SFML Game

Generated by Doxygen 1.8.17

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 AI Namespace Reference

Classes

- class PathNode
- class RTBPathGenerator

5.2 RiD Namespace Reference

Classes

- class AssetManager
- class AudioManager
- class ConfigurationLoader
- struct gameDat
- class Movement
- class RiDGame

Typedefs

- typedef std::shared_ptr< gameDat > gameDatReference

5.2.1 Typedef Documentation

5.2.1.1 gameDatReference

```
typedef std::shared_ptr<gameDat> RiD::gameDatReference
```

5.3 RTB Namespace Reference

Classes

- class Archer
- class Arrow
- class ArrowHitbox
- class Bot
- class Character
- class Hitbox
- class HPBar
- class MapElement
- class OrientedHitbox
- class Player
- class RealTimeBattle
- class SpearHitbox
- class Spearman
- class SwordHitbox
- class Swordsman
- class TileMap

5.4 RTBGUI Namespace Reference

Classes

- class BookButton
- class Button
- class Cursor
- class GUI
- class Menu
- class Message
- class Panel
- class WindowBorder

Chapter 6

Class Documentation

6.1 RiD::Movement::animationDuration Struct Reference

Public Attributes

- sf::Time walking_animation_start_time
- sf::Time walking_animation_frame_duration
- sf::Time sword_animation_start_time
- sf::Time sword_animation_frame_duration
- sf::Time bow_shot_animation_start_time
- sf::Time bow_shot_animation_frame_duration
- sf::Time death_animation_start_time
- sf::Time death_animation_frame_duration
- sf::Time spear_animation_start_time
- sf::Time spear_animation_frame_duration

6.1.1 Member Data Documentation

6.1.1.1 bow_shot_animation_frame_duration

```
sf::Time RiD::Movement::animationDuration::bow_shot_animation_frame_duration
```

6.1.1.2 bow_shot_animation_start_time

```
sf::Time RiD::Movement::animationDuration::bow_shot_animation_start_time
```

6.1.1.3 death_animation_frame_duration

```
sf::Time RiD::Movement::animationDuration::death_animation_frame_duration
```

6.1.1.4 death_animation_start_time

```
sf::Time RiD::Movement::animationDuration::death_animation_start_time
```

6.1.1.5 spear_animation_frame_duration

```
sf::Time RiD::Movement::animationDuration::spear_animation_frame_duration
```

6.1.1.6 spear_animation_start_time

```
sf::Time RiD::Movement::animationDuration::spear_animation_start_time
```

6.1.1.7 sword_animation_frame_duration

```
sf::Time RiD::Movement::animationDuration::sword_animation_frame_duration
```

6.1.1.8 sword_animation_start_time

```
sf::Time RiD::Movement::animationDuration::sword_animation_start_time
```

6.1.1.9 walking_animation_frame_duration

```
sf::Time RiD::Movement::animationDuration::walking_animation_frame_duration
```

6.1.1.10 walking_animation_start_time

```
sf::Time RiD::Movement::animationDuration::walking_animation_start_time
```

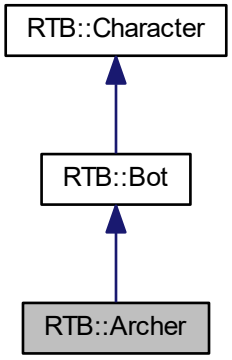
The documentation for this struct was generated from the following file:

- Movement.h

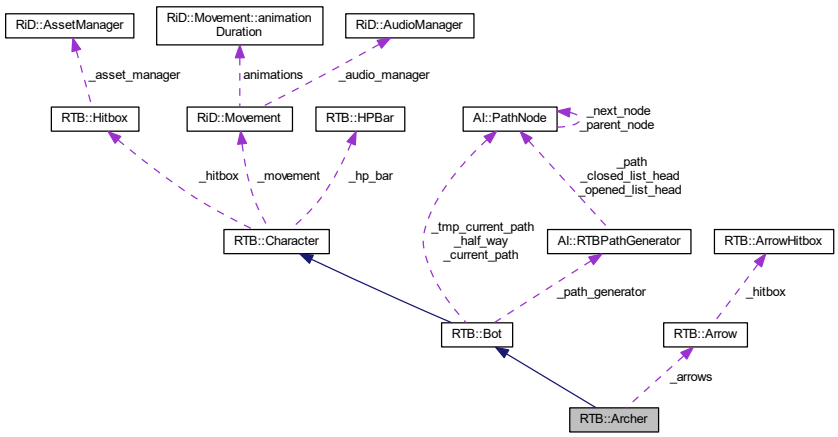
6.2 RTB::Archer Class Reference

```
#include <Archer.h>
```

Inheritance diagram for RTB::Archer:



Collaboration diagram for RTB::Archer:



Public Member Functions

- Archer (sf::Texture texture, short health_points, sf::Texture &arrow_texture, std::vector< std::vector< AI::PathNode >> &walkable_area)
- ~Archer ()
- void update (sf::Time time, std::vector< std::vector< std::unique_ptr< MapElement >>> &map_objects, std::list< std::shared_ptr< Character >> &list_of_bots, sf::RenderWindow &window)
- void dealDamage (sf::Time time, std::list< std::shared_ptr< Character >> &list_of_bots, sf::RenderTarget &window)

Private Member Functions

- void _dealBowDamage (std::list< std::shared_ptr< Character >> &list_of_bots)

Private Attributes

- Arrow * _arrows

Additional Inherited Members

6.2.1 Constructor & Destructor Documentation

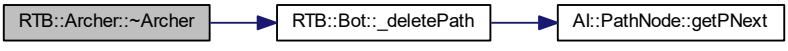
6.2.1.1 Archer()

```
RTB::Archer::Archer (
            sf::Texture texture,
            short health_points,
            sf::Texture & arrow_texture,
            std::vector< std::vector< AI::PathNode >> & walkable_area )
```

6.2.1.2 ~Archer()

```
RTB::Archer::~Archer ( )
```

Here is the call graph for this function:



6.2.2 Member Function Documentation

6.2.2.1 _dealBowDamage()

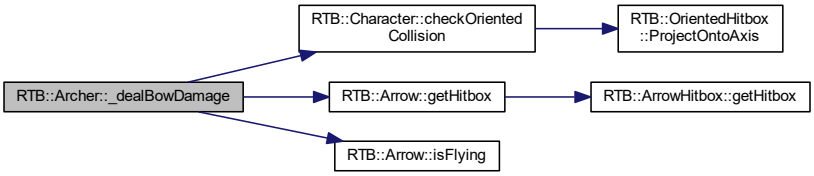
```
void RTB::Archer::_dealBowDamage (
            std::list< std::shared_ptr< Character >> & list_of_bots )  [private]
```

Function responsible for dealing damage to enemies

Parameters

list_of_bots	list of possible enemies

Here is the call graph for this function:



6.2.2.2 dealDamage()

```
void RTB::Archer::dealDamage (
            sf::Time time,
            std::list< std::shared_ptr< Character >> & list_of_bots,
            sf::RenderTarget & window )  [virtual]
```

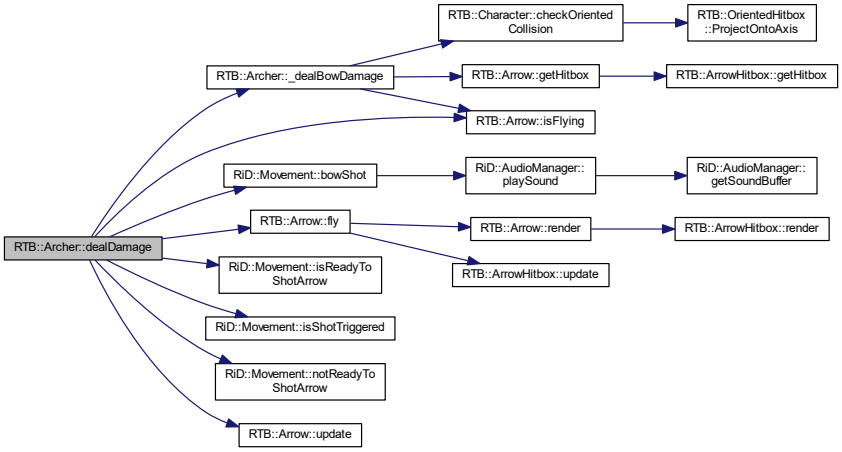
Dealing damage to bots of enemy team

Parameters

time	time needed for combat animations
list_of_bots	list of possible enemies
window	render window

Implements RTB::Character.

Here is the call graph for this function:



6.2.2.3 update()

```
void RTB::Archer::update (
            sf::Time time,
            std::vector< std::vector< std::unique_ptr< MapElement >>> & map_objects,
            std::list< std::shared_ptr< Character >> & list_of_bots,
            sf::RenderWindow & window )  [virtual]
```

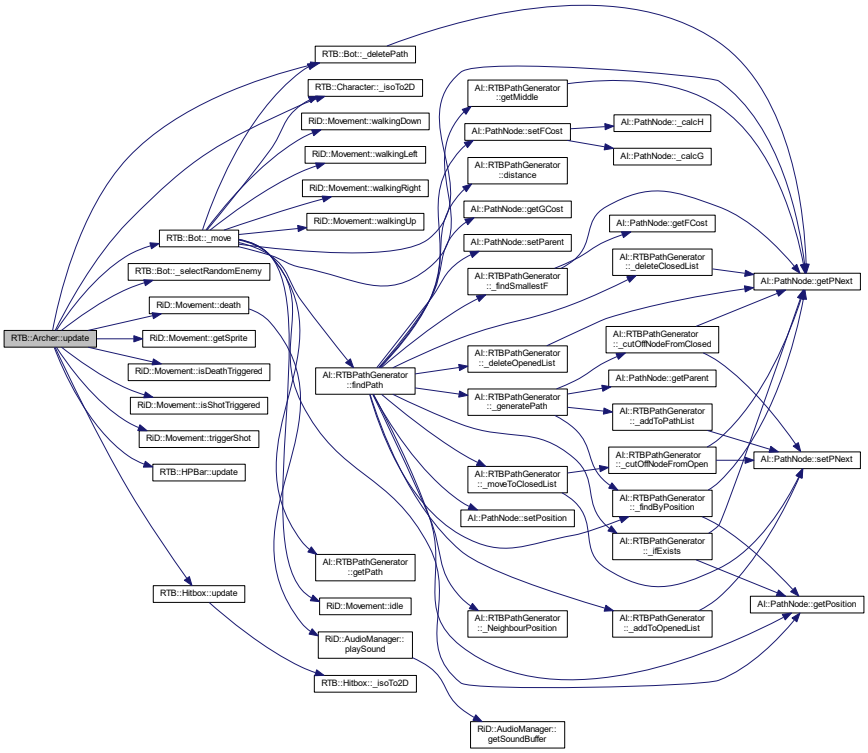
Function responsible for all of the behaviours

Parameters

time	game time
map_objects	all collidable objects to avoid
list_of_bots	list of bots given as target to attack

Implements RTB::Character.

Here is the call graph for this function:



6.2.3 Member Data Documentation

6.2.3.1 _arrows

```
Arrow* RTB::Archer::_arrows  [private]
```

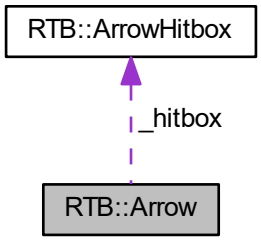
The documentation for this class was generated from the following files:

- Archer.h
- Archer.cpp

6.3 RTB::Arrow Class Reference

```
#include <Arrow.h>
```

Collaboration diagram for RTB::Arrow:



Public Member Functions

- Arrow (sf::Sprite *&character, sf::Texture &texture)
- ~Arrow ()
- void update ()

  Updates arrow starting position.
- void render (sf::RenderTarget &window)

  Renders arrow.
- void fly (sf::Time time, sf::RenderTarget &window, sf::Vector2i destination)

  Function responsible for moving arrow.
- bool isFlying ()

  Checks if arrow is flying.
- sf::Vector2f getPosition ()

  Gets arrow position.
- sf::RectangleShape getHitbox ()

Private Types

- enum directions { up, left, down, right }

Private Member Functions

- sf::Vector2i _isoTo2D (sf::Vector2f position)

Private Attributes

- ArrowHitbox * _hitbox = nullptr
- sf::Sprite * _sprite = nullptr
- sf::Sprite * _character
- sf::Time _fly_time
- sf::Time _start_time
- bool _is_flying
- bool _change_direction
- short _direction
- float _speed = 7.f
- float _rotation
- float _deltaX
- float _deltaY
- float _B

6.3.1 Member Enumeration Documentation

6.3.1.1 directions

```
enum RTB::Arrow::directions  [private]
```

Enumerator

up	
left	
down	
right	

6.3.2 Constructor & Destructor Documentation

6.3.2.1 Arrow()

```
RTB::Arrow::Arrow (
            sf::Sprite *& character,
            sf::Texture & texture )
```

6.3.2.2 ~Arrow()

```
RTB::Arrow::~Arrow ( )
```

6.3.3 Member Function Documentation

6.3.3.1 _isoTo2D()

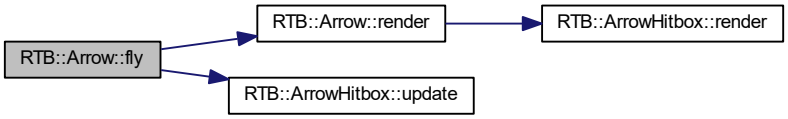
```
sf::Vector2i RTB::Arrow::_isoTo2D (
            sf::Vector2f position )  [private]
```

6.3.3.2 fly()

```
void RTB::Arrow::fly (
            sf::Time time,
            sf::RenderTarget & window,
            sf::Vector2i destination )
```

Function responsible for moving arrow.

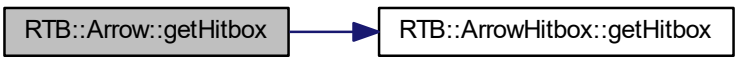
Here is the call graph for this function:



6.3.3.3 getHitbox()

```
sf::RectangleShape RTB::Arrow::getHitbox ( )
```

Here is the call graph for this function:



6.3.3.4 getPosition()

```
sf::Vector2f RTB::Arrow::getPosition ( )
```

Gets arrow position.

6.3.3.5 isFlying()

```
bool RTB::Arrow::isFlying ( )
```

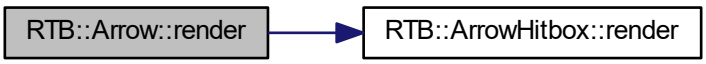
Checks if arrow is flying.

6.3.3.6 render()

```
void RTB::Arrow::render (
            sf::RenderTarget & window )
```

Renders arrow.

Here is the call graph for this function:



6.3.3.7 update()

```
void RTB::Arrow::update ( )
```

Updates arrow starting position.

6.3.4 Member Data Documentation

6.3.4.1 _B

```
float RTB::Arrow::_B  [private]
```

6.3.4.2 _change_direction

```
bool RTB::Arrow::_change_direction  [private]
```

6.3.4.3 _character

```
sf::Sprite* RTB::Arrow::_character  [private]
```

6.3.4.4 _deltaX

```
float RTB::Arrow::_deltaX  [private]
```

6.3.4.5 _deltaY

```
float RTB::Arrow::_deltaY  [private]
```

6.3.4.6 _direction

```
short RTB::Arrow::_direction  [private]
```

6.3.4.7 _fly_time

```
sf::Time RTB::Arrow::_fly_time  [private]
```

6.3.4.8 _hitbox

```
ArrowHitbox* RTB::Arrow::_hitbox = nullptr  [private]
```

6.3.4.9 _is_flying

```
bool RTB::Arrow::_is_flying  [private]
```

6.3.4.10 _rotation

```
float RTB::Arrow::_rotation  [private]
```

6.3.4.11 _speed

```
float RTB::Arrow::_speed = 7.f  [private]
```

6.3.4.12 _sprite

```
sf::Sprite* RTB::Arrow::_sprite = nullptr  [private]
```

6.3.4.13 _start_time

```
sf::Time RTB::Arrow::_start_time  [private]
```

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

- Arrow.h
- Arrow.cpp

6.4 RTB::ArrowHitbox Class Reference

```
#include <ArrowHitbox.h>
```

Public Member Functions

- ArrowHitbox (sf::Sprite *&arrow)
- void update (short direction, float angle)

  Changes position of hitbox as arrow is moving.
- void render (sf::RenderTarget &window)

  Draws hitbox.
- sf::RectangleShape & getHitbox ()

  Returns rectangle of hitbox.

Private Types

- enum directions { up, left, down, right }

Private Attributes

- sf::RectangleShape _hitbox
- sf::Sprite * _arrow = nullptr

6.4.1 Member Enumeration Documentation

6.4.1.1 directions

```
enum RTB::ArrowHitbox::directions  [private]
```

Enumerator

up	
left	
down	
right	

6.4.2 Constructor & Destructor Documentation

6.4.2.1 ArrowHitbox()

```
RTB::ArrowHitbox::ArrowHitbox (
            sf::Sprite *& arrow )
```

6.4.3 Member Function Documentation

6.4.3.1 getHitbox()

```
sf::RectangleShape & RTB::ArrowHitbox::getHitbox ( )
```

Returns rectangle of hitbox.

6.4.3.2 render()

```
void RTB::ArrowHitbox::render (
            sf::RenderTarget & window )
```

Draws hitbox.

6.4.3.3 update()

```
void RTB::ArrowHitbox::update (
            short direction,
            float angle )
```

Changes position of hitbox as arrow is moving.

6.4.4 Member Data Documentation

6.4.4.1 _arrow

```
sf::Sprite* RTB::ArrowHitbox::_arrow = nullptr  [private]
```

6.4.4.2 _hitbox

```
sf::RectangleShape RTB::ArrowHitbox::_hitbox  [private]
```

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

- ArrowHitbox.h
- ArrowHitbox.cpp

6.5 RiD::AssetManager Class Reference

```
#include <AssetManager.h>
```

Public Member Functions

- AssetManager ()
- ~AssetManager ()
- void setTexture (std::string tex_name, std::string file_name)
- sf::Texture & getTexture (std::string tex_name)
- void setFont (std::string font_name, std::string file_name)
- sf::Font & getFont (std::string font_name)

Private Attributes

- std::map< std::string, sf::Texture > _textures
- std::map< std::string, sf::Font > _fonts

6.5.1 Constructor & Destructor Documentation

6.5.1.1 AssetManager()

```
RiD::AssetManager::AssetManager ( )
```

6.5.1.2 ~AssetManager()

```
RiD::AssetManager::~AssetManager ( )
```

6.5.2 Member Function Documentation

6.5.2.1 getFont()

```
sf::Font & RiD::AssetManager::getFont (
            std::string font_name )
```

6.5.2.2 getTexture()

```
sf::Texture & RiD::AssetManager::getTexture (
            std::string tex_name )
```

6.5.2.3 setFont()

```
void RiD::AssetManager::setFont (
            std::string font_name,
            std::string file_name )
```

6.5.2.4 setTexture()

```
void RiD::AssetManager::setTexture (
            std::string tex_name,
            std::string file_name )
```

6.5.3 Member Data Documentation

6.5.3.1 _fonts

```
std::map<std::string, sf::Font> RiD::AssetManager::_fonts  [private]
```

6.5.3.2 _textures

```
std::map<std::string, sf::Texture> RiD::AssetManager::_textures  [private]
```

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

- AssetManager.h
- AssetManager.cpp

6.6 RiD::AudioManager Class Reference

```
#include <AudioManager.h>
```

Public Member Functions

- AudioManager ()
- void setSoundBuffer (std::string sound_name, std::string file_name)
- sf::SoundBuffer & getSoundBuffer (std::string sound_name)
- void playSound (const std::string &sound_name, const float &volume, const float &pitch=1.f, const bool &loop=false)

  Plays sound with specified parameters.
- void playMusic (const std::string &file_name, const float &volume)

  Plays music with specified parameters.

Private Attributes

- std::map< std::string, sf::SoundBuffer > _audio_buffer
- std::map< std::string, sf::Sound > _sounds
- sf::Sound _sound
- sf::Music _music

6.6.1 Constructor & Destructor Documentation

6.6.1.1 AudioManager()

```
RiD::AudioManager::AudioManager ( )
```

Here is the call graph for this function:

6.6.2 Member Function Documentation

6.6.2.1 getSoundBuffer()

```
sf::SoundBuffer & RiD::AudioManager::getSoundBuffer (
            std::string sound_name )
```

Returns

sound buffer

6.6.2.2 playMusic()

```
void RiD::AudioManager::playMusic (
            const std::string & file_name,
            const float & volume )
```

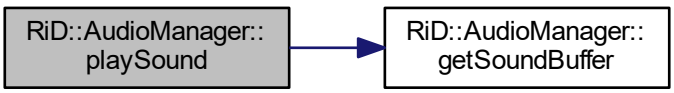
Plays music with specified parameters.

6.6.2.3 playSound()

```
void RiD::AudioManager::playSound (
            const std::string & sound_name,
            const float & volume,
            const float & pitch = 1.f,
            const bool & loop = false )
```

Plays sound with specified parameters.

Here is the call graph for this function:



6.6.2.4 setSoundBuffer()

```
void RiD::AudioManager::setSoundBuffer (
            std::string sound_name,
            std::string file_name )
```

Adds sound buffer

Parameters

sound_name	name
file_name	sound file

6.6.3 Member Data Documentation

6.6.3.1 _audio_buffer

```
std::map<std::string, sf::SoundBuffer> RiD::AudioManager::_audio_buffer  [private]
```

6.6.3.2 _music

```
sf::Music RiD::AudioManager::_music  [private]
```

6.6.3.3 _sound

```
sf::Sound RiD::AudioManager::_sound  [private]
```

6.6.3.4 _sounds

```
std::map<std::string, sf::Sound> RiD::AudioManager::_sounds  [private]
```

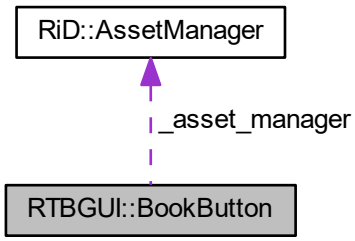
The documentation for this class was generated from the following files:

- AudioManager.h
- AudioManager.cpp

6.7 RTBGUI::BookButton Class Reference

`#include <BookButton.h>`

Collaboration diagram for RTBGUI::BookButton:



Public Member Functions

- BookButton ()
- void update (sf::Vector2f position)
- void render (sf::RenderWindow *&window)
- void setHovered (bool hover)
- sf::Sprite getSprite ()

Private Attributes

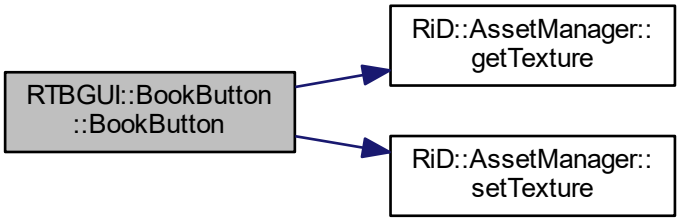
- sf::Sprite _book
- sf::Sprite _book_hovered
- RiD::AssetManager _asset_manager
- bool _is_hovered

6.7.1 Constructor & Destructor Documentation

6.7.1.1 BookButton()

```
RTBGUI::BookButton::BookButton ( )
```

Here is the call graph for this function:



6.7.2 Member Function Documentation

6.7.2.1 getSprite()

```
sf::Sprite RTBGUI::BookButton::getSprite ( )
```

6.7.2.2 render()

```
void RTBGUI::BookButton::render (
            sf::RenderWindow *& window )
```

6.7.2.3 setHovered()

```
void RTBGUI::BookButton::setHovered (
            bool hover )
```

6.7.2.4 update()

```
void RTBGUI::BookButton::update (
            sf::Vector2f position )
```

6.7.3 Member Data Documentation

6.7.3.1 _asset_manager

```
RiD::AssetManager RTBGUI::BookButton::_asset_manager  [private]
```

6.7.3.2 _book

```
sf::Sprite RTBGUI::BookButton::_book  [private]
```

6.7.3.3 _book_hovered

```
sf::Sprite RTBGUI::BookButton::_book_hovered  [private]
```

6.7.3.4 _is_hovered

```
bool RTBGUI::BookButton::_is_hovered  [private]
```

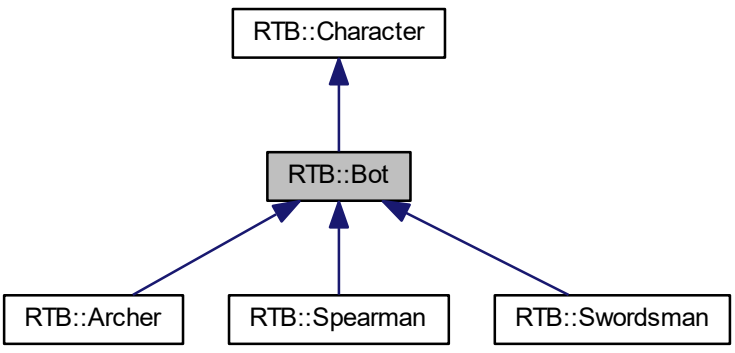
The documentation for this class was generated from the following files:

- BookButton.h
- BookButton.cpp

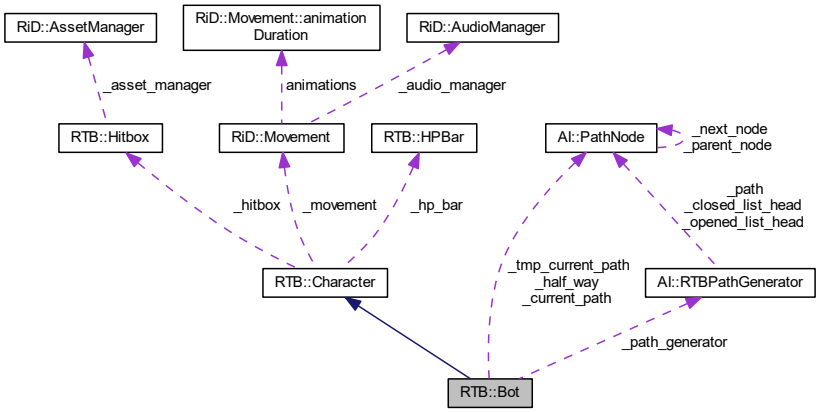
6.8 RTB::Bot Class Reference

```
#include <Bot.h>
```

Inheritance diagram for RTB::Bot:



Collaboration diagram for RTB::Bot:



Protected Member Functions

- std::list< std::shared_ptr< Character > >::iterator _selectRandomEnemy (std::list< std::shared_ptr< Character >>::iterator start, std::list< std::shared_ptr< Character >>::iterator end)
- void _deletePath ()

  Deletes path.
- void _move (sf::Time time)

Protected Attributes

- AI::RTBPathGenerator * _path_generator = nullptr
- AI::PathNode * _current_path = nullptr
- AI::PathNode * _tmp_current_path = nullptr
- AI::PathNode * _half_way = nullptr
- sf::Vector2i _current_enemy_position
- sf::Vector2i _end_path_position
- sf::Vector2i _attack_position
- bool _is_enemy_choosen = false
- sf::Vector2i _shot_destination
- std::list< std::shared_ptr< Character > >::iterator _choosen_enemy

Additional Inherited Members

6.8.1 Member Function Documentation

6.8.1.1 _deletePath()

```
void RTB::Bot::_deletePath ( )  [protected]
```

Deletes path.

Here is the call graph for this function:

6.8.1.2 _move()

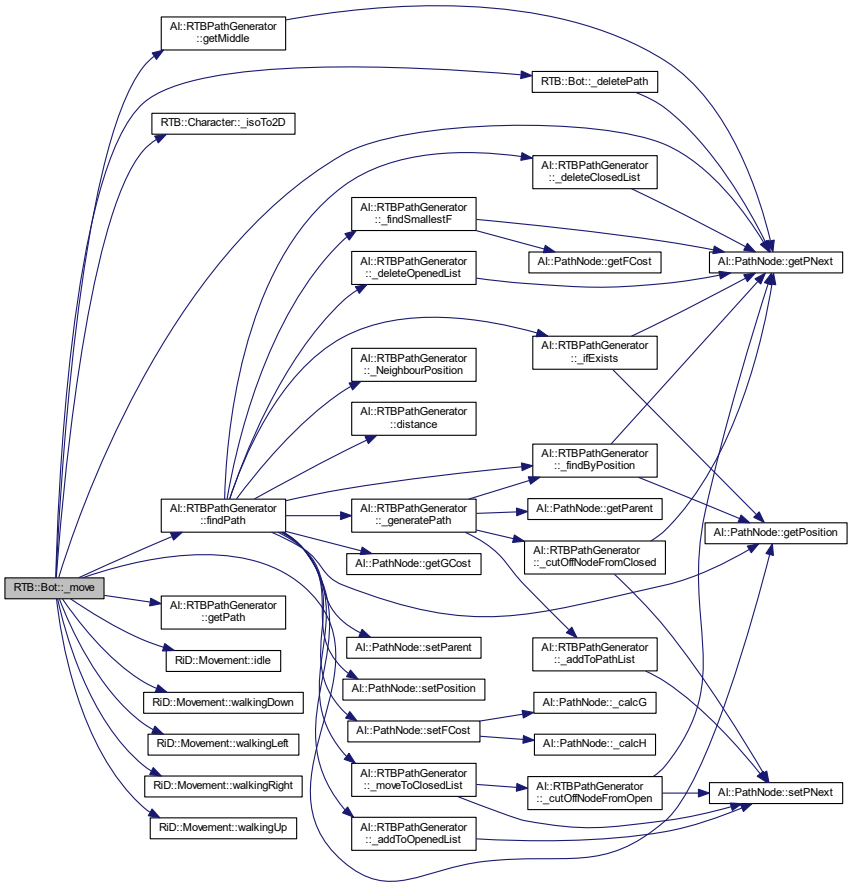
```
void RTB::Bot::_move (
            sf::Time time )  [protected]
```

Function responsible for movement (up,down,left,right and diagonally)

Parameters

time	game time

Here is the call graph for this function:



6.8.1.3 _selectRandomEnemy()

```
std::list< std::shared_ptr< Character > >::iterator RTB::Bot::_selectRandomEnemy (
            std::list< std::shared_ptr< Character >>::iterator start,
            std::list< std::shared_ptr< Character >>::iterator end )  [protected]
```

Function selects random enemy from the list

Parameters

start	begin iterator
end	end iterator

Returns

choosen enemy returned as an iterator

6.8.2 Member Data Documentation

6.8.2.1 _attack_position

```
sf::Vector2i RTB::Bot::_attack_position  [protected]
```

6.8.2.2 _choosen_enemy

```
std::list<std::shared_ptr<Character> >::iterator RTB::Bot::_choosen_enemy  [protected]
```

6.8.2.3 _current_enemy_position

```
sf::Vector2i RTB::Bot::_current_enemy_position  [protected]
```

6.8.2.4 _current_path

```
AI::PathNode* RTB::Bot::_current_path = nullptr  [protected]
```

6.8.2.5 _end_path_position

```
sf::Vector2i RTB::Bot::_end_path_position  [protected]
```

6.8.2.6 _half_way

```
AI::PathNode * RTB::Bot::_half_way = nullptr  [protected]
```

6.8.2.7 _is_enemy_choosen

```
bool RTB::Bot::_is_enemy_choosen = false  [protected]
```

6.8.2.8 _path_generator

```
AI::RTBPathGenerator* RTB::Bot::_path_generator = nullptr  [protected]
```

6.8.2.9 _shot_destination

```
sf::Vector2i RTB::Bot::_shot_destination  [protected]
```

6.8.2.10 _tmp_current_path

```
AI::PathNode * RTB::Bot::_tmp_current_path = nullptr  [protected]
```

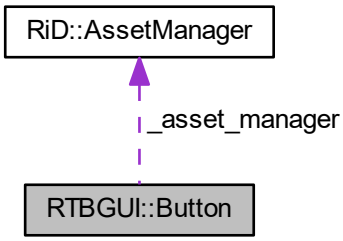
The documentation for this class was generated from the following files:

- Bot.h
- Bot.cpp

6.9 RTBGUI::Button Class Reference

```
#include <Button.h>
```

Collaboration diagram for RTBGUI::Button:



Public Member Functions

- Button (std::string text)
- void update (sf::Vector2f position)
- void render (sf::RenderWindow *&window)
- void setHovered (bool hover)
- sf::Sprite getSprite ()

Private Attributes

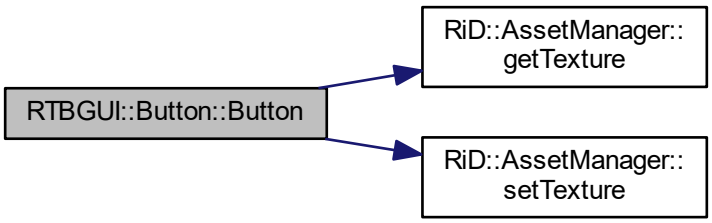
- std::unique_ptr< Message > _text
- sf::Sprite _rectangular_button
- sf::Sprite _rectangular_button_pressed
- RiD::AssetManager _asset_manager
- bool _is_hovered

6.9.1 Constructor & Destructor Documentation

6.9.1.1 Button()

```
RTBGUI::Button::Button (
            std::string text )
```

Here is the call graph for this function:



6.9.2 Member Function Documentation

6.9.2.1 getSprite()

```
sf::Sprite RTBGUI::Button::getSprite ( )
```

6.9.2.2 render()

```
void RTBGUI::Button::render (
            sf::RenderWindow *& window )
```

6.9.2.3 setHovered()

```
void RTBGUI::Button::setHovered (
            bool hover )
```

6.9.2.4 update()

```
void RTBGUI::Button::update (
            sf::Vector2f position )
```

6.9.3 Member Data Documentation

6.9.3.1 _asset_manager

```
RiD::AssetManager RTBGUI::Button::_asset_manager  [private]
```

6.9.3.2 _is_hovered

```
bool RTBGUI::Button::_is_hovered  [private]
```

6.9.3.3 _rectangular_button

```
sf::Sprite RTBGUI::Button::_rectangular_button  [private]
```

6.9.3.4 _rectangular_button_pressed

```
sf::Sprite RTBGUI::Button::_rectangular_button_pressed  [private]
```

6.9.3.5 _text

```
std::unique_ptr<Message> RTBGUI::Button::_text  [private]
```

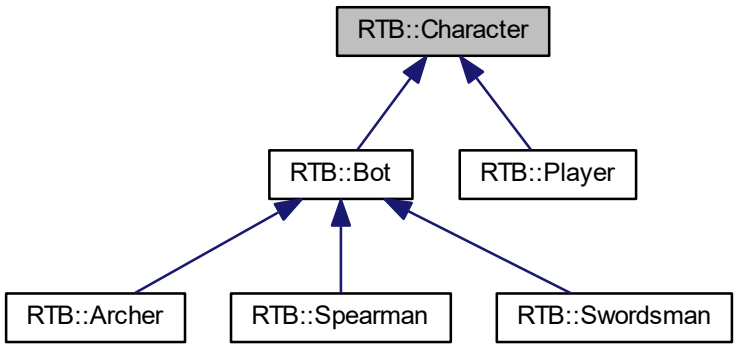
The documentation for this class was generated from the following files:

- Button.h
- Button.cpp

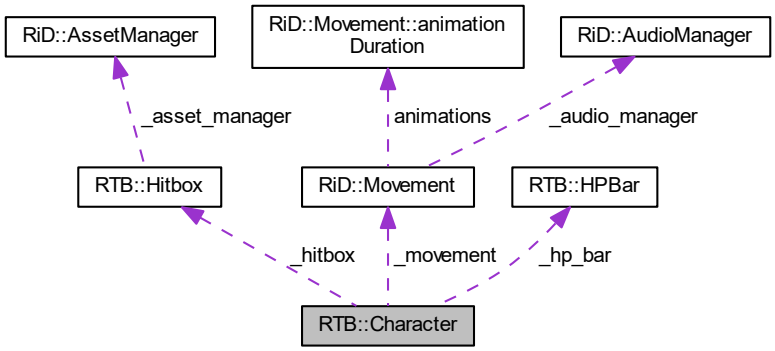
6.10 RTB::Character Class Reference

`#include <Character.h>`

Inheritance diagram for RTB::Character:



Collaboration diagram for RTB::Character:



Public Member Functions

- virtual void setPosition (sf::Vector2f position)
- virtual void update (sf::Time time, std::vector< std::vector< std::unique_ptr< MapElement >>> &map_↩ objects, std::list< std::shared_ptr< Character >> &list_of_bots, sf::RenderWindow &window)=0
- virtual void render (sf::RenderWindow &window)
- virtual bool isAlive ()

  Checks if character is alive.
- virtual sf::Vector2f getPosition ()
- bool checkOrientedCollision (const sf::RectangleShape &Object1, const sf::RectangleShape &Object2)

- virtual void subtractHP (short value)
- virtual sf::RectangleShape getHitbox ()
- virtual void dealDamage (sf::Time time, std::list< std::shared_ptr< Character >> &list_of_bots, sf::Render↩ Target &window)=0
- virtual void deadBody (sf::RenderWindow &window)

Protected Types

- enum directions { directions::up, directions::left, directions::down, directions::right }

Protected Member Functions

- sf::Vector2i _isoTo2D (sf::Vector2f position)

Protected Attributes

- sf::Sprite * _character_sprite = nullptr
- HPBar * _hp_bar = nullptr
- RiD::Movement * _movement = nullptr
- Hitbox * _hitbox = nullptr
- short _health_points = 0
- short _sword_damage = 0
- short _spear_damage = 0
- short _bow_damage = 0
- sf::Vector2f _position
- bool _is_alive = false
- short _direction = 1
- float _speed = 0.0
- bool _moving_up = true
- bool _moving_down = true
- bool _moving_right = true
- bool _moving_left = true

6.10.1 Member Enumeration Documentation

6.10.1.1 directions

`enum RTB::Character::directions [strong], [protected]`

Enumerator

up	
left	
down	
right	

6.10.2 Member Function Documentation

6.10.2.1 _isoTo2D()

```
sf::Vector2i RTB::Character::_isoTo2D (
            sf::Vector2f position )  [protected]
```

Changes isometric coordinates to 2D coordinates

Parameters

position	position in 2D system

6.10.2.2 checkOrientedCollision()

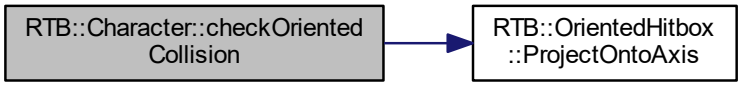
```
bool RTB::Character::checkOrientedCollision (
            const sf::RectangleShape & Object1,
            const sf::RectangleShape & Object2 )
```

Checks if two objects collide witch each other

Parameters

Object1	
Object2	

Here is the call graph for this function:



6.10.2.3 deadBody()

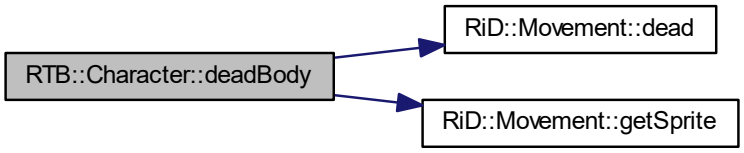
```
void RTB::Character::deadBody (
            sf::RenderWindow & window )  [virtual]
```

Function drows dead body of a character

Parameters

window	render target

Here is the call graph for this function:



6.10.2.4 dealDamage()

```
virtual void RTB::Character::dealDamage (
            sf::Time time,
            std::list< std::shared_ptr< Character >> & list_of_bots,
            sf::RenderTarget & window )  [pure virtual]
```

Dealing damage to bots of enemy team

Parameters

time	time needed for combat animations
list_of_bots	list of possible enemies
window	render window

Implemented in RTB::Player, RTB::Archer, RTB::Spearman, and RTB::Swordsman.

6.10.2.5 getHitbox()

```
sf::RectangleShape RTB::Character::getHitbox ( )  [virtual]
```

Returns character's hitbox as rectangle

Returns

rectangle hitbox

Here is the call graph for this function:

6.10.2.6 getPosition()

```
sf::Vector2f RTB::Character::getPosition ( )  [virtual]
```

Gets character position

Returns

characters position

Here is the call graph for this function:

6.10.2.7 isAlive()

```
bool RTB::Character::isAlive ( )  [virtual]
```

Checks if character is alive.

6.10.2.8 render()

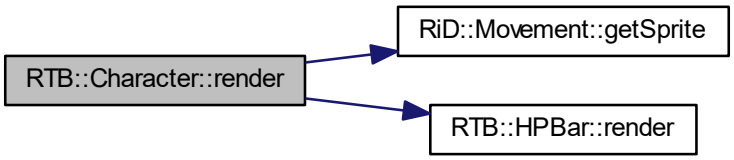
```
void RTB::Character::render (
            sf::RenderWindow & window )  [virtual]
```

Draws character's sprite

Parameters

window	render target

Here is the call graph for this function:



6.10.2.9 setPosition()

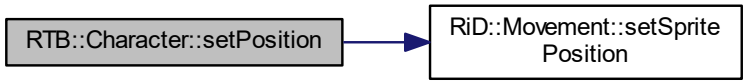
```
void RTB::Character::setPosition (
            sf::Vector2f position )  [virtual]
```

Sets character's position

Parameters

position	position

Here is the call graph for this function:



6.10.2.10 subtractHP()

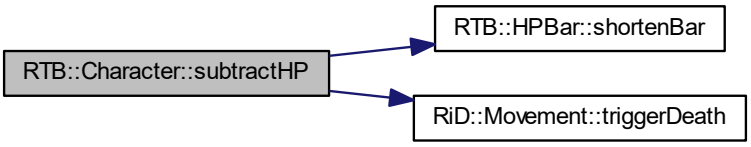
```
void RTB::Character::subtractHP (
            short value )  [virtual]
```

Subtracts character's hp

Parameters

value	health points to subtract

Here is the call graph for this function:



6.10.2.11 update()

```
virtual void RTB::Character::update (
            sf::Time time,
            std::vector< std::vector< std::unique_ptr< MapElement >>> & map_objects,
            std::list< std::shared_ptr< Character >> & list_of_bots,
            sf::RenderWindow & window )  [pure virtual]
```

Function responsible for all of the moves

Parameters

time	game time
map_objects	all collidable objects to avoid
list_of_bots	list of bots given as target to attack

Implemented in RTB::Player, RTB::Archer, RTB::Spearman, and RTB::Swordsman.

6.10.3 Member Data Documentation

6.10.3.1 _bow_damage

```
short RTB::Character::_bow_damage = 0  [protected]
```

6.10.3.2 _character_sprite

```
sf::Sprite* RTB::Character::_character_sprite = nullptr  [protected]
```

6.10.3.3 _direction

```
short RTB::Character::_direction = 1  [protected]
```

6.10.3.4 _health_points

```
short RTB::Character::_health_points = 0  [protected]
```

6.10.3.5 _hitbox

```
Hitbox* RTB::Character::_hitbox = nullptr  [protected]
```

6.10.3.6 _hp_bar

```
HPBar* RTB::Character::_hp_bar = nullptr  [protected]
```

6.10.3.7 _is_alive

```
bool RTB::Character::_is_alive = false  [protected]
```

6.10.3.8 _movement

```
RiD::Movement* RTB::Character::_movement = nullptr  [protected]
```

6.10.3.9 _moving_down

```
bool RTB::Character::_moving_down = true  [protected]
```

6.10.3.10 _moving_left

```
bool RTB::Character::_moving_left = true  [protected]
```

6.10.3.11 _moving_right

```
bool RTB::Character::_moving_right = true  [protected]
```

6.10.3.12 _moving_up

```
bool RTB::Character::_moving_up = true  [protected]
```

6.10.3.13 _position

```
sf::Vector2f RTB::Character::_position  [protected]
```

6.10.3.14 _spear_damage

```
short RTB::Character::_spear_damage = 0  [protected]
```

6.10.3.15 _speed

```
float RTB::Character::_speed = 0.0  [protected]
```

6.10.3.16 _sword_damage

```
short RTB::Character::_sword_damage = 0  [protected]
```

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

- Character.h
- Character.cpp

6.11 RiD::ConfigurationLoader Class Reference

`#include <ConfigurationLoader.h>`

Static Public Member Functions

- static int getIntData (std::string sectionName, std::string dataName)
- static std::string getStringData (std::string sectionName, std::string dataName)

6.11.1 Member Function Documentation

6.11.1.1 getIntData()

```
int RiD::ConfigurationLoader::getIntData (
            std::string sectionName,
            std::string dataName )  [static]
```

6.11.1.2 getStringData()

```
std::string RiD::ConfigurationLoader::getStringData (
            std::string sectionName,
            std::string dataName )  [static]
```

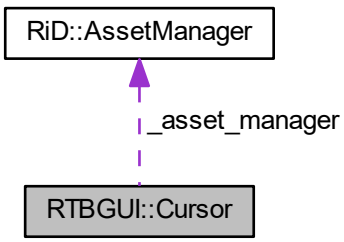
The documentation for this class was generated from the following files:

- ConfigurationLoader.h
- ConfigurationLoader.cpp

6.12 RTBGUI::Cursor Class Reference

`#include <Cursor.h>`

Collaboration diagram for RTBGUI::Cursor:



Public Member Functions

- Cursor ()
- void update (sf::Vector2f position)
- void render (sf::RenderWindow *&window)

Private Attributes

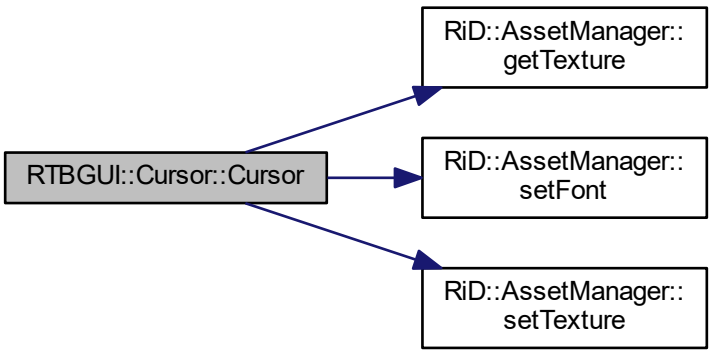
- sf::Sprite _cursor
- RiD::AssetManager _asset_manager

6.12.1 Constructor & Destructor Documentation

6.12.1.1 Cursor()

```
RTBGUI::Cursor::Cursor ( )
```

Here is the call graph for this function:



6.12.2 Member Function Documentation

6.12.2.1 render()

```
void RTBGUI::Cursor::render (
            sf::RenderWindow *& window )
```

6.12.2.2 update()

```
void RTBGUI::Cursor::update (
            sf::Vector2f position )
```

6.12.3 Member Data Documentation

6.12.3.1 _asset_manager

```
RiD::AssetManager RTBGUI::Cursor::_asset_manager  [private]
```

6.12.3.2 _cursor

```
sf::Sprite RTBGUI::Cursor::_cursor  [private]
```

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

- Cursor.h
- Cursor.cpp

6.13 RiD::gameDat Struct Reference

```
#include <RiDGame.h>
```

Public Attributes

- sf::RenderWindow window

6.13.1 Member Data Documentation

6.13.1.1 window

```
sf::RenderWindow RiD::gameDat::window
```

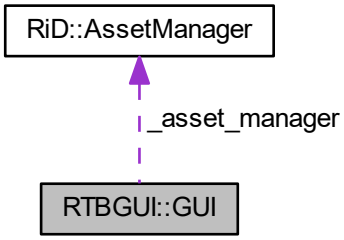
The documentation for this struct was generated from the following file:

- RiDGame.h

6.14 RTBGUI::GUI Class Reference

`#include <GUI.h>`

Collaboration diagram for RTBGUI::GUI:



Public Member Functions

- GUI (sf::RenderWindow &window)
- ~GUI ()
- void update (bool &is_paused, bool &is_surrendered, bool ally_team_dead, bool enemy_team_dead, bool &return_from_battle)

  Updates all GUI.
- void render (bool &is_paused, bool &is_surrendered, bool ally_team_dead, bool enemy_team_dead)

  Renders all GUI.
- void setCameraZoom (float zoom)

  Zooming.
- void setCameraCenter (sf::Vector2f center)

  Centers camera on certain position.
- void showLeftPanel (bool show)

  Shows panel with controls.
- bool isLeftPanelShown ()

  Checks if panel with controls is hidden or not.
- sf::View getCamera ()
- std::shared_ptr< Button > getButtonNo ()
- std::shared_ptr< Button > getButtonYes ()
- std::shared_ptr< Button > getButtonOk ()
- std::shared_ptr< BookButton > getButtonBook ()

Private Attributes

- sf::RenderWindow * _window
- std::unique_ptr< Cursor > _cursor
- std::shared_ptr< Button > _button_yes
- std::shared_ptr< Button > _button_no
- std::shared_ptr< Button > _button_ok

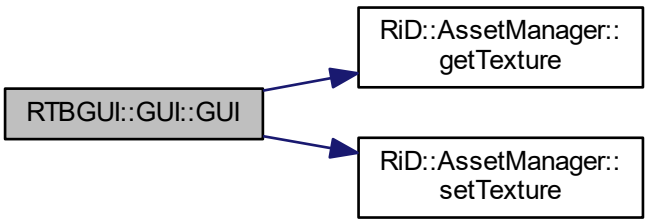
- std::unique_ptr< WindowBorder > _window_border
- std::unique_ptr< Panel > _panel
- std::unique_ptr< Menu > _menu
- std::unique_ptr< Message > _message_menu
- std::unique_ptr< Message > _message_lost
- std::unique_ptr< Message > _message_won
- std::shared_ptr< BookButton > _book
- RiD::AssetManager _asset_manager
- sf::Sprite _menu_background
- sf::Sprite _lost_background
- sf::Sprite _won_background
- sf::View _camera
- sf::View _gui
- sf::Event _event
- bool _show_left_panel

6.14.1 Constructor & Destructor Documentation

6.14.1.1 GUI()

```
RTBGUI::GUI::GUI (
            sf::RenderWindow & window )
```

Here is the call graph for this function:



6.14.1.2 ~GUI()

```
RTBGUI::GUI::~GUI ( )
```

6.14.2 Member Function Documentation

6.14.2.1 getButtonBook()

```
std::shared_ptr< BookButton > RTBGUI::GUI::getButtonBook ( )
```

6.14.2.2 getButtonNo()

```
std::shared_ptr< Button > RTBGUI::GUI::getButtonNo ( )
```

6.14.2.3 getButtonOk()

```
std::shared_ptr< Button > RTBGUI::GUI::getButtonOk ( )
```

6.14.2.4 getButtonYes()

```
std::shared_ptr< Button > RTBGUI::GUI::getButtonYes ( )
```

6.14.2.5 getCamera()

```
sf::View RTBGUI::GUI::getCamera ( )
```

6.14.2.6 isLeftPanelShown()

```
bool RTBGUI::GUI::isLeftPanelShown ( )
```

Checks if panel with controls is hidden or not.

6.14.2.7 render()

```
void RTBGUI::GUI::render (
            bool & is_paused,
            bool & is_surrendered,
            bool ally_team_dead,
            bool enemy_team_dead )
```

Renders all GUI.

6.14.2.8 setCameraCenter()

```
void RTBGUI::GUI::setCameraCenter (
            sf::Vector2f center )
```

Centers camera on certain position.

6.14.2.9 setCameraZoom()

```
void RTBGUI::GUI::setCameraZoom (
            float zoom )
```

Zooming.

6.14.2.10 showLeftPanel()

```
void RTBGUI::GUI::showLeftPanel (
            bool show )
```

Shows panel with controls.

6.14.2.11 update()

```
void RTBGUI::GUI::update (
            bool & is_paused,
            bool & is_surrendered,
            bool ally_team_dead,
            bool enemy_team_dead,
            bool & return_from_battle )
```

Updates all GUI.

6.14.3 Member Data Documentation

6.14.3.1 _asset_manager

```
RiD::AssetManager RTBGUI::GUI::_asset_manager  [private]
```

6.14.3.2 _book

```
std::shared_ptr<BookButton> RTBGUI::GUI::_book  [private]
```

6.14.3.3 _button_no

```
std::shared_ptr<Button> RTBGUI::GUI::_button_no  [private]
```

6.14.3.4 _button_ok

```
std::shared_ptr<Button> RTBGUI::GUI::_button_ok  [private]
```

6.14.3.5 _button_yes

```
std::shared_ptr<Button> RTBGUI::GUI::_button_yes  [private]
```

6.14.3.6 _camera

```
sf::View RTBGUI::GUI::_camera  [private]
```

6.14.3.7 _cursor

```
std::unique_ptr<Cursor> RTBGUI::GUI::_cursor  [private]
```

6.14.3.8 _event

```
sf::Event RTBGUI::GUI::_event  [private]
```

6.14.3.9 _gui

```
sf::View RTBGUI::GUI::_gui  [private]
```

6.14.3.10 _lost_background

```
sf::Sprite RTBGUI::GUI::_lost_background  [private]
```

6.14.3.11 _menu

```
std::unique_ptr<Menu> RTBGUI::GUI::_menu  [private]
```

6.14.3.12 _menu_background

```
sf::Sprite RTBGUI::GUI::_menu_background  [private]
```

6.14.3.13 _message_lost

```
std::unique_ptr<Message> RTBGUI::GUI::_message_lost  [private]
```

6.14.3.14 _message_menu

```
std::unique_ptr<Message> RTBGUI::GUI::_message_menu  [private]
```

6.14.3.15 _message_won

```
std::unique_ptr<Message> RTBGUI::GUI::_message_won  [private]
```

6.14.3.16 _panel

```
std::unique_ptr<Panel> RTBGUI::GUI::_panel  [private]
```

6.14.3.17 _show_left_panel

```
bool RTBGUI::GUI::_show_left_panel  [private]
```

6.14.3.18 _window

```
sf::RenderWindow* RTBGUI::GUI::_window  [private]
```

6.14.3.19 _window_border

```
std::unique_ptr<WindowBorder> RTBGUI::GUI::_window_border  [private]
```

6.14.3.20 _won_background

```
sf::Sprite RTBGUI::GUI::_won_background  [private]
```

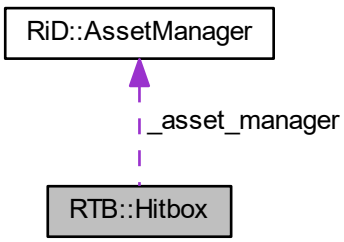
The documentation for this class was generated from the following files:

- GUI.h
- GUI.cpp

6.15 RTB::Hitbox Class Reference

```
#include <Hitbox.h>
```

Collaboration diagram for RTB::Hitbox:



Public Member Functions

- Hitbox (sf::Sprite ∗&object, sf::Vector2f size, sf::Vector2f offset)
- ∼Hitbox ()
- bool checkIntersection (const sf::FloatRect &rectangle)

  Checks if hitbox collides with another object's hitbox given as rectangle.
- void update ()

  Updates hitbox position.
- void render (sf::RenderTarget &window)

  Renders hitbox on the window.
- sf::RectangleShape & getRectangle ()

  Returns rectangle of hitbox.

Private Member Functions

- sf::Vector2f _isoTo2D (sf::Vector2f position)

Private Attributes

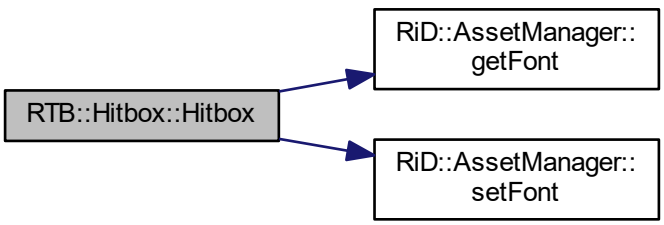
- RiD::AssetManager _asset_manager
- sf::Text _cords_map
- sf::RectangleShape _hitbox
- sf::Sprite ∗ _object
- sf::Vector2f _offset

6.15.1 Constructor & Destructor Documentation

6.15.1.1 Hitbox()

```
RTB::Hitbox::Hitbox (
            sf::Sprite *& object,
            sf::Vector2f size,
            sf::Vector2f offset )
```

Here is the call graph for this function:



6.15.1.2 ~Hitbox()

```
RTB::Hitbox::~Hitbox ( )
```

6.15.2 Member Function Documentation

6.15.2.1 _isoTo2D()

```
sf::Vector2f RTB::Hitbox::_isoTo2D (
            sf::Vector2f position )  [private]
```

6.15.2.2 checkIntersection()

```
bool RTB::Hitbox::checkIntersection (
            const sf::FloatRect & rectangle )
```

Checks if hitbox collides with another object's hitbox given as rectangle.

6.15.2.3 getRectangle()

```
sf::RectangleShape & RTB::Hitbox::getRectangle ( )
```

Returns rectangle of hitbox.

6.15.2.4 render()

```
void RTB::Hitbox::render (
            sf::RenderTarget & window )
```

Renders hitbox on the window.

6.15.2.5 update()

```
void RTB::Hitbox::update ( )
```

Updates hitbox position.

Here is the call graph for this function:

6.15.3 Member Data Documentation

6.15.3.1 _asset_manager

```
RiD::AssetManager RTB::Hitbox::_asset_manager  [private]
```

6.15.3.2 _cords_map

```
sf::Text RTB::Hitbox::_cords_map  [private]
```

6.15.3.3 _hitbox

```
sf::RectangleShape RTB::Hitbox::_hitbox  [private]
```

6.15.3.4 _object

```
sf::Sprite* RTB::Hitbox::_object  [private]
```

6.15.3.5 _offset

```
sf::Vector2f RTB::Hitbox::_offset  [private]
```

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

- Hitbox.h
- Hitbox.cpp

6.16 RTB::HPBar Class Reference

```
#include <HPBar.h>
```

Public Member Functions

- HPBar (sf::Sprite *&object, short hp)
- void update ()

  Updates bar position.
- void render (sf::RenderTarget &window)

  Renders bar.
- void shortenBar (short value)

  Shortens bar after something hit the object.

Private Attributes

- sf::RectangleShape _hp_bar
- sf::RectangleShape _hp_background
- sf::Sprite * _object
- sf::Vector2f _offset
- short _health_points
- float _bar_width

6.16.1 Constructor & Destructor Documentation

6.16.1.1 HPBar()

```
RTB::HPBar::HPBar (
            sf::Sprite *& object,
            short hp )
```

6.16.2 Member Function Documentation

6.16.2.1 render()

```
void RTB::HPBar::render (
            sf::RenderTarget & window )
```

Renders bar.

6.16.2.2 shortenBar()

```
void RTB::HPBar::shortenBar (
            short value )
```

Shortens bar after something hit the object.

6.16.2.3 update()

```
void RTB::HPBar::update ( )
```

Updates bar position.

6.16.3 Member Data Documentation

6.16.3.1 _bar_width

```
float RTB::HPBar::_bar_width  [private]
```

6.16.3.2 _health_points

```
short RTB::HPBar::_health_points  [private]
```

6.16.3.3 _hp_background

```
sf::RectangleShape RTB::HPBar::_hp_background  [private]
```

6.16.3.4 _hp_bar

```
sf::RectangleShape RTB::HPBar::_hp_bar  [private]
```

6.16.3.5 _object

```
sf::Sprite* RTB::HPBar::_object  [private]
```

6.16.3.6 _offset

```
sf::Vector2f RTB::HPBar::_offset  [private]
```

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

- HPBar.h
- HPBar.cpp

6.17 RTB::MapElement Class Reference

```
#include <MapElement.h>
```

Public Member Functions

- MapElement (sf::Texture *ground, sf::Vector2f position)
- void setFlora (sf::Texture *flora, sf::Vector2f position)

  Places flora elements like flowers.
- void setObject (sf::Texture *object, sf::Vector2f position, sf::Vector2f origin, sf::Vector2f hitbox_size, sf::Vector2f hitbox_position_offset, float hitbox_rotation_angle)

  Places collidable elements trees etc.
- sf::Sprite getGround ()
- sf::Sprite getFlora ()
- sf::Sprite getObjects ()
- sf::RectangleShape getObjectsHitbox ()
- bool isFloraNull ()
- bool isObjectNull ()
- bool isWalkable ()
- MapElement (sf::Texture *ground, sf::Vector2f position)
- void setFlora (sf::Texture *flora, sf::Vector2f position)
- void setObject (sf::Texture *object, sf::Vector2f position, sf::Vector2f origin, sf::Vector2f hitbox_size, sf::Vector2f hitbox_position_offset, float hitbox_rotation_angle)
- sf::Sprite getGround ()
- sf::Sprite getFlora ()
- sf::Sprite getObjects ()
- sf::RectangleShape getObjectsHitbox ()
- bool isFloraNull ()
- bool isObjectNull ()
- bool isWalkable ()

Private Attributes

- std::unique_ptr< sf::Sprite > _object_sprite = nullptr
- std::unique_ptr< sf::Sprite > _ground_sprite = nullptr
- std::unique_ptr< sf::Sprite > _flora_sprite = nullptr
- sf::Vector2u _object_sprite_size
- sf::Vector2u _ground_sprite_size
- sf::Vector2u _flora_sprite_size
- sf::RectangleShape _hitbox
- bool _walkable

6.17.1 Constructor & Destructor Documentation

6.17.1.1 MapElement() [1/2]

```
RTB::MapElement::MapElement (
            sf::Texture * ground,
            sf::Vector2f position )
```

6.17.1.2 MapElement() [2/2]

```
RTB::MapElement::MapElement (
            sf::Texture * ground,
            sf::Vector2f position )
```

6.17.2 Member Function Documentation

6.17.2.1 getFlora() [1/2]

```
sf::Sprite RTB::MapElement::getFlora ( )
```

6.17.2.2 getFlora() [2/2]

```
sf::Sprite RTB::MapElement::getFlora ( )
```

6.17.2.3 getGround() [1/2]

```
sf::Sprite RTB::MapElement::getGround ( )
```

6.17.2.4 getGround() [2/2]

```
sf::Sprite RTB::MapElement::getGround ( )
```

6.17.2.5 getObjects() [1/2]

```
sf::Sprite RTB::MapElement::getObjects ( )
```

6.17.2.6 getObjects() [2/2]

```
sf::Sprite RTB::MapElement::getObjects ( )
```

6.17.2.7 getObjectsHitbox() [1/2]

```
sf::RectangleShape RTB::MapElement::getObjectsHitbox ( )
```

6.17.2.8 getObjectsHitbox() [2/2]

```
sf::RectangleShape RTB::MapElement::getObjectsHitbox ( )
```

6.17.2.9 isFloraNull() [1/2]

```
bool RTB::MapElement::isFloraNull ( )
```

6.17.2.10 isFloraNull() [2/2]

```
bool RTB::MapElement::isFloraNull ( )
```

6.17.2.11 isObjectNull() [1/2]

```
bool RTB::MapElement::isObjectNull ( )
```

6.17.2.12 isObjectNull() [2/2]

```
bool RTB::MapElement::isObjectNull ( )
```

6.17.2.13 isWalkable() [1/2]

```
bool RTB::MapElement::isWalkable ( )
```

6.17.2.14 isWalkable() [2/2]

```
bool RTB::MapElement::isWalkable ( )
```

6.17.2.15 setFlora() [1/2]

```
void RTB::MapElement::setFlora (
            sf::Texture * flora,
            sf::Vector2f position )
```

6.17.2.16 setFlora() [2/2]

```
void RTB::MapElement::setFlora (
            sf::Texture * flora,
            sf::Vector2f position )
```

Places flora elements like flowers.

6.17.2.17 setObject() [1/2]

```
void RTB::MapElement::setObject (
            sf::Texture * object,
            sf::Vector2f position,
            sf::Vector2f origin,
            sf::Vector2f hitbox_size,
            sf::Vector2f hitbox_position_offset,
            float hitbox_rotation_angle )
```

6.17.2.18 setObject() [2/2]

```
void RTB::MapElement::setObject (
            sf::Texture * object,
            sf::Vector2f position,
            sf::Vector2f origin,
            sf::Vector2f hitbox_size,
            sf::Vector2f hitbox_position_offset,
            float hitbox_rotation_angle )
```

Places collidable elements trees etc.

6.17.3 Member Data Documentation

6.17.3.1 _flora_sprite

```
std::unique_ptr< sf::Sprite > RTB::MapElement::_flora_sprite = nullptr  [private]
```

6.17.3.2 _flora_sprite_size

```
sf::Vector2u RTB::MapElement::_flora_sprite_size  [private]
```

6.17.3.3 _ground_sprite

```
std::unique_ptr< sf::Sprite > RTB::MapElement::_ground_sprite = nullptr  [private]
```

6.17.3.4 _ground_sprite_size

```
sf::Vector2u RTB::MapElement::_ground_sprite_size  [private]
```

6.17.3.5 _hitbox

```
sf::RectangleShape RTB::MapElement::_hitbox  [private]
```

6.17.3.6 _object_sprite

```
std::unique_ptr< sf::Sprite > RTB::MapElement::_object_sprite = nullptr  [private]
```

6.17.3.7 _object_sprite_size

```
sf::Vector2u RTB::MapElement::_object_sprite_size  [private]
```

6.17.3.8 _walkable

```
bool RTB::MapElement::_walkable  [private]
```

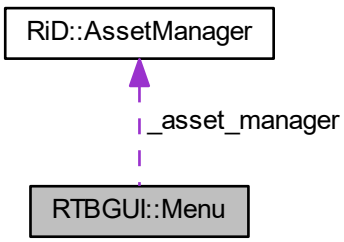
The documentation for this class was generated from the following files:

- Map/MapElement.h
- Map/MapElement.cpp

6.18 RTBGUI::Menu Class Reference

```
#include <Menu.h>
```

Collaboration diagram for RTBGUI::Menu:



Public Member Functions

- Menu ()
- void update (sf::Vector2f position)
- void render (sf::RenderWindow *&window)

Private Attributes

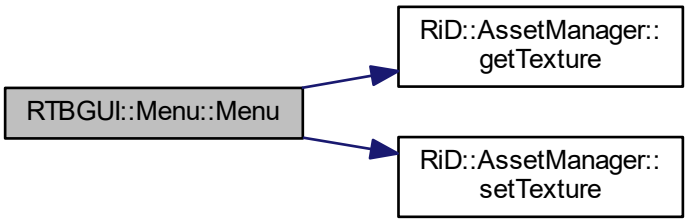
- sf::Sprite _menu
- RiD::AssetManager _asset_manager

6.18.1 Constructor & Destructor Documentation

6.18.1.1 Menu()

```
RTBGUI::Menu::Menu ( )
```

Here is the call graph for this function:



6.18.2 Member Function Documentation

6.18.2.1 render()

```
void RTBGUI::Menu::render (
            sf::RenderWindow *& window )
```

6.18.2.2 update()

```
void RTBGUI::Menu::update (
            sf::Vector2f position )
```

6.18.3 Member Data Documentation

6.18.3.1 _asset_manager

```
RiD::AssetManager RTBGUI::Menu::_asset_manager  [private]
```

6.18.3.2 _menu

```
sf::Sprite RTBGUI::Menu::_menu  [private]
```

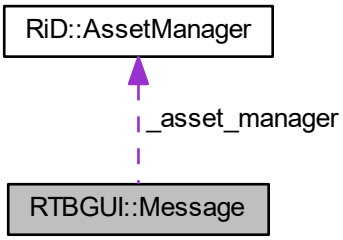
The documentation for this class was generated from the following files:

- Menu.h
- Menu.cpp

6.19 RTBGUI::Message Class Reference

```
#include <Message.h>
```

Collaboration diagram for RTBGUI::Message:



Public Member Functions

- Message (std::string message, int size)
- void update (sf::Vector2f position)
- void render (sf::RenderWindow *&window)

Private Attributes

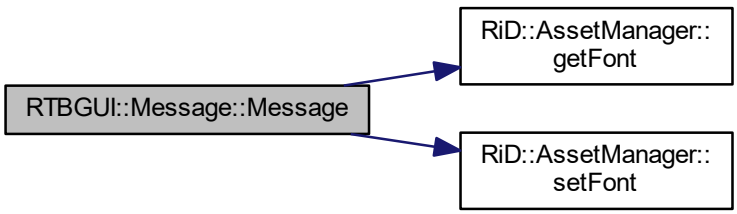
- sf::Text _message
- sf::Text _shadow
- RiD::AssetManager _asset_manager

6.19.1 Constructor & Destructor Documentation

6.19.1.1 Message()

```
RTBGUI::Message::Message (
            std::string message,
            int size )
```

Here is the call graph for this function:



6.19.2 Member Function Documentation

6.19.2.1 render()

```
void RTBGUI::Message::render (
            sf::RenderWindow *& window )
```

6.19.2.2 update()

```
void RTBGUI::Message::update (
            sf::Vector2f position )
```

6.19.3 Member Data Documentation

6.19.3.1 _asset_manager

```
RiD::AssetManager RTBGUI::Message::_asset_manager  [private]
```

6.19.3.2 _message

```
sf::Text RTBGUI::Message::_message  [private]
```

6.19.3.3 _shadow

```
sf::Text RTBGUI::Message::_shadow  [private]
```

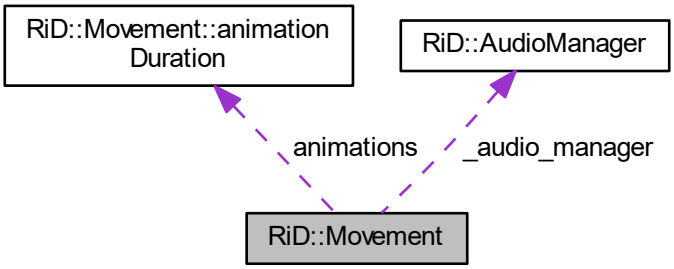
The documentation for this class was generated from the following files:

- Message.h
- Message.cpp

6.20 RiD::Movement Class Reference

```
#include <Movement.h>
```

Collaboration diagram for RiD::Movement:



Classes

- struct animationDuration

Public Member Functions

- Movement (sf::Texture texture, sf::Sprite ∗&object)
- void walkingUp (sf::Time time, float speed_x, float speed_y)

  Walking up animation.
- void walkingDown (sf::Time time, float speed_x, float speed_y)

  Walking down animation.
- void walkingLeft (sf::Time time, float speed_x, float speed_y)

  Walking left animation.
- void walkingRight (sf::Time time, float speed_x, float speed_y)

  Walking right animation.
- void swordSwing (sf::Time time)

  Sword attack.
- void spearPoke (sf::Time time)

  Spear attack.
- void bowShot (sf::Time time, short direction)

  Bow attack.
- void idle (sf::Time time)

  Standing.
- void dead ()

  Lying dead.
- void death (sf::Time time)

  Death animation.
- void triggerSpear ()

  Function triggers spear attack.
- void triggerAttack ()

  Function triggers sword attack.
- void triggerShot ()

  Function triggers bow shot.
- void triggerDeath ()

  Function triggers death.
- bool isSpearTriggered ()
- bool isAttackTriggered ()
- bool isShotTriggered ()
- bool isDeathTriggered ()
- bool isReadyToDealSpearDamage ()

  Checks if previous animation stopped.
- bool isReadyToDealSwordDamage ()

  Checks if previous animation stopped.
- void notReadyToDealSpearDamage ()

  Set not ready.
- void notReadyToDealSwordDamage ()

  Set not ready.
- bool isReadyToShotArrow ()

  Checks if previous animation stopped.
- void notReadyToShotArrow ()

  Set not ready.

- sf::Sprite getSprite ()

  Returns sprite.
- short getDirection ()
- void setSpritePosition (sf::Vector2f position)

  Sets sprite position.

Private Types

- enum directions { up, left, down, right }
- enum rowsInPNGFile {
  walkUpAnim = 8, walkLeftAnim, walkDownAnim, walkRightAnim,
  shotAnim = 16, deathAnim = 20, swordSwingAnim }

Private Attributes

- animationDuration animations
- RiD::AudioManager _audio_manager
- bool _is_attack_triggered
- bool _is_shot_triggered
- bool _is_death_triggered
- bool _is_spear_triggered
- short _row
- short _xCord
- short _yCord
- short _direction
- short _xAttackCord
- short _yAttackCord
- short _xshotCord
- short _yShotCord
- short _yDeathCord
- short _xDeathCord
- short _ySpearCord
- short _xSpearCord
- sf::Texture _texture
- sf::Sprite * _object
- bool _ready_to_deal_sword_damage
- bool _ready_to_shot_arrow
- bool _ready_to_deal_spear_damage

6.20.1 Member Enumeration Documentation

6.20.1.1 directions

```
enum RiD::Movement::directions  [private]
```

Enumerator

up	
left	
down	
right	

6.20.1.2 rowsInPNGFile

```
enum RiD::Movement::rowsInPNGFile  [private]
```

Enumerator

walkUpAnim	
walkLeftAnim	
walkDownAnim	
walkRightAnim	
shotAnim	
deathAnim	
swordSwingAnim	

6.20.2 Constructor & Destructor Documentation

6.20.2.1 Movement()

```
RiD::Movement::Movement (
            sf::Texture texture,
            sf::Sprite *& object )
```

6.20.3 Member Function Documentation

6.20.3.1 bowShot()

```
void RiD::Movement::bowShot (
            sf::Time time,
            short direction )
```

Bow attack.

Here is the call graph for this function:

6.20.3.2 dead()

```
void RiD::Movement::dead ( )
```

Lying dead.

6.20.3.3 death()

```
void RiD::Movement::death (
            sf::Time time )
```

Death animation.

Here is the call graph for this function:

6.20.3.4 getDirection()

```
short RiD::Movement::getDirection ( )
```

Returns

character direction

6.20.3.5 getSprite()

```
sf::Sprite RiD::Movement::getSprite ( )
```

Returns sprite.

6.20.3.6 idle()

```
void RiD::Movement::idle (
            sf::Time time )
```

Standing.

6.20.3.7 isAttackTriggered()

```
bool RiD::Movement::isAttackTriggered ( )
```

Returns

is sword attack animation triggered

6.20.3.8 isDeathTriggered()

```
bool RiD::Movement::isDeathTriggered ( )
```

Returns

is death animation triggered

6.20.3.9 isReadyToDealSpearDamage()

```
bool RiD::Movement::isReadyToDealSpearDamage ( )
```

Checks if previous animation stopped.

6.20.3.10 isReadyToDealSwordDamage()

```
bool RiD::Movement::isReadyToDealSwordDamage ( )
```

Checks if previous animation stopped.

6.20.3.11 isReadyToShotArrow()

```
bool RiD::Movement::isReadyToShotArrow ( )
```

Checks if previous animation stopped.

6.20.3.12 isShotTriggered()

```
bool RiD::Movement::isShotTriggered ( )
```

Returns

is shot animation triggered

6.20.3.13 isSpearTriggered()

```
bool RiD::Movement::isSpearTriggered ( )
```

Returns

is spear attack animation triggered

6.20.3.14 notReadyToDealSpearDamage()

```
void RiD::Movement::notReadyToDealSpearDamage ( )
```

Set not ready.

6.20.3.15 notReadyToDealSwordDamage()

```
void RiD::Movement::notReadyToDealSwordDamage ( )
```

Set not ready.

6.20.3.16 notReadyToShotArrow()

```
void RiD::Movement::notReadyToShotArrow ( )
```

Set not ready.

6.20.3.17 setSpritePosition()

```
void RiD::Movement::setSpritePosition (
            sf::Vector2f position )
```

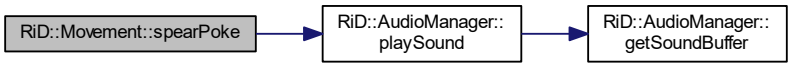
Sets sprite position.

6.20.3.18 spearPoke()

```
void RiD::Movement::spearPoke (
            sf::Time time )
```

Spear attack.

Here is the call graph for this function:

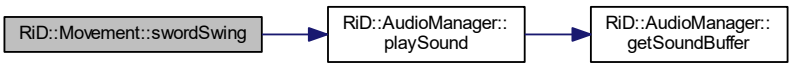


6.20.3.19 swordSwing()

```
void RiD::Movement::swordSwing (
            sf::Time time )
```

Sword attack.

Here is the call graph for this function:



6.20.3.20 triggerAttack()

```
void RiD::Movement::triggerAttack ( )
```

Function triggers sword attack.

6.20.3.21 triggerDeath()

```
void RiD::Movement::triggerDeath ( )
```

Function triggers death.

6.20.3.22 triggerShot()

```
void RiD::Movement::triggerShot ( )
```

Function triggers bow shot.

6.20.3.23 triggerSpear()

```
void RiD::Movement::triggerSpear ( )
```

Function triggers spear attack.

6.20.3.24 walkingDown()

```
void RiD::Movement::walkingDown (
            sf::Time time,
            float speed_x,
            float speed_y )
```

Walking down animation.

6.20.3.25 walkingLeft()

```
void RiD::Movement::walkingLeft (
            sf::Time time,
            float speed_x,
            float speed_y )
```

Walking left animation.

6.20.3.26 walkingRight()

```
void RiD::Movement::walkingRight (
            sf::Time time,
            float speed_x,
            float speed_y )
```

Walking right animation.

6.20.3.27 walkingUp()

```
void RiD::Movement::walkingUp (
            sf::Time time,
            float speed_x,
            float speed_y )
```

Walking up animation.

6.20.4 Member Data Documentation

6.20.4.1 _audio_manager

```
RiD::AudioManager RiD::Movement::_audio_manager  [private]
```

6.20.4.2 _direction

```
short RiD::Movement::_direction  [private]
```

6.20.4.3 _is_attack_triggered

```
bool RiD::Movement::_is_attack_triggered  [private]
```

6.20.4.4 _is_death_triggered

```
bool RiD::Movement::_is_death_triggered  [private]
```

6.20.4.5 _is_shot_triggered

```
bool RiD::Movement::_is_shot_triggered  [private]
```

6.20.4.6 _is_spear_triggered

```
bool RiD::Movement::_is_spear_triggered  [private]
```

6.20.4.7 _object

```
sf::Sprite* RiD::Movement::_object  [private]
```

6.20.4.8 _ready_to_deal_spear_damage

```
bool RiD::Movement::_ready_to_deal_spear_damage  [private]
```

6.20.4.9 _ready_to_deal_sword_damage

```
bool RiD::Movement::_ready_to_deal_sword_damage  [private]
```

6.20.4.10 _ready_to_shot_arrow

```
bool RiD::Movement::_ready_to_shot_arrow  [private]
```

6.20.4.11 _row

```
short RiD::Movement::_row  [private]
```

6.20.4.12 _texture

```
sf::Texture RiD::Movement::_texture  [private]
```

6.20.4.13 _xAttackCord

```
short RiD::Movement::_xAttackCord  [private]
```

6.20.4.14 _xCord

```
short RiD::Movement::_xCord  [private]
```

6.20.4.15 _xDeathCord

```
short RiD::Movement::_xDeathCord  [private]
```

6.20.4.16 _xshotCord

```
short RiD::Movement::_xshotCord  [private]
```

6.20.4.17 _xSpearCord

```
short RiD::Movement::_xSpearCord  [private]
```

6.20.4.18 _yAttackCord

```
short RiD::Movement::_yAttackCord  [private]
```

6.20.4.19 _yCord

```
short RiD::Movement::_yCord  [private]
```

6.20.4.20 _yDeathCord

```
short RiD::Movement::_yDeathCord  [private]
```

6.20.4.21 _yShotCord

```
short RiD::Movement::_yShotCord  [private]
```

6.20.4.22 _ySpearCord

```
short RiD::Movement::_ySpearCord  [private]
```

6.20.4.23 animations

```
animationDuration RiD::Movement::animations  [private]
```

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

- Movement.h
- Movement.cpp

6.21 RTB::OrientedHitbox Class Reference

```
#include <OrientedHitbox.h>
```

Public Member Functions

- OrientedHitbox (const sf::RectangleShape &Object, const int width, const int height)

  Calculate the four points of the OBB from a transformed (scaled, rotated...) sprite.
- void ProjectOntoAxis (const sf::Vector2f &Axis, float &Min, float &Max)

  Project all four points of the OBB onto the given axis and return the dotproducts of the two outermost points.

Public Attributes

- sf::Vector2f Points [4]

6.21.1 Constructor & Destructor Documentation

6.21.1.1 OrientedHitbox()

```
RTB::OrientedHitbox::OrientedHitbox (
            const sf::RectangleShape & Object,
            const int width,
            const int height )
```

Calculate the four points of the OBB from a transformed (scaled, rotated...) sprite.

6.21.2 Member Function Documentation

6.21.2.1 ProjectOntoAxis()

```
void RTB::OrientedHitbox::ProjectOntoAxis (
            const sf::Vector2f & Axis,
            float & Min,
            float & Max )
```

Project all four points of the OBB onto the given axis and return the dotproducts of the two outermost points.

6.21.3 Member Data Documentation

6.21.3.1 Points

```
sf::Vector2f RTB::OrientedHitbox::Points[4]
```

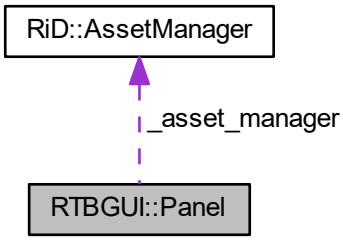
The documentation for this class was generated from the following files:

- OrientedHitbox.h
- OrientedHitbox.cpp

6.22 RTBGUI::Panel Class Reference

`#include <Panel.h>`

Collaboration diagram for RTBGUI::Panel:



Public Member Functions

- Panel ()
- void update (sf::Vector2f position)
- void render (sf::RenderWindow ∗&window)

Private Attributes

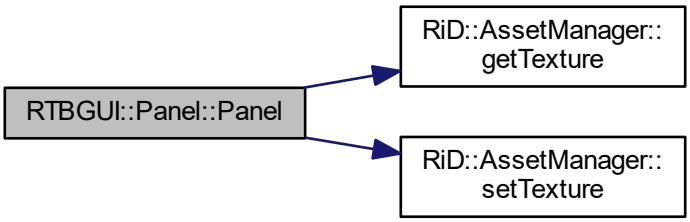
- sf::Sprite _panel
- std::unique_ptr< Message > _message
- RiD::AssetManager _asset_manager

6.22.1 Constructor & Destructor Documentation

6.22.1.1 Panel()

`RTBGUI::Panel::Panel ( )`

Here is the call graph for this function:



6.22.2 Member Function Documentation

6.22.2.1 render()

```
void RTBGUI::Panel::render (
            sf::RenderWindow *& window )
```

6.22.2.2 update()

```
void RTBGUI::Panel::update (
            sf::Vector2f position )
```

6.22.3 Member Data Documentation

6.22.3.1 _asset_manager

```
RiD::AssetManager RTBGUI::Panel::_asset_manager  [private]
```

6.22.3.2 _message

```
std::unique_ptr<Message> RTBGUI::Panel::_message  [private]
```

6.22.3.3 _panel

```
sf::Sprite RTBGUI::Panel::_panel  [private]
```

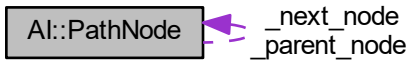
The documentation for this class was generated from the following files:

- Panel.h
- Panel.cpp

6.23 AI::PathNode Class Reference

`#include <PathNode.h>`

Collaboration diagram for AI::PathNode:



Public Member Functions

- PathNode ()
- sf::Vector2i getPosition ()

  Gets position of current node.
- int getFCost ()
- PathNode ∗& getParent ()
- PathNode ∗& getPNext ()
- void setPNext (PathNode ∗node)
- void setParent (PathNode ∗node)
- void setNotWalkable ()

  Determines which areas of the map are not walkable.
- void setFCost (sf::Vector2i end)
- void setPosition (sf::Vector2i pos)
- bool isWalkable ()

  Checks if area is walkable.
- int getGCost (sf::Vector2i start)

Private Member Functions

- void _calcG (sf::Vector2i start)
- void _calcH (sf::Vector2i end)

Private Attributes

- int _F
- int _G
- int _H
- PathNode ∗ _next_node
- PathNode ∗ _parent_node
- sf::Vector2i _position
- bool _walkable

6.23.1 Constructor & Destructor Documentation

6.23.1.1 PathNode()

```
AI::PathNode::PathNode ( )
```

6.23.2 Member Function Documentation

6.23.2.1 _calcG()

```
void AI::PathNode::_calcG (
            sf::Vector2i start )  [private]
```

Calculates G value which is the movement cost to move from the starting point to a given position

Parameters

start	starting position

6.23.2.2 _calcH()

```
void AI::PathNode::_calcH (
            sf::Vector2i end )  [private]
```

Calculates H value which is the estimated movement cost to move from that given position to the final destination.

Parameters

end	destination

6.23.2.3 getFCost()

```
int AI::PathNode::getFCost ( )
```

Returns

F cost which is sum of G and H

6.23.2.4 getGCost()

```
int AI::PathNode::getGCost (
            sf::Vector2i start )
```

Returns

G cost

Parameters

start	

6.23.2.5 getParent()

```
PathNode *& AI::PathNode::getParent ( )
```

Returns

parent of current node

6.23.2.6 getPNext()

```
PathNode *& AI::PathNode::getPNext ( )
```

Returns

next node

6.23.2.7 getPosition()

```
sf::Vector2i AI::PathNode::getPosition ( )
```

Gets position of current node.

6.23.2.8 isWalkable()

```
bool AI::PathNode::isWalkable ( )
```

Checks if area is walkable.

6.23.2.9 setFCost()

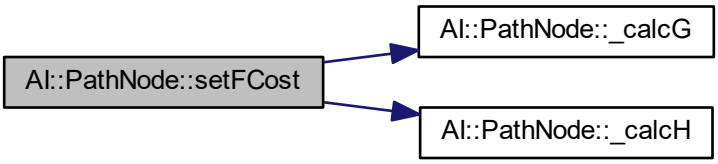
```
void AI::PathNode::setFCost (
            sf::Vector2i end )
```

Sets F cost

Parameters

end	destination

Here is the call graph for this function:



6.23.2.10 setNotWalkable()

```
void AI::PathNode::setNotWalkable ( )
```

Determines which areas of the map are not walkable.

6.23.2.11 setParent()

```
void AI::PathNode::setParent (
            PathNode * node )
```

Sets parent node

Parameters

node	pointer to the parent node

6.23.2.12 setPNext()

```
void AI::PathNode::setPNext (
            PathNode * node )
```

Sets next node

Parameters

node	pointer to the next node

6.23.2.13 setPosition()

```
void AI::PathNode::setPosition (
            sf::Vector2i pos )
```

Sets position

Parameters

pos	position

6.23.3 Member Data Documentation

6.23.3.1 _F

```
int AI::PathNode::_F  [private]
```

6.23.3.2 _G

```
int AI::PathNode::_G  [private]
```

6.23.3.3 _H

```
int AI::PathNode::_H  [private]
```

6.23.3.4 _next_node

```
PathNode* AI::PathNode::_next_node  [private]
```

6.23.3.5 _parent_node

```
PathNode * AI::PathNode::_parent_node  [private]
```

6.23.3.6 _position

```
sf::Vector2i AI::PathNode::_position  [private]
```

6.23.3.7 _walkable

```
bool AI::PathNode::_walkable  [private]
```

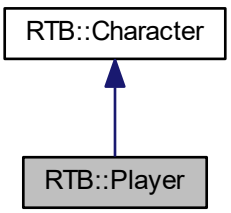
The documentation for this class was generated from the following files:

- PathNode.h
- PathNode.cpp

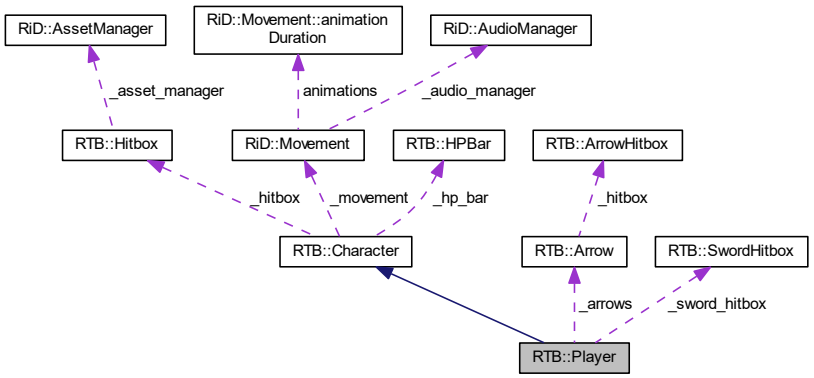
6.24 RTB::Player Class Reference

```
#include <Player.h>
```

Inheritance diagram for RTB::Player:



Collaboration diagram for RTB::Player:



Public Member Functions

- Player (sf::Texture texture, short health_points, sf::Texture &arrow_texture)
- ~Player ()
- void update (sf::Time time, std::vector< std::vector< std::unique_ptr< MapElement >>> &map_objects, std::list< std::shared_ptr< Character >> &list_of_bots, sf::RenderWindow &window)
- void dealDamage (sf::Time time, std::list< std::shared_ptr< Character >> &list_of_bots, sf::RenderTarget &window)

Private Member Functions

- void _dealSwordDamage (std::list< std::shared_ptr< Character >> &list_of_bots)
- void _dealBowDamage (std::list< std::shared_ptr< Character >> &list_of_bots)
- void _isColidingWithTile (std::vector< std::vector< std::unique_ptr< MapElement >>> &map_objects)

Private Attributes

- Arrow * _arrows
- SwordHitbox * _sword_hitbox
- sf::Vector2i _shot_destination

Additional Inherited Members

6.24.1 Constructor & Destructor Documentation

6.24.1.1 Player()

```
RTB::Player::Player (
            sf::Texture texture,
            short health_points,
            sf::Texture & arrow_texture )
```

6.24.1.2 ~Player()

```
RTB::Player::~Player ( )
```

6.24.2 Member Function Documentation

6.24.2.1 _dealBowDamage()

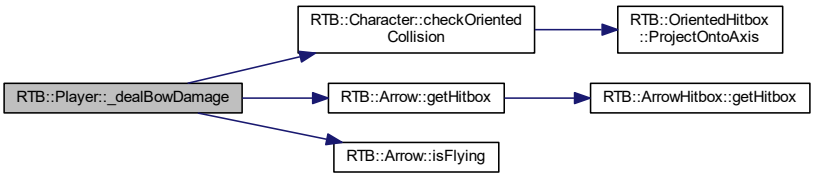
```
void RTB::Player::_dealBowDamage (
            std::list< std::shared_ptr< Character >> & list_of_bots )  [private]
```

Function responsible for dealing damage to enemies

Parameters

list_of_bots	list of possible enemies

Here is the call graph for this function:



6.24.2.2 _dealSwordDamage()

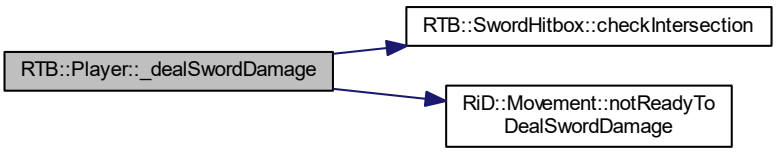
```
void RTB::Player::_dealSwordDamage (
            std::list< std::shared_ptr< Character >> & list_of_bots )  [private]
```

Function responsible for dealing damage to enemies

Parameters

list_of_bots	list of possible enemies

Here is the call graph for this function:



6.24.2.3 _isColidingWithTile()

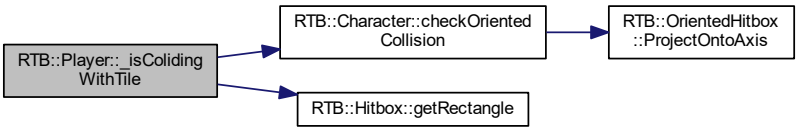
```
void RTB::Player::_isColidingWithTile (
            std::vector< std::vector< std::unique_ptr< MapElement >>> & map_objects )  [private]
```

Function checks collisions with objects

Parameters

map_objects	all collidable objects

Here is the call graph for this function:



6.24.2.4 dealDamage()

```
void RTB::Player::dealDamage (
            sf::Time time,
            std::list< std::shared_ptr< Character >> & list_of_bots,
            sf::RenderTarget & window )  [virtual]
```

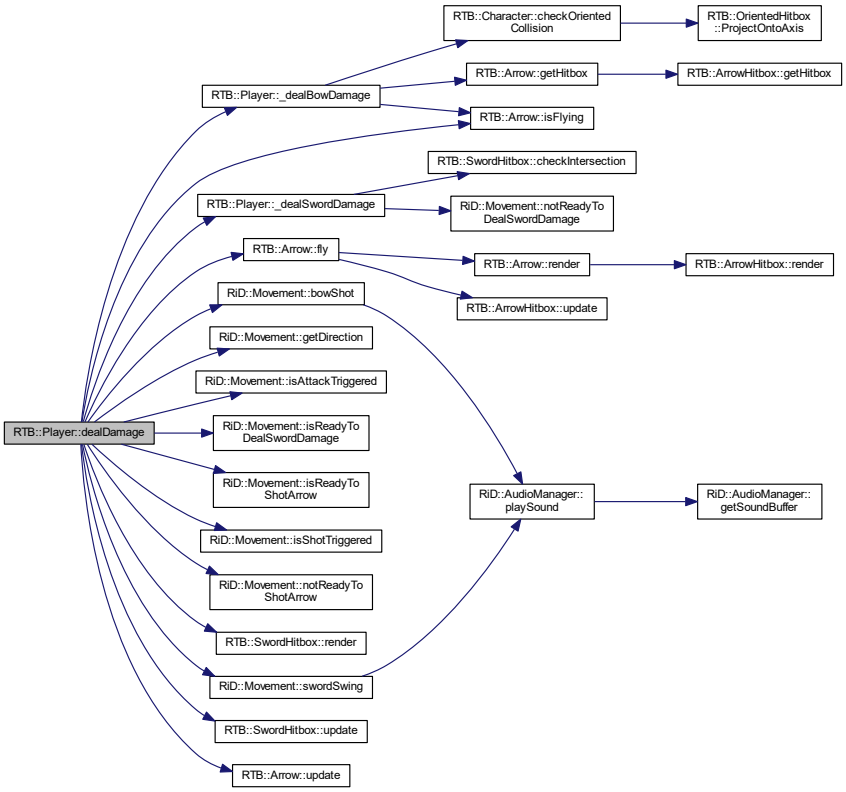
Dealing damage to bots of enemy team

Parameters

time	time needed for combat animations
list_of_bots	list of possible enemies
window	render window

Implements RTB::Character.

Here is the call graph for this function:



6.24.2.5 update()

```
void RTB::Player::update (
            sf::Time time,
            std::vector< std::vector< std::unique_ptr< MapElement >>> & map_objects,
            std::list< std::shared_ptr< Character >> & list_of_bots,
            sf::RenderWindow & window )  [virtual]
```

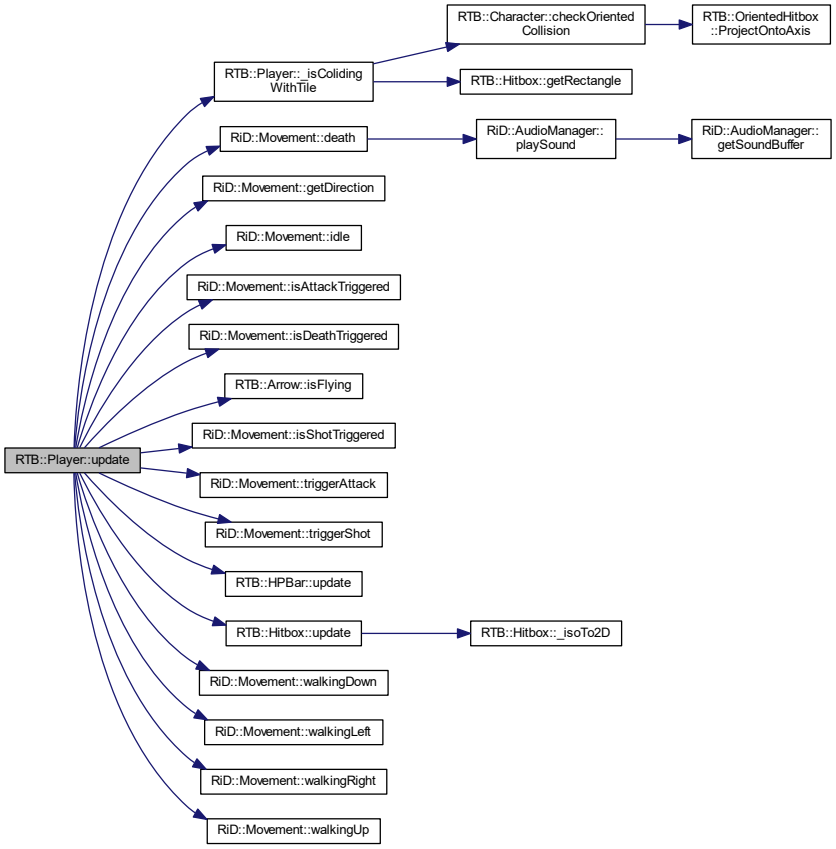
Function responsible for all of the player moves and behaviours

Parameters

time	game time
map_objects	all collidable objects to avoid
list_of_bots	list of bots given as target to attack

Implements RTB::Character.

Here is the call graph for this function:



6.24.3 Member Data Documentation

6.24.3.1 _arrows

```
Arrow* RTB::Player::_arrows  [private]
```

6.24.3.2 _shot_destination

```
sf::Vector2i RTB::Player::_shot_destination  [private]
```

6.24.3.3 _sword_hitbox

```
SwordHitbox* RTB::Player::_sword_hitbox  [private]
```

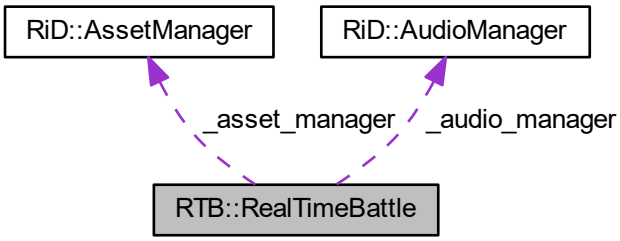
The documentation for this class was generated from the following files:

- Player.h
- Player.cpp

6.25 RTB::RealTimeBattle Class Reference

```
#include <RealTimeBattle.h>
```

Collaboration diagram for RTB::RealTimeBattle:



Public Member Functions

- RealTimeBattle (sf::RenderWindow &window)
- ~RealTimeBattle ()
- void mainLoop ()

Private Member Functions

- void _zoomEvent ()
- void _armyCreation ()
- bool _checkPlacementCollisions (std::list< std::shared_ptr< Character >>::iterator character)
- bool _isEnemyTeamDead ()
- bool _isAllyTeamDead ()
- std::list< std::shared_ptr< Character > >::iterator _pickRandomAlly ()
- void _charactersUpdatesAndRenders ()

Private Attributes

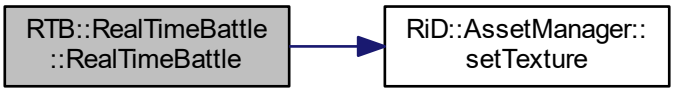
- std::shared_ptr< Character > _player
- sf::RenderWindow * _window
- sf::Event _event
- sf::View _camera
- sf::Clock _clock
- RiD::AssetManager _asset_manager
- RiD::AudioManager _audio_manager
- std::list< std::shared_ptr< Character > > _list_of_enemies
- std::list< std::shared_ptr< Character > > _list_of_allies
- std::unique_ptr< TileMap > _tile_map
- std::unique_ptr< RTBGUI::GUI > _user_interface
- std::list< std::shared_ptr< Character > >::iterator _choosen_ally
- double _zoom = 1.f
- bool _is_paused
- bool _is_surrendered
- bool _return_from_battle
- bool _is_ally_choosen

6.25.1 Constructor & Destructor Documentation

6.25.1.1 RealTimeBattle()

```
RTB::RealTimeBattle::RealTimeBattle (
            sf::RenderWindow & window )
```

Here is the call graph for this function:



6.25.1.2 ~RealTimeBattle()

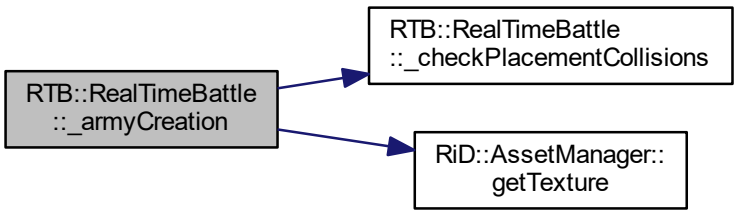
```
RTB::RealTimeBattle::~RealTimeBattle ( )
```

6.25.2 Member Function Documentation

6.25.2.1 _armyCreation()

```
void RTB::RealTimeBattle::_armyCreation ( )  [private]
```

Here is the call graph for this function:



6.25.2.2 _charactersUpdatesAndRenders()

```
void RTB::RealTimeBattle::_charactersUpdatesAndRenders ( )  [private]
```

6.25.2.3 _checkPlacementCollisions()

```
bool RTB::RealTimeBattle::_checkPlacementCollisions (
            std::list< std::shared_ptr< Character >>::iterator character )  [private]
```

6.25.2.4 _isAllyTeamDead()

```
bool RTB::RealTimeBattle::_isAllyTeamDead ( )  [private]
```

6.25.2.5 _isEnemyTeamDead()

```
bool RTB::RealTimeBattle::_isEnemyTeamDead ( )  [private]
```

6.25.2.6 _pickRandomAlly()

```
std::list< std::shared_ptr< Character > >::iterator RTB::RealTimeBattle::_pickRandomAlly ( )
[private]
```

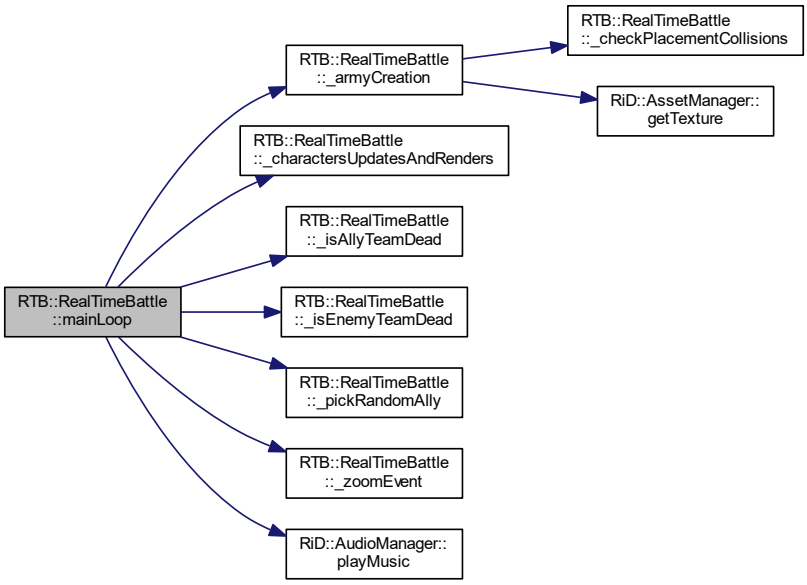
6.25.2.7 _zoomEvent()

```
void RTB::RealTimeBattle::_zoomEvent ( )  [private]
```

6.25.2.8 mainLoop()

```
void RTB::RealTimeBattle::mainLoop ( )
```

Here is the call graph for this function:



6.25.3 Member Data Documentation

6.25.3.1 _asset_manager

RiD::AssetManager RTB::RealTimeBattle::_asset_manager [private]

6.25.3.2 _audio_manager

RiD::AudioManager RTB::RealTimeBattle::_audio_manager [private]

6.25.3.3 _camera

sf::View RTB::RealTimeBattle::_camera [private]

6.25.3.4 _choosen_ally

std::list<std::shared_ptr<Character> >::iterator RTB::RealTimeBattle::_choosen_ally [private]

6.25.3.5 _clock

sf::Clock RTB::RealTimeBattle::_clock [private]

6.25.3.6 _event

sf::Event RTB::RealTimeBattle::_event [private]

6.25.3.7 _is_ally_choosen

bool RTB::RealTimeBattle::_is_ally_choosen [private]

6.25.3.8 _is_paused

bool RTB::RealTimeBattle::_is_paused [private]

6.25.3.9 _is_surrendered

```
bool RTB::RealTimeBattle::_is_surrendered  [private]
```

6.25.3.10 _list_of_allies

```
std::list<std::shared_ptr<Character> > RTB::RealTimeBattle::_list_of_allies  [private]
```

6.25.3.11 _list_of_enemies

```
std::list<std::shared_ptr<Character> > RTB::RealTimeBattle::_list_of_enemies  [private]
```

6.25.3.12 _player

```
std::shared_ptr<Character> RTB::RealTimeBattle::_player  [private]
```

6.25.3.13 _return_from_battle

```
bool RTB::RealTimeBattle::_return_from_battle  [private]
```

6.25.3.14 _tile_map

```
std::unique_ptr<TileMap> RTB::RealTimeBattle::_tile_map  [private]
```

6.25.3.15 _user_interface

```
std::unique_ptr<RTBGUI::GUI> RTB::RealTimeBattle::_user_interface  [private]
```

6.25.3.16 _window

```
sf::RenderWindow* RTB::RealTimeBattle::_window  [private]
```

6.25.3.17 _zoom

```
double RTB::RealTimeBattle::_zoom = 1.f  [private]
```

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

- RealTimeBattle.h
- RealTimeBattle.cpp

6.26 RiD::RiDGame Class Reference

```
#include <RiDGame.h>
```

Public Member Functions

- RiDGame (int width, int height, std::string title)
- ~RiDGame ()

Private Member Functions

- void Exec ()

Private Attributes

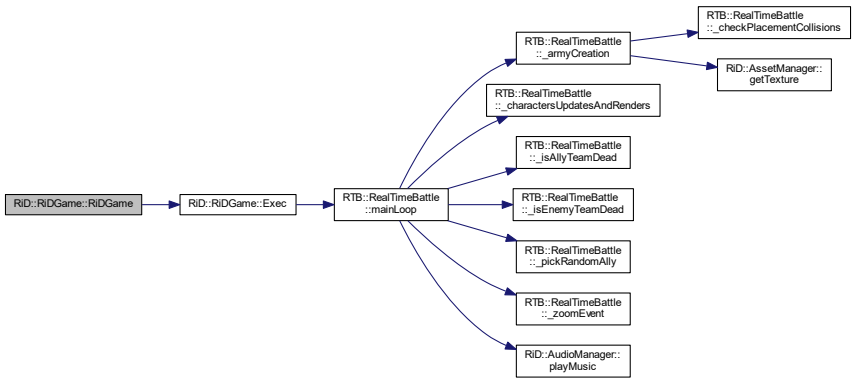
- sf::Clock _clock
- gameDatReference _data = std::make_shared<gameDat>()

6.26.1 Constructor & Destructor Documentation

6.26.1.1 RiDGame()

```
RiD::RiDGame::RiDGame (
            int width,
            int height,
            std::string title )
```

Here is the call graph for this function:



6.26.1.2 ~RiDGame()

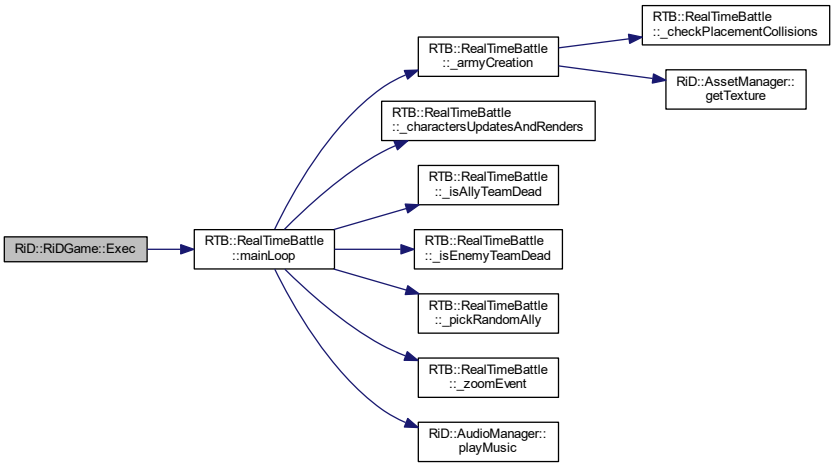
```
RiD::RiDGame::~RiDGame ( )
```

6.26.2 Member Function Documentation

6.26.2.1 Exec()

```
void RiD::RiDGame::Exec ( )  [private]
```

Here is the call graph for this function:



6.26.3 Member Data Documentation

6.26.3.1 _clock

```
sf::Clock RiD::RiDGame::_clock  [private]
```

6.26.3.2 _data

```
gameDatReference RiD::RiDGame::_data = std::make_shared<gameDat>()  [private]
```

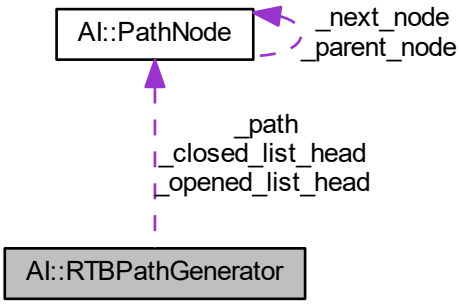
The documentation for this class was generated from the following files:

- RiDGame.h
- RiDGame.cpp

6.27 AI::RTBPathGenerator Class Reference

`#include <RTBPathGenerator.h>`

Collaboration diagram for AI::RTBPathGenerator:



Public Member Functions

- RTBPathGenerator (std::vector< std::vector< PathNode >> &walkable_area)
- void findPath (sf::Vector2i start, sf::Vector2i end)
- PathNode *& getPath ()
- PathNode * getMiddle ()
- int distance (sf::Vector2i start, sf::Vector2i end)

Private Member Functions

- void _addToOpenedList (PathNode *&node)
- void _addToPathList (PathNode *&node)
- void _moveToClosedList (PathNode *&node)
- void _deleteOpenedList ()

  Deletes opened list.
- void _deleteClosedList ()

  Deletes closed list.
- void _NeighbourPosition (unsigned short i, unsigned short j, unsigned short points[])

  Gets information about 8 closest nodes near the examined node.
- void _generatePath ()

  Generates path.
- bool _ifExists (PathNode neighbour, PathNode *pHead)
- PathNode * _cutOffNodeFromOpen (PathNode *&node)
- PathNode * _cutOffNodeFromClosed (PathNode *&node)
- PathNode * _findSmallestF ()
- PathNode * _findByPosition (sf::Vector2i position)

Private Attributes

- std::vector< std::vector< PathNode > > _walkable_area
- PathNode * _path
- PathNode * _opened_list_head
- PathNode * _closed_list_head
- sf::Vector2i _start
- sf::Vector2i _end
- unsigned int _width
- unsigned int _height

6.27.1 Constructor & Destructor Documentation

6.27.1.1 RTBPathGenerator()

```
AI::RTBPathGenerator::RTBPathGenerator (
            std::vector< std::vector< PathNode >> & walkable_area )
```

6.27.2 Member Function Documentation

6.27.2.1 _addToOpenedList()

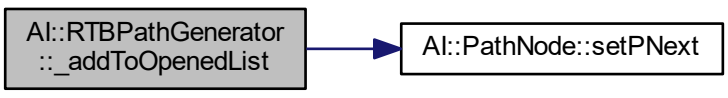
```
void AI::RTBPathGenerator::_addToOpenedList (
            PathNode *& node )  [private]
```

Adds node to opened list which keeps track of those nodes that need to be examined

Parameters

node	new node

Here is the call graph for this function:



6.27.2.2 _addToPathList()

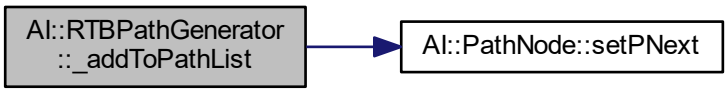
```
void AI::RTBPathGenerator::_addToPathList (
            PathNode *& node )  [private]
```

Adds node to path which is ready to be implemented

Parameters

node	new node

Here is the call graph for this function:



6.27.2.3 _cutOffNodeFromClosed()

```
PathNode * AI::RTBPathGenerator::_cutOffNodeFromClosed (
            PathNode *& node )  [private]
```

Cuts off node

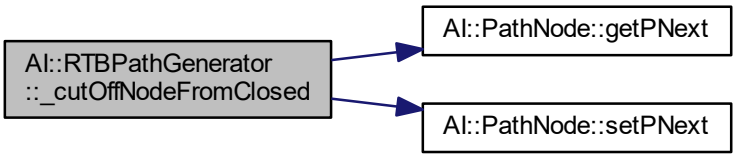
Returns

pointer to path node

Parameters

node	node to cut off

Here is the call graph for this function:



6.27.2.4 _cutOffNodeFromOpen()

```
PathNode * AI::RTBPathGenerator::_cutOffNodeFromOpen (
            PathNode *& node )  [private]
```

Cuts off node

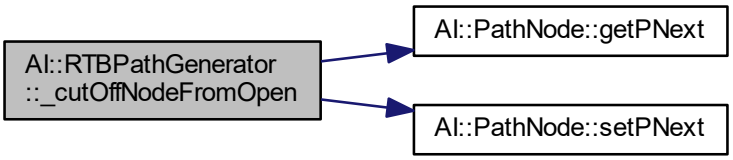
Returns

pointer to path node

Parameters

node	node to cut off

Here is the call graph for this function:

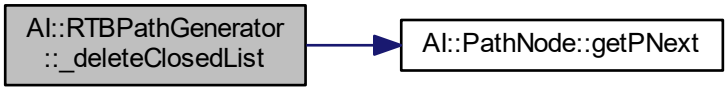


6.27.2.5 _deleteClosedList()

```
void AI::RTBPathGenerator::_deleteClosedList ( )  [private]
```

Deletes closed list.

Here is the call graph for this function:

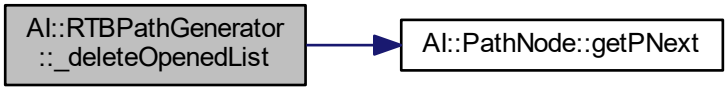


6.27.2.6 _deleteOpenedList()

```
void AI::RTBPathGenerator::_deleteOpenedList ( )  [private]
```

Deletes opened list.

Here is the call graph for this function:



6.27.2.7 _findByPosition()

```
PathNode * AI::RTBPathGenerator::_findByPosition (
            sf::Vector2i position )  [private]
```

Finds node with by position in closed list

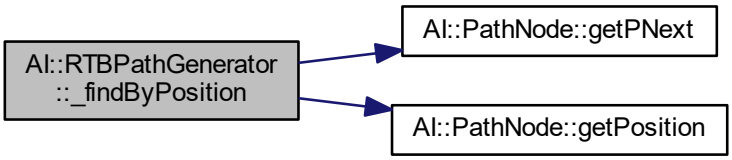
Returns

pointer to node

Parameters

position	

Here is the call graph for this function:



6.27.2.8 _findSmallestF()

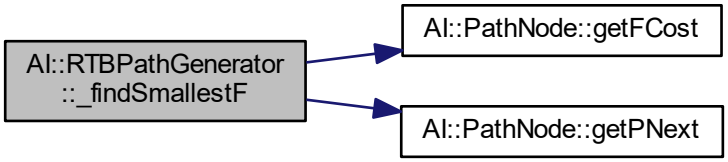
```
PathNode * AI::RTBPathGenerator::_findSmallestF ( )  [private]
```

Finds node with smallest F value in opened list

Returns

pointer to node

Here is the call graph for this function:

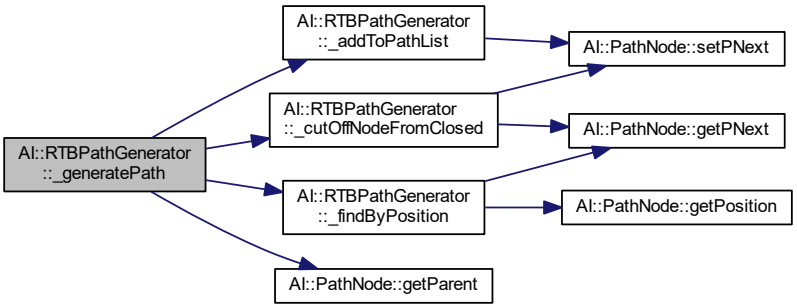


6.27.2.9 _generatePath()

```
void AI::RTBPathGenerator::_generatePath ( )  [private]
```

Generates path.

Here is the call graph for this function:



6.27.2.10 _ifExists()

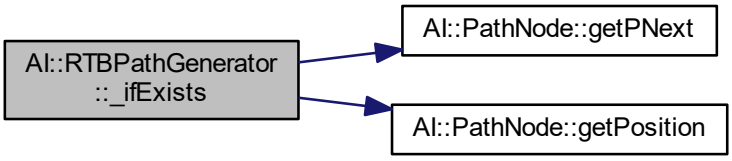
```
bool AI::RTBPathGenerator::_ifExists (
            PathNode neighbour,
            PathNode * pHead )  [private]
```

Checks if neighbour exists in the list

Parameters

neighbour	
pHead	pointer to current list

Here is the call graph for this function:



6.27.2.11 _moveToClosedList()

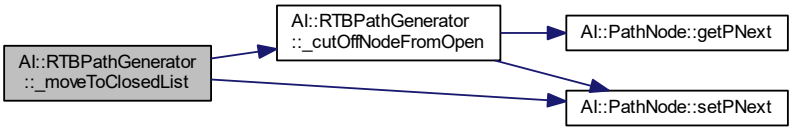
```
void AI::RTBPathGenerator::_moveToClosedList (
            PathNode *& node )  [private]
```

Adds node to closed list which keeps track of nodes that have already been examined

Parameters

node	new node

Here is the call graph for this function:



6.27.2.12 _NeighbourPosition()

```
void AI::RTBPathGenerator::_NeighbourPosition (
            unsigned short i,
            unsigned short j,
            unsigned short points[] )  [private]
```

Gets information about 8 closest nodes near the examined node.

6.27.2.13 distance()

```
int AI::RTBPathGenerator::distance (
            sf::Vector2i start,
            sf::Vector2i end )
```

@rateurn distance between start and end

Parameters

start	starting point
end	destination

6.27.2.14 findPath()

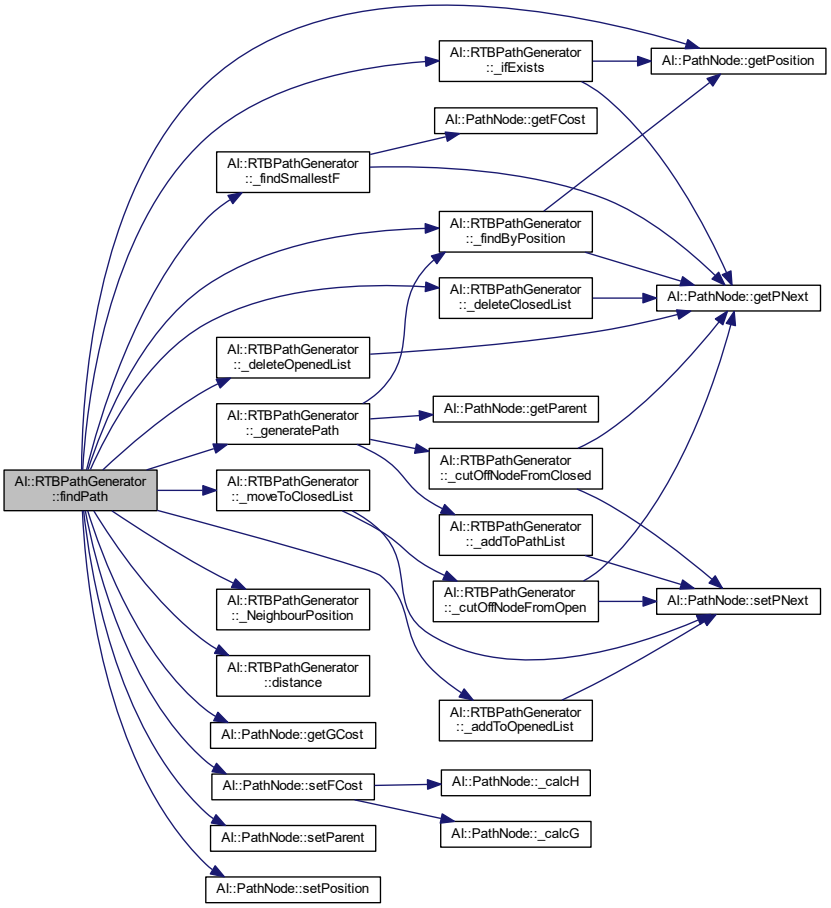
```
void AI::RTBPathGenerator::findPath (
            sf::Vector2i start,
            sf::Vector2i end )
```

Performs process of finding path from start to end

Parameters

start	
end	

Here is the call graph for this function:



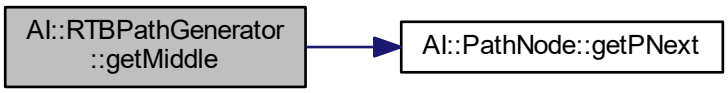
6.27.2.15 getMiddle()

```
PathNode * AI::RTBPathGenerator::getMiddle ( )
```

Returns

middle element of path list

Here is the call graph for this function:



6.27.2.16 getPath()

```
PathNode *& AI::RTBPathGenerator::getPath ( )
```

Returns

path ready to be implemented

6.27.3 Member Data Documentation

6.27.3.1 _closed_list_head

```
PathNode * AI::RTBPathGenerator::_closed_list_head  [private]
```

6.27.3.2 _end

```
sf::Vector2i AI::RTBPathGenerator::_end  [private]
```

6.27.3.3 _height

```
unsigned int AI::RTBPathGenerator::_height  [private]
```

6.27.3.4 _opened_list_head

```
PathNode* AI::RTBPathGenerator::_opened_list_head  [private]
```

6.27.3.5 _path

```
PathNode* AI::RTBPathGenerator::_path  [private]
```

6.27.3.6 _start

```
sf::Vector2i AI::RTBPathGenerator::_start  [private]
```

6.27.3.7 _walkable_area

```
std::vector<std::vector<PathNode> > AI::RTBPathGenerator::_walkable_area  [private]
```

6.27.3.8 _width

```
unsigned int AI::RTBPathGenerator::_width  [private]
```

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

- RTBPathGenerator.h
- RTBPathGenerator.cpp

6.28 RTB::SpearHitbox Class Reference

```
#include <SpearHitbox.h>
```

Public Member Functions

- SpearHitbox (sf::Sprite *&object)
- ~SpearHitbox ()
- bool checkIntersection (const sf::FloatRect &rectangle)

  Checks if hitbox collides with another object's hitbox given as rectangle.
- void update (short direction)

  Updates hitbox position.
- void render (sf::RenderTarget &window)

  Renders hitbox.

Private Types

- enum directions { up, left, down, right }

Private Attributes

- sf::RectangleShape _hitbox
- sf::Sprite * _object
- sf::Vector2f _offset

6.28.1 Member Enumeration Documentation

6.28.1.1 directions

```
enum RTB::SpearHitbox::directions  [private]
```

Enumerator

up	
left	
down	
right	

6.28.2 Constructor & Destructor Documentation

6.28.2.1 SpearHitbox()

```
RTB::SpearHitbox::SpearHitbox (
            sf::Sprite *& object )
```

6.28.2.2 ~SpearHitbox()

```
RTB::SpearHitbox::~SpearHitbox ( )
```

6.28.3 Member Function Documentation

6.28.3.1 checkIntersection()

```
bool RTB::SpearHitbox::checkIntersection (
            const sf::FloatRect & rectangle )
```

Checks if hitbox collides with another object's hitbox given as rectangle.

6.28.3.2 render()

```
void RTB::SpearHitbox::render (
            sf::RenderTarget & window )
```

Renders hitbox.

6.28.3.3 update()

```
void RTB::SpearHitbox::update (
            short direction )
```

Updates hitbox position.

6.28.4 Member Data Documentation

6.28.4.1 _hitbox

```
sf::RectangleShape RTB::SpearHitbox::_hitbox  [private]
```

6.28.4.2 _object

```
sf::Sprite* RTB::SpearHitbox::_object  [private]
```

6.28.4.3 _offset

```
sf::Vector2f RTB::SpearHitbox::_offset  [private]
```

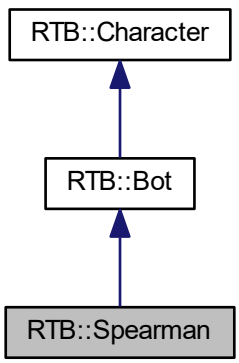
The documentation for this class was generated from the following files:

- SpearHitbox.h
- SpearHitbox.cpp

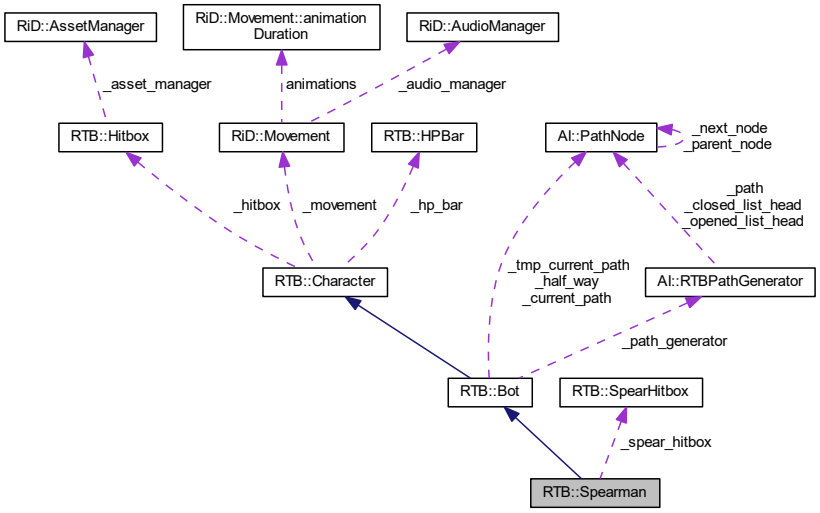
6.29 RTB::Spearman Class Reference

```
#include <Spearman.h>
```

Inheritance diagram for RTB::Spearman:



Collaboration diagram for RTB::Spearman:



Public Member Functions

- Spearman (sf::Texture texture, short health_points, std::vector< std::vector< AI::PathNode >> &walkable↩ _area)
- ~Spearman ()
- void update (sf::Time time, std::vector< std::vector< std::unique_ptr< MapElement >>> &map_objects, std::list< std::shared_ptr< Character >> &list_of_bots, sf::RenderWindow &window)
- void dealDamage (sf::Time time, std::list< std::shared_ptr< Character >> &list_of_bots, sf::RenderTarget &window)

Private Member Functions

- void _dealSpearDamage (std::list< std::shared_ptr< Character >> &list_of_bots)

Private Attributes

- SpearHitbox * _spear_hitbox

Additional Inherited Members

6.29.1 Constructor & Destructor Documentation

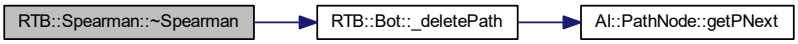
6.29.1.1 Spearman()

```
RTB::Spearman::Spearman (
            sf::Texture texture,
            short health_points,
            std::vector< std::vector< AI::PathNode >> & walkable_area )
```

6.29.1.2 ~Spearman()

```
RTB::Spearman::~Spearman ( )
```

Here is the call graph for this function:



6.29.2 Member Function Documentation

6.29.2.1 _dealSpearDamage()

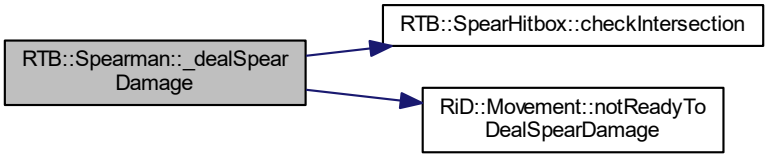
```
void RTB::Spearman::_dealSpearDamage (
            std::list< std::shared_ptr< Character >> & list_of_bots )  [private]
```

Function responsible for dealing damage to enemies

Parameters

list_of_bots	list of possible enemies

Here is the call graph for this function:



6.29.2.2 dealDamage()

```
void RTB::Spearman::dealDamage (
            sf::Time time,
            std::list< std::shared_ptr< Character >> & list_of_bots,
            sf::RenderTarget & window )  [virtual]
```

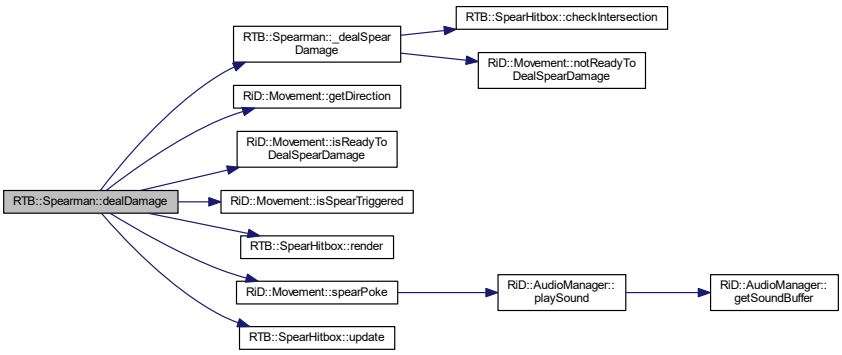
Dealing damage to Swordsmans of enemy team

Parameters

time	time needed for combat animations
list_of_Swordsmans	list of possible enemies
window	render window

Implements RTB::Character.

Here is the call graph for this function:



6.29.2.3 update()

```
void RTB::Spearman::update (
            sf::Time time,
            std::vector< std::vector< std::unique_ptr< MapElement >>> & map_objects,
            std::list< std::shared_ptr< Character >> & list_of_bots,
            sf::RenderWindow & window )  [virtual]
```

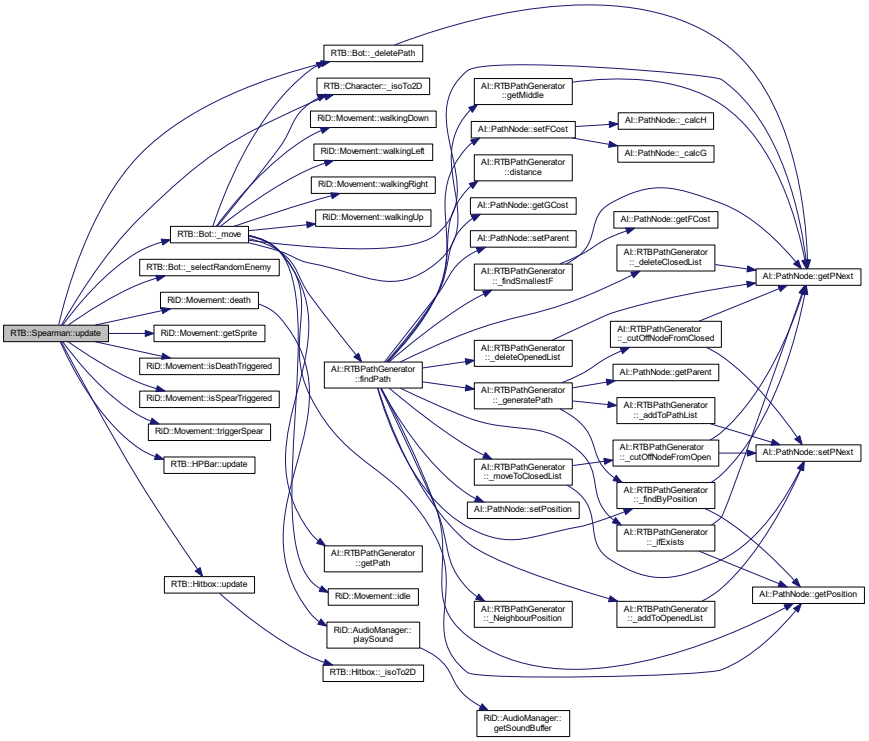
Function responsible for all of the behaviours

Parameters

time	game time
map_objects	all collidable objects to avoid
list_of_bots	list of bots given as target to attack

Implements RTB::Character.

Here is the call graph for this function:



6.29.3 Member Data Documentation

6.29.3.1 _spear_hitbox

```
SpearHitbox* RTB::Spearman::_spear_hitbox  [private]
```

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

- Spearman.h
- Spearman.cpp

6.30 RTB::SwordHitbox Class Reference

```
#include <SwordHitbox.h>
```

Public Member Functions

- SwordHitbox (sf::Sprite *&object)
- ~SwordHitbox ()
- bool checkIntersection (const sf::FloatRect &rectangle)

  Checks if hitbox collides with another object's hitbox given as rectangle.
- void update (short direction)

  Updates hitbox position.
- void render (sf::RenderTarget &window)

  Renders hitbox.

Private Types

- enum directions { up, left, down, right }

Private Attributes

- sf::RectangleShape _hitbox
- sf::Sprite * _object
- sf::Vector2f _offset

6.30.1 Member Enumeration Documentation

6.30.1.1 directions

```
enum RTB::SwordHitbox::directions  [private]
```

Enumerator

up	
left	
down	
right	

6.30.2 Constructor & Destructor Documentation

6.30.2.1 SwordHitbox()

```
RTB::SwordHitbox::SwordHitbox (
            sf::Sprite *& object )
```

6.30.2.2 ~SwordHitbox()

```
RTB::SwordHitbox::~SwordHitbox ( )
```

6.30.3 Member Function Documentation

6.30.3.1 checkIntersection()

```
bool RTB::SwordHitbox::checkIntersection (
            const sf::FloatRect & rectangle )
```

Checks if hitbox collides with another object's hitbox given as rectangle.

6.30.3.2 render()

```
void RTB::SwordHitbox::render (
            sf::RenderTarget & window )
```

Renders hitbox.

6.30.3.3 update()

```
void RTB::SwordHitbox::update (
            short direction )
```

Updates hitbox position.

6.30.4 Member Data Documentation

6.30.4.1 _hitbox

```
sf::RectangleShape RTB::SwordHitbox::_hitbox  [private]
```

6.30.4.2 _object

```
sf::Sprite* RTB::SwordHitbox::_object  [private]
```

6.30.4.3 _offset

```
sf::Vector2f RTB::SwordHitbox::_offset  [private]
```

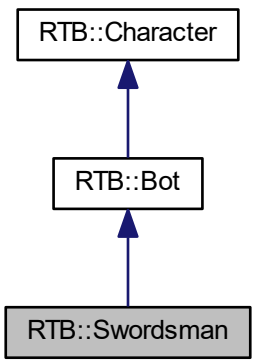
The documentation for this class was generated from the following files:

- SwordHitbox.h
- SwordHitbox.cpp

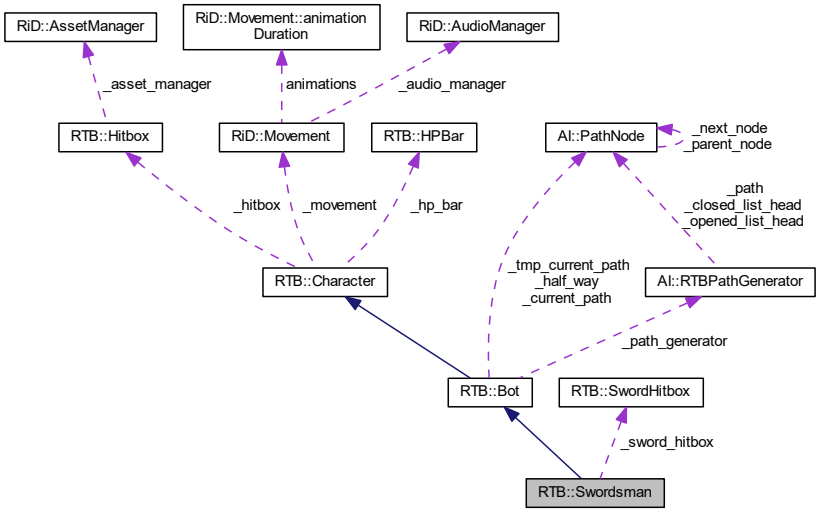
6.31 RTB::Swordsman Class Reference

```
#include <Swordsman.h>
```

Inheritance diagram for RTB::Swordsman:



Collaboration diagram for RTB::Swordsman:



Public Member Functions

- Swordsman (sf::Texture texture, short health_points, std::vector< std::vector< AI::PathNode >> &walkable_area)
- ~Swordsman ()
- void update (sf::Time time, std::vector< std::vector< std::unique_ptr< MapElement >>> &map_objects, std::list< std::shared_ptr< Character >> &list_of_bots, sf::RenderWindow &window)
- void dealDamage (sf::Time time, std::list< std::shared_ptr< Character >> &list_of_bots, sf::RenderTarget &window)

Private Member Functions

- void _dealSwordDamage (std::list< std::shared_ptr< Character >> &list_of_bots)

Private Attributes

- SwordHitbox * _sword_hitbox

Additional Inherited Members

6.31.1 Constructor & Destructor Documentation

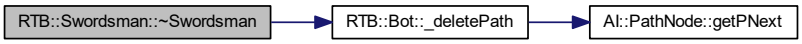
6.31.1.1 Swordsman()

```
RTB::Swordsman::Swordsman (
            sf::Texture texture,
            short health_points,
            std::vector< std::vector< AI::PathNode >> & walkable_area )
```

6.31.1.2 ~Swordsman()

```
RTB::Swordsman::~Swordsman ( )
```

Here is the call graph for this function:



6.31.2 Member Function Documentation

6.31.2.1 _dealSwordDamage()

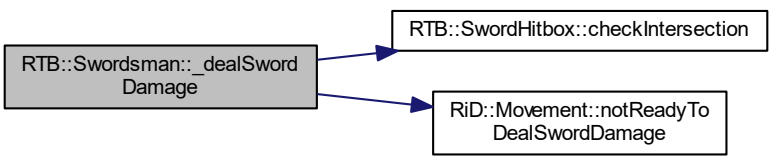
```
void RTB::Swordsman::_dealSwordDamage (
            std::list< std::shared_ptr< Character >> & list_of_bots ) [private]
```

Function responsible for dealing damage to enemies

Parameters

list_of_bots	list of possible enemies

Here is the call graph for this function:



6.31.2.2 dealDamage()

```
void RTB::Swordsman::dealDamage (
            sf::Time time,
            std::list< std::shared_ptr< Character >> & list_of_bots,
            sf::RenderTarget & window )  [virtual]
```

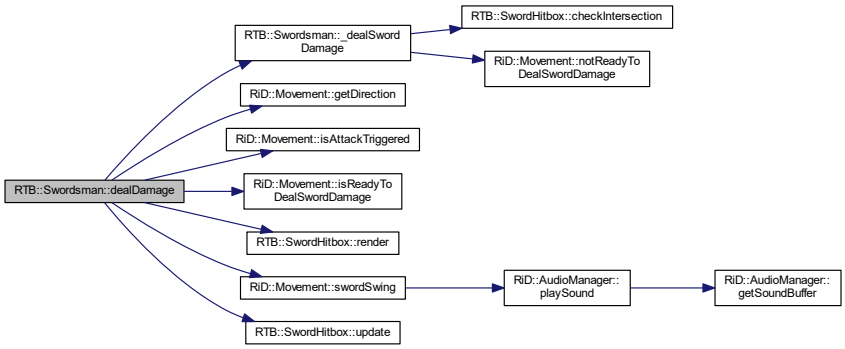
Dealing damage to Swordsmans of enemy team

Parameters

time	time needed for combat animations
list_of_Swordsmans	list of possible enemies
window	render window

Implements RTB::Character.

Here is the call graph for this function:



6.31.2.3 update()

```
void RTB::Swordsman::update (
            sf::Time time,
            std::vector< std::vector< std::unique_ptr< MapElement >>> & map_objects,
            std::list< std::shared_ptr< Character >> & list_of_bots,
            sf::RenderWindow & window )  [virtual]
```

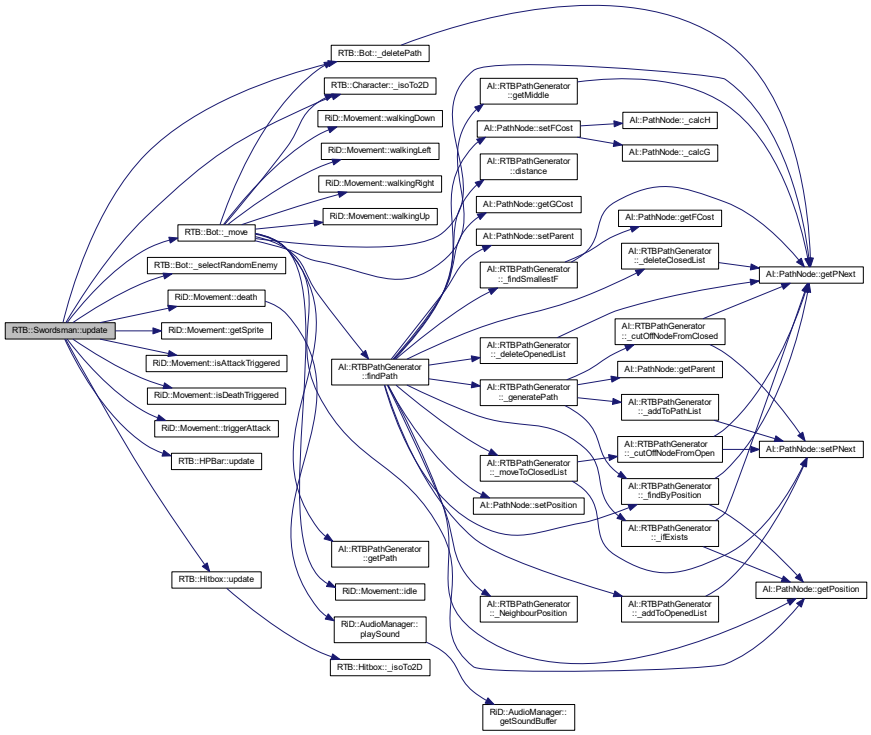
Function responsible for all of the behaviours

Parameters

time	game time
map_objects	all collidable objects to avoid
list_of_bots	list of bots given as target to attack

Implements RTB::Character.

Here is the call graph for this function:



6.31.3 Member Data Documentation

6.31.3.1 _sword_hitbox

```
SwordHitbox* RTB::Swordsman::_sword_hitbox  [private]
```

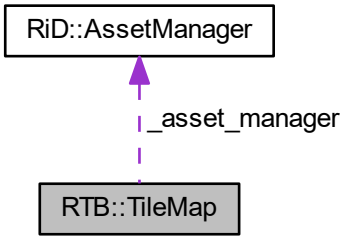
The documentation for this class was generated from the following files:

- Swordsman.h
- Swordsman.cpp

6.32 RTB::TileMap Class Reference

```
#include <TileMap.h>
```

Collaboration diagram for RTB::TileMap:



Public Member Functions

- TileMap (sf::Vector2i tile_size)
- void drawTiles (sf::RenderTarget &window)

  Draws tiles: grass, dirt, etc. and flowers.
- void drawObjects (sf::RenderTarget &window)

  Draws collidable objects: trees, fences, etc.
- std::vector< std::vector< std::unique_ptr< MapElement > > > & getCollidableObjects ()
- std::vector< std::vector< AI::PathNode > > & getWalkableArea ()
- unsigned int & getWidth ()
- unsigned int & getHeight ()
- sf::Vector2f _twoDToIso (sf::Vector2f position)
- TileMap (sf::Vector2i tile_size)
- void drawTiles (sf::RenderTarget &window)
- void drawObjects (sf::RenderTarget &window)
- std::vector< std::vector< std::unique_ptr< MapElement > > > & getCollidableObjects ()
- std::vector< std::vector< AI::PathNode > > & getWalkableArea ()
- unsigned int & getWidth ()
- unsigned int & getHeight ()
- sf::Vector2f _twoDToIso (sf::Vector2f position)

Private Types

- enum _tiles {
  dirt, grass, water, road,
  dirt, grass, water, road }
- enum _flora {
  no_flora, tinyFlower, redFlower, fence1Fallen,
  no_flora, tinyFlower, redFlower, fence1Fallen }
- enum _Collidable_objects {
  no_object, fence1, sign, tree,
  chest, no_object, fence1, sign,
  tree, chest }
- enum _tiles {
  dirt, grass, water, road,
  dirt, grass, water, road }

- enum _flora {
  no_flora, tinyFlower, redFlower, fence1Fallen,
  no_flora, tinyFlower, redFlower, fence1Fallen }
- enum _Collidable_objects {
  no_object, fence1, sign, tree,
  chest, no_object, fence1, sign,
  tree, chest }

Private Member Functions

- sf::Vector2f _twoDToIso ()
- void _loadFromFile (std::string map_file_name, std::string flora_file_name, std::string objects_file_name)

  Loads from files info about 3 layers: tiles, flora and objects.
- void _placeTile (unsigned short _position_x, unsigned short _position_y)

  Assigns tile sprite on certain position.
- void _placeFlora (unsigned short _position_x, unsigned short _position_y)

  Assigns flora sprite on certain position.
- void _placeObjects (unsigned short _position_x, unsigned short _position_y)

  Assigns object sprite on certain position.
- void _generateWalkableArea ()

  Generates walkable area.
- sf::Vector2f _twoDToIso ()
- void _loadFromFile (std::string map_file_name, std::string flora_file_name, std::string objects_file_name)
- void _placeTile (unsigned short _position_x, unsigned short _position_y)
- void _placeFlora (unsigned short _position_x, unsigned short _position_y)
- void _placeObjects (unsigned short _position_x, unsigned short _position_y)
- void _generateWalkableArea ()

Private Attributes

- RiD::AssetManager _asset_manager
- std::vector< std::vector< unsigned int > > _level
- std::vector< std::vector< unsigned int > > _flora
- std::vector< std::vector< unsigned int > > _objects
- std::vector< std::vector< std::unique_ptr< MapElement > > > _map_elements
- std::vector< std::vector< AI::PathNode > > _walkable_area
- unsigned int _width
- unsigned int _height
- unsigned int _tile_type
- sf::Vector2f _point

6.32.1 Member Enumeration Documentation

6.32.1.1 _Collidable_objects [1/2]

```
enum RTB::TileMap::_Collidable_objects  [private]
```

Enumerator

no_object	
fence1	
sign	
tree	
chest	
no_object	
fence1	
sign	
tree	
chest	

6.32.1.2 _Collidable_objects [2/2]

`enum RTB::TileMap::_Collidable_objects [private]`

Enumerator

no_object	
fence1	
sign	
tree	
chest	
no_object	
fence1	
sign	
tree	
chest	

6.32.1.3 _flora [1/2]

`enum RTB::TileMap::_flora [private]`

Enumerator

no_flora	
tinyFlower	
redFlower	
fence1Fallen	
no_flora	
tinyFlower	
redFlower	
fence1Fallen	

6.32.1.4 _flora [2/2]

enum std::vector< std::vector< unsigned int > > RTB::TileMap::_flora [private]

Enumerator

no_flora	
tinyFlower	
redFlower	
fence1Fallen	
no_flora	
tinyFlower	
redFlower	
fence1Fallen	

6.32.1.5 _tiles [1/2]

enum RTB::TileMap::_tiles [private]

Enumerator

dirt	
grass	
water	
road	
dirt	
grass	
water	
road	

6.32.1.6 _tiles [2/2]

enum RTB::TileMap::_tiles [private]

Enumerator

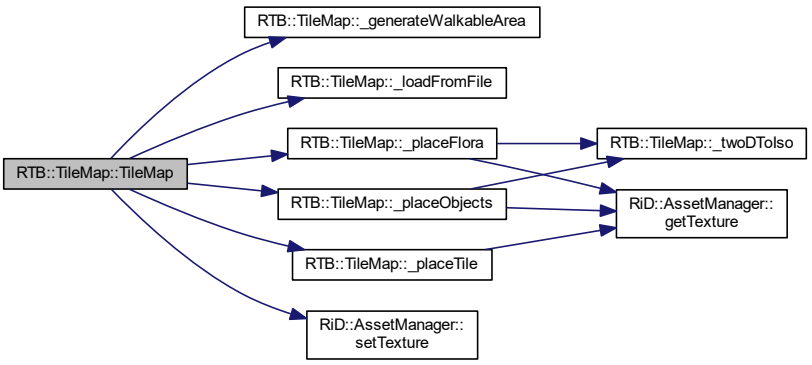
dirt	
grass	
water	
road	
dirt	
grass	
water	
road	

6.32.2 Constructor & Destructor Documentation

6.32.2.1 TileMap() [1/2]

```
RTB::TileMap::TileMap (
            sf::Vector2i tile_size )
```

Here is the call graph for this function:



6.32.2.2 TileMap() [2/2]

```
RTB::TileMap::TileMap (
            sf::Vector2i tile_size )
```

6.32.3 Member Function Documentation

6.32.3.1 _generateWalkableArea() [1/2]

```
void RTB::TileMap::_generateWalkableArea ( )  [private]
```

Generates walkable area.

6.32.3.2 _generateWalkableArea() [2/2]

```
void RTB::TileMap::_generateWalkableArea ( )  [private]
```

6.32.3.3 _loadFromFile() [1/2]

```
void RTB::TileMap::_loadFromFile (
            std::string map_file_name,
            std::string flora_file_name,
            std::string objects_file_name )  [private]
```

Loads from files info about 3 layers: tiles, flora and objects.

6.32.3.4 _loadFromFile() [2/2]

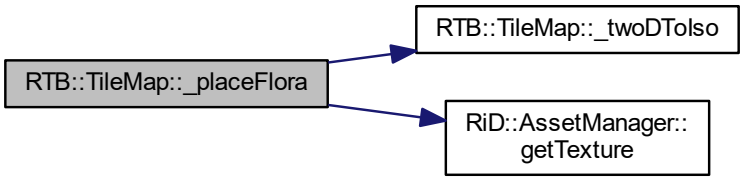
```
void RTB::TileMap::_loadFromFile (
            std::string map_file_name,
            std::string flora_file_name,
            std::string objects_file_name )  [private]
```

6.32.3.5 _placeFlora() [1/2]

```
void RTB::TileMap::_placeFlora (
            unsigned short _position_x,
            unsigned short _position_y )  [private]
```

Assigns flora sprite on certain position.

Here is the call graph for this function:



6.32.3.6 _placeFlora() [2/2]

```
void RTB::TileMap::_placeFlora (
            unsigned short _position_x,
            unsigned short _position_y )  [private]
```

6.32.3.7 _placeObjects() [1/2]

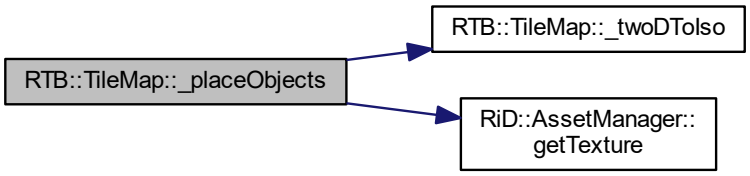
```
void RTB::TileMap::_placeObjects (
            unsigned short _position_x,
            unsigned short _position_y )  [private]
```

6.32.3.8 _placeObjects() [2/2]

```
void RTB::TileMap::_placeObjects (
            unsigned short _position_x,
            unsigned short _position_y )  [private]
```

Assigns object sprite on certain position.

Here is the call graph for this function:



6.32.3.9 _placeTile() [1/2]

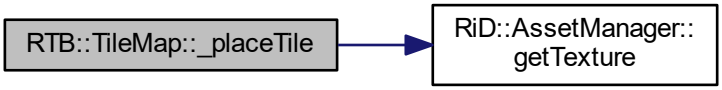
```
void RTB::TileMap::_placeTile (
            unsigned short _position_x,
            unsigned short _position_y )  [private]
```

6.32.3.10 _placeTile() [2/2]

```
void RTB::TileMap::_placeTile (
            unsigned short _position_x,
            unsigned short _position_y )  [private]
```

Assigns tile sprite on certain position.

Here is the call graph for this function:



6.32.3.11 _twoDToIso() [1/4]

```
sf::Vector2f RTB::TileMap::_twoDToIso ( )  [private]
```

6.32.3.12 _twoDToIso() [2/4]

```
sf::Vector2f RTB::TileMap::_twoDToIso ( )  [private]
```

6.32.3.13 _twoDToIso() [3/4]

```
sf::Vector2f RTB::TileMap::_twoDToIso (
            sf::Vector2f position )
```

6.32.3.14 _twoDToIso() [4/4]

```
sf::Vector2f RTB::TileMap::_twoDToIso (
            sf::Vector2f position )
```

6.32.3.15 drawObjects() [1/2]

```
void RTB::TileMap::drawObjects (
            sf::RenderTarget & window )
```

Draws collidable objects: trees, fences, etc.

6.32.3.16 drawObjects() [2/2]

```
void RTB::TileMap::drawObjects (
            sf::RenderTarget & window )
```

6.32.3.17 drawTiles() [1/2]

```
void RTB::TileMap::drawTiles (
            sf::RenderTarget & window )
```

6.32.3.18 drawTiles() [2/2]

```
void RTB::TileMap::drawTiles (
            sf::RenderTarget & window )
```

Draws tiles: grass, dirt, etc. and flowers.

6.32.3.19 getCollidableObjects() [1/2]

```
std::vector< std::vector< std::unique_ptr< MapElement > > > & RTB::TileMap::getCollidable↩
Objects ( )
```

Returns

2d vector of objects that player may collide with

6.32.3.20 getCollidableObjects() [2/2]

```
std::vector<std::vector<std::unique_ptr<MapElement> > >& RTB::TileMap::getCollidableObjects
( )
```

6.32.3.21 getHeight() [1/2]

```
unsigned int& RTB::TileMap::getHeight ( )
```

6.32.3.22 getHeight() [2/2]

```
unsigned int & RTB::TileMap::getHeight ( )
```

Returns

height of the map

6.32.3.23 getWalkableArea() [1/2]

```
std::vector<std::vector<AI::PathNode> >& RTB::TileMap::getWalkableArea ( )
```

6.32.3.24 getWalkableArea() [2/2]

```
std::vector< std::vector< AI::PathNode > > & RTB::TileMap::getWalkableArea ( )
```

Returns

2d vector of walkable area for bots

6.32.3.25 getWidth() [1/2]

```
unsigned int& RTB::TileMap::getWidth ( )
```

6.32.3.26 getWidth() [2/2]

```
unsigned int & RTB::TileMap::getWidth ( )
```

Returns

width of the map

6.32.4 Member Data Documentation

6.32.4.1 _asset_manager

```
RiD::AssetManager RTB::TileMap::_asset_manager  [private]
```

6.32.4.2 _flora

```
std::vector<std::vector<unsigned int> > RTB::TileMap::_flora  [private]
```

6.32.4.3 _height

```
unsigned int RTB::TileMap::_height  [private]
```

6.32.4.4 _level

```
std::vector< std::vector< unsigned int > > RTB::TileMap::_level  [private]
```

6.32.4.5 _map_elements

```
std::vector< std::vector< std::unique_ptr< MapElement > > > RTB::TileMap::_map_elements
[private]
```

6.32.4.6 _objects

```
std::vector< std::vector< unsigned int > > RTB::TileMap::_objects  [private]
```

6.32.4.7 _point

```
sf::Vector2f RTB::TileMap::_point  [private]
```

6.32.4.8 _tile_type

```
unsigned int RTB::TileMap::_tile_type  [private]
```

6.32.4.9 _walkable_area

```
std::vector< std::vector< AI::PathNode > > RTB::TileMap::_walkable_area  [private]
```

6.32.4.10 _width

```
unsigned int RTB::TileMap::_width  [private]
```

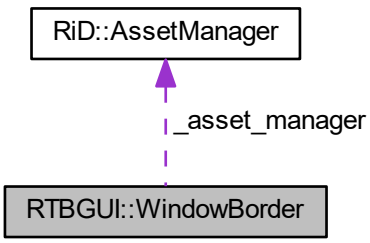
The documentation for this class was generated from the following files:

- Map/TileMap.h
- Map/TileMap.cpp

6.33 RTBGUI::WindowBorder Class Reference

```
#include <WindowBorder.h>
```

Collaboration diagram for RTBGUI::WindowBorder:



Public Member Functions

- WindowBorder ()
- void update (sf::Vector2f position)
- void render (sf::RenderWindow *&window)

Private Attributes

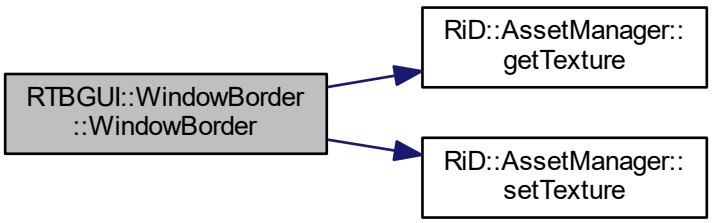
- sf::Sprite _window_border
- RiD::AssetManager _asset_manager

6.33.1 Constructor & Destructor Documentation

6.33.1.1 WindowBorder()

```
RTBGUI::WindowBorder::WindowBorder ( )
```

Here is the call graph for this function:



6.33.2 Member Function Documentation

6.33.2.1 render()

```
void RTBGUI::WindowBorder::render (
            sf::RenderWindow *& window )
```

6.33.2.2 update()

```
void RTBGUI::WindowBorder::update (
            sf::Vector2f position )
```

6.33.3 Member Data Documentation

6.33.3.1 _asset_manager

```
RiD::AssetManager RTBGUI::WindowBorder::_asset_manager  [private]
```

6.33.3.2 _window_border

```
sf::Sprite RTBGUI::WindowBorder::_window_border  [private]
```

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

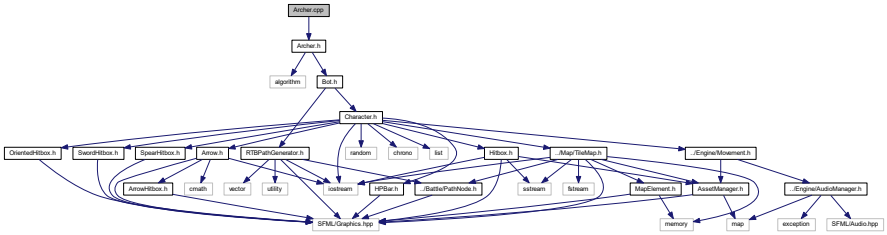
- WindowBorder.h
- WindowBorder.cpp

Chapter 7

File Documentation

7.1 Archer.cpp File Reference

```
#include "Archer.h"
```
Include dependency graph for Archer.cpp:

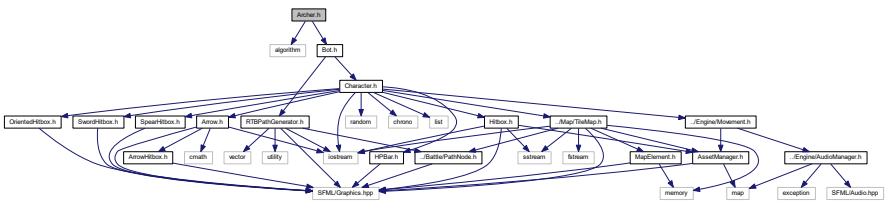


Namespaces

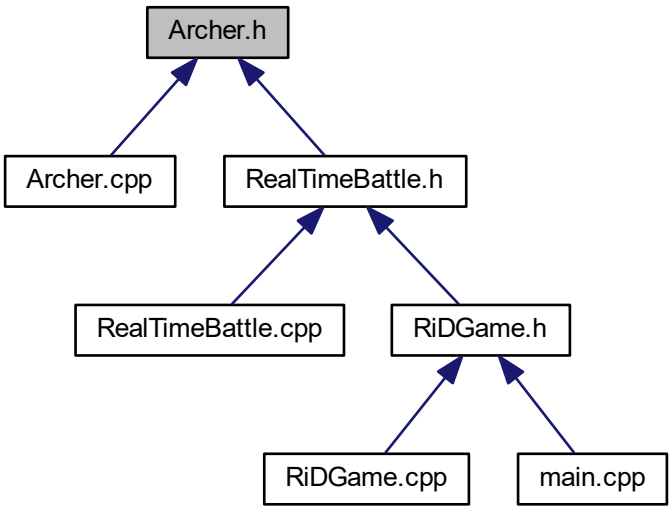
- RTB

7.2 Archer.h File Reference

```
#include <algorithm>
#include "Bot.h"
```
Include dependency graph for Archer.h:



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



Classes

- class RTB::Archer

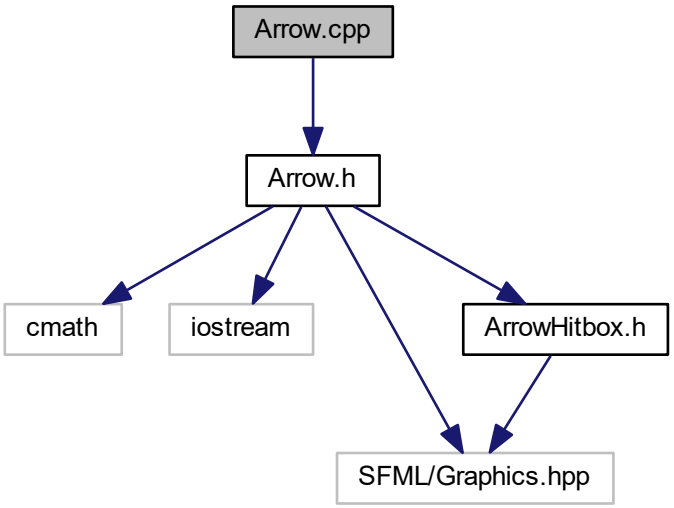
Namespaces

- RTB

7.3 Arrow.cpp File Reference

```
#include "Arrow.h"
```

Include dependency graph for Arrow.cpp:



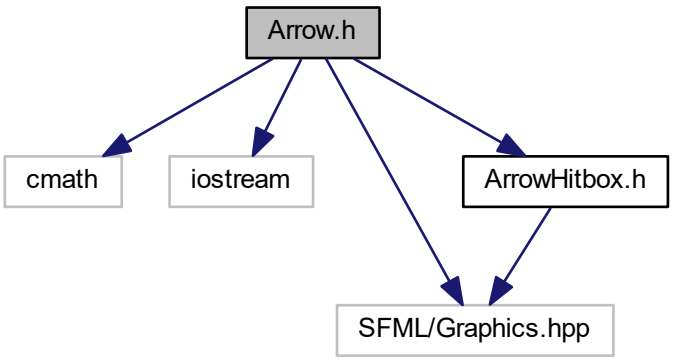
Namespaces

- RTB

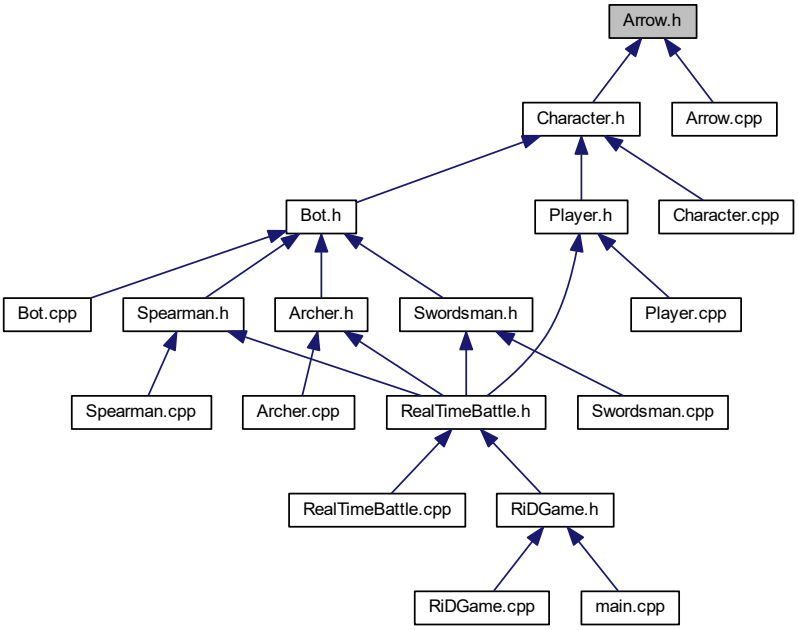
7.4 Arrow.h File Reference

```
#include <cmath>
#include <iostream>
#include "SFML/Graphics.hpp"
#include "ArrowHitbox.h"
```

Include dependency graph for Arrow.h:



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



Classes

- class RTB::Arrow

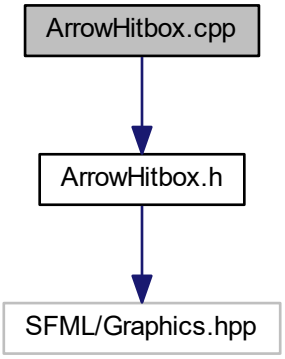
Namespaces

- RTB

7.5 ArrowHitbox.cpp File Reference

```
#include "ArrowHitbox.h"
```

Include dependency graph for ArrowHitbox.cpp:



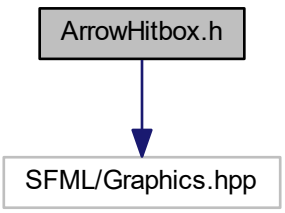
Namespaces

- RTB

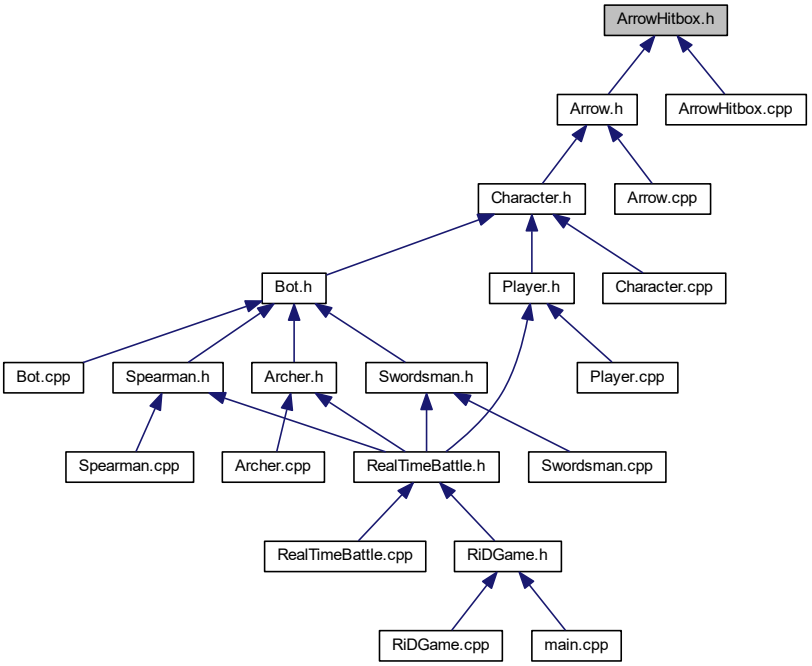
7.6 ArrowHitbox.h File Reference

```
#include "SFML/Graphics.hpp"
```

Include dependency graph for ArrowHitbox.h:



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



Classes

- class RTB::ArrowHitbox

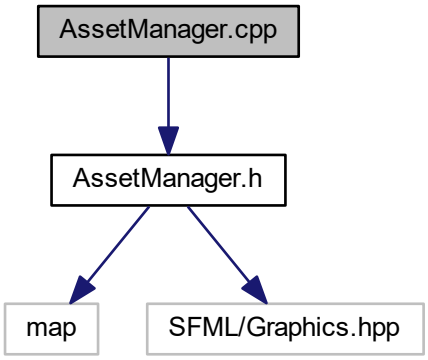
Namespaces

- RTB

7.7 AssetManager.cpp File Reference

```
#include "AssetManager.h"
```

Include dependency graph for AssetManager.cpp:



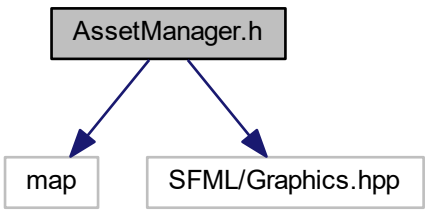
Namespaces

- RiD

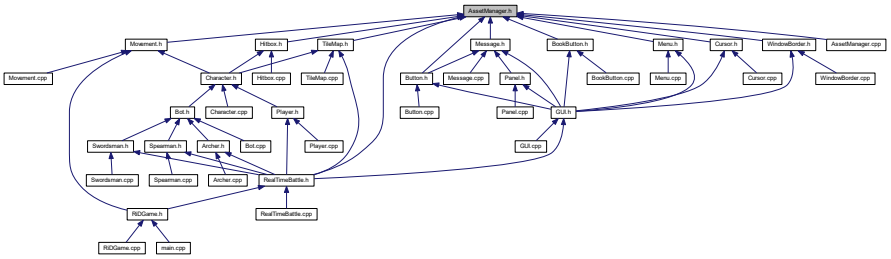
7.8 AssetManager.h File Reference

```
#include <map>
#include <SFML/Graphics.hpp>
```

Include dependency graph for AssetManager.h:



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



Classes

- class RiD::AssetManager

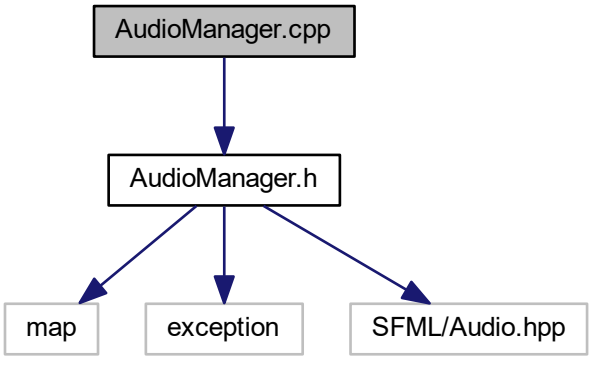
Namespaces

- RiD

7.9 AudioManager.cpp File Reference

```
#include "AudioManager.h"
```

Include dependency graph for AudioManager.cpp:



