X-coordinate of the click.
y	The Y-coordinate of the click.
x	Mouse X coordinate.
y	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

newID	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

hour	The current hour.
minute	The current minute.
hour	Current hour.
minute	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:

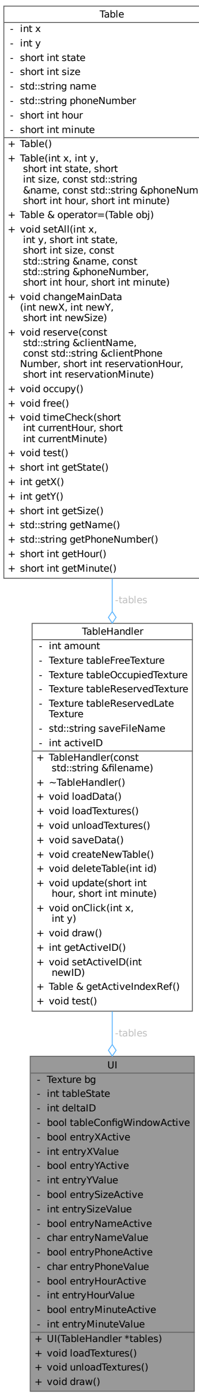


Table
- int x
- int y
- short int state
- short int size
- std::string name
- std::string phoneNumber
- short int hour
- short int minute
+ Table()
+ Table(int x, int y, short int state, short int size, const std::string &name, const std::string &phoneNumber, short int hour, short int minute)
+ Table & operator=(Table obj)
+ 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)
+ void changeMainData (int newX, int newY, short int newSize)
+ void reserve(const std::string &clientName, const std::string &clientPhoneNumber, short int reservationHour, short int reservationMinute)
+ void occupy()
+ void free()
+ void timeCheck(short int currentHour, short int currentMinute)
+ void test()
+ short int getState()
+ int getX()
+ int getY()
+ short int getSize()
+ std::string getName()
+ std::string getPhoneNumber()
+ short int getHour()
+ short int getMinute()
-tables
TableHandler
- int amount
- Texture tableFreeTexture
- Texture tableOccupiedTexture
- Texture tableReservedTexture
- Texture tableReservedLateTexture
- std::string saveFileName
- int activeID
+ TableHandler(const std::string &filename)
+ ~TableHandler()
+ void loadData()
+ void loadTextures()
+ void unloadTextures()
+ void saveData()
+ void createNewTable()
+ void deleteTable(int id)
+ void update(short int hour, short int minute)
+ void onClick(int x, int y)
+ void draw()
+ int getActiveID()
+ void setActiveID(int newID)
+ Table & getActiveIndexRef()
+ void test()
-tables
UI
- Texture bg
- int tableState
- int deltaID
- bool tableConfigWindowActive
- bool entryXActive
- int entryXValue
- bool entryYActive
- int entryYValue
- bool entrySizeActive
- int entrySizeValue
- bool entryNameActive
- char entryNameValue
- bool entryPhoneActive
- char entryPhoneValue
- bool entryHourActive
- int entryHourValue
- bool entryMinuteActive
- int entryMinuteValue
+ UI(TableHandler *tables)
+ void loadTextures()
+ void unloadTextures()
+ void draw()

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

tables	Pointer to the TableHandler 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

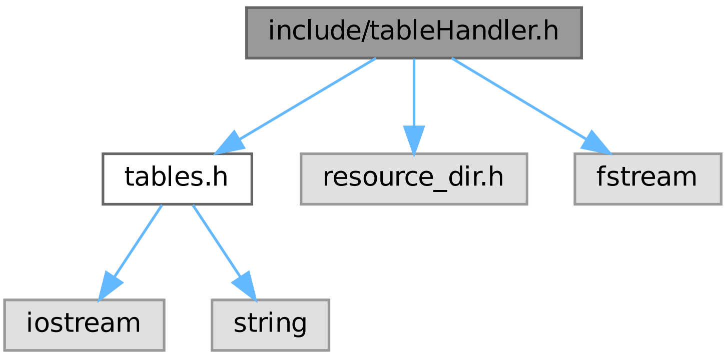
Chapter 4

File Documentation

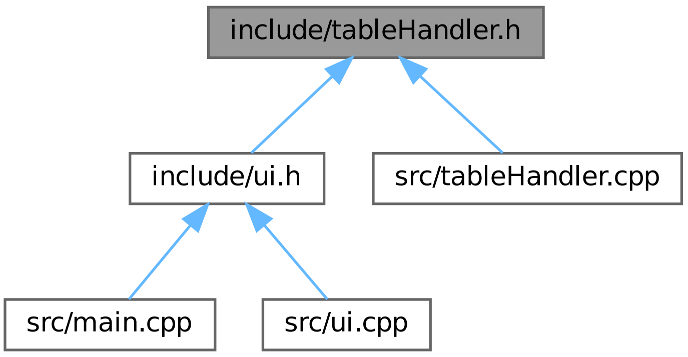
4.1 include/tableHandler.h File Reference

```
#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

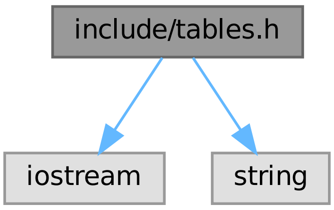
Go to the documentation of this file.

```
#pragma once

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

class TableHandler
{
private:
    Table* tables;
    int amount;
    Texture tableFreeTexture;
    Texture tableOccupiedTexture;
    Texture tableReservedTexture;
    Texture tableReservedLateTexture;
    std::string saveFileName;
    int activeID;

public:
    TableHandler(const std::string& filename);

    ~TableHandler();

    void loadData();

    void loadTextures();

    void unloadTextures();

    void saveData();

    void createNewTable();

    void deleteTable(int id);

    void update(short int hour, short int minute);

    void onClick(int x, int y);

    void draw();

    int getActiveID();

    void setActiveID(int newID);

    Table& getActiveIndexRef();

    void test();
};

```

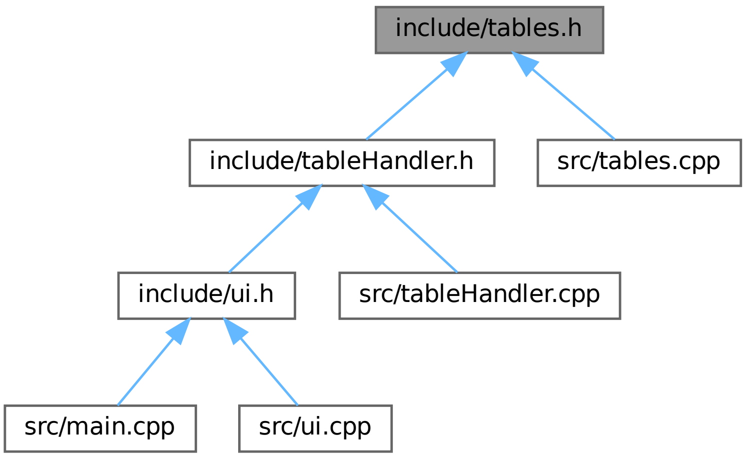
4.3 include/tables.h File Reference

```
#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

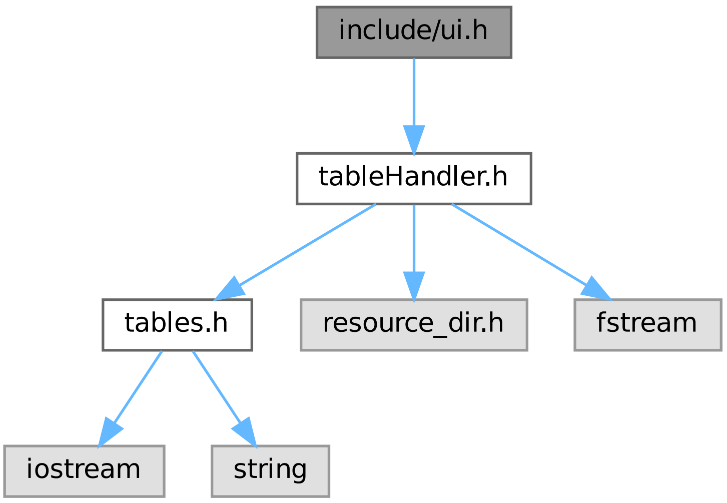
Go to the documentation of this file.

```
#pragma once

#include <iostream>
#include <string>

class Table
{
private:
    // GUI data
    int x;
    int y;

    // Table properties
    short int state;
    short int size;

    // Reservation data
    std::string name;
    std::string phoneNumber;
    short int hour;
    short int minute;

public:
    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);

    Table& operator=(Table obj);

    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);

    void changeMainData(int newX, int newY, short int newSize);

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

    void occupy();

    void free();

    void timeCheck(short int currentHour, short int currentMinute);

    void test();

    // Getters

    short int getState();

    int getX();

    int getY();

    short int getSize();

    std::string getName();

    std::string getPhoneNumber();

    short int getHour();

    short int getMinute();
};

```

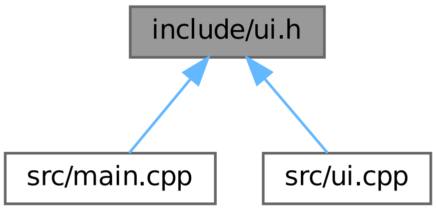
4.5 include/ui.h File Reference

```
#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.

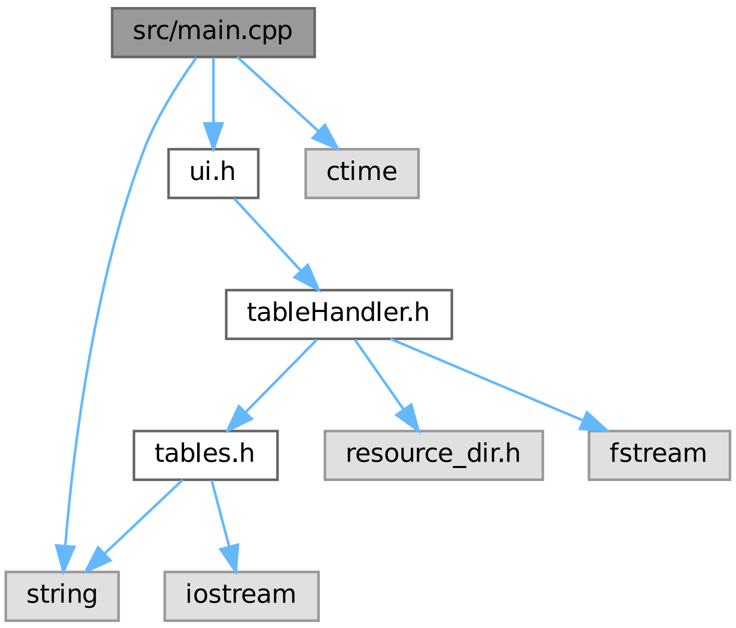
```
#pragma once

#include "tableHandler.h"

class UI
{
private:
    Texture bg;
    TableHandler* tables;
    int tableState;
    int deltaID;
    bool tableConfigWindowActive;

    // Entry field state and values
    bool entryXActive;
    int entryXValue;

    bool entryYActive;
    int entryYValue;

    bool entrySizeActive;
    int entrySizeValue;

    bool entryNameActive;
    char entryNameValue[128] = "";

    bool entryPhoneActive;
    char entryPhoneValue[128] = "";

    bool entryHourActive;
    int entryHourValue;

    bool entryMinuteActive;
    int entryMinuteValue;

public:
    UI(TableHandler* tables);

    void loadTextures();

    void unloadTextures();

    void draw();
};

```

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

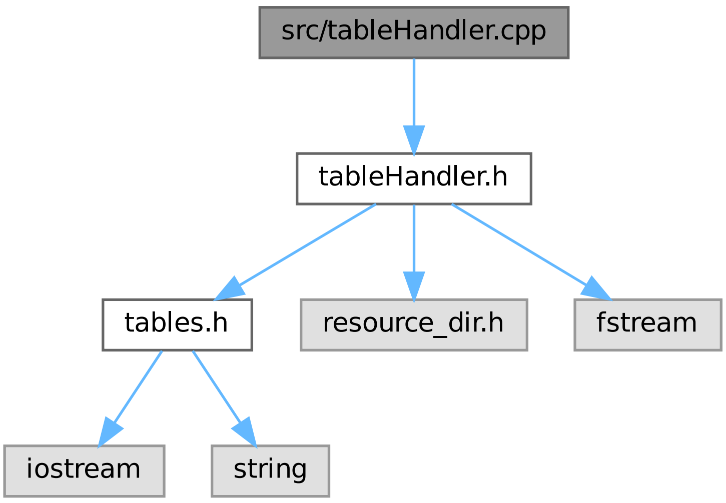
Exit code (0 if successful).

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

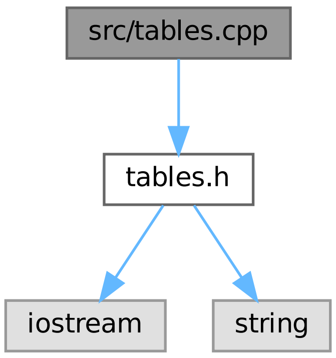
Manages a collection of Table objects, including their creation, deletion, serialization, and rendering.

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:



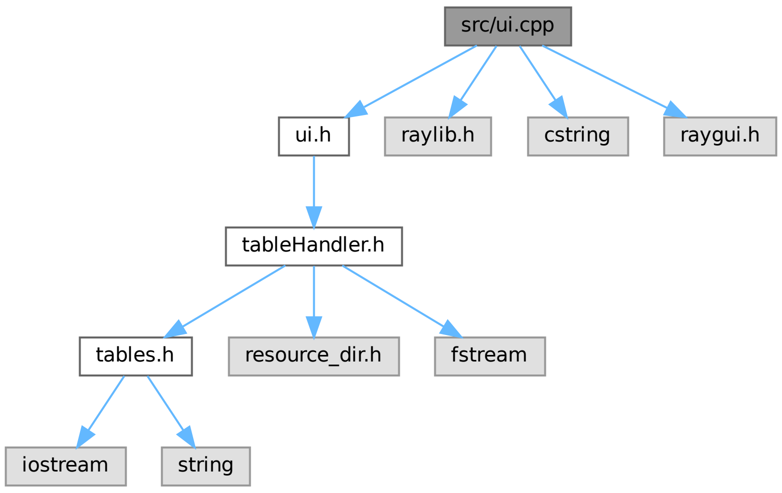
4.9.1 Detailed Description

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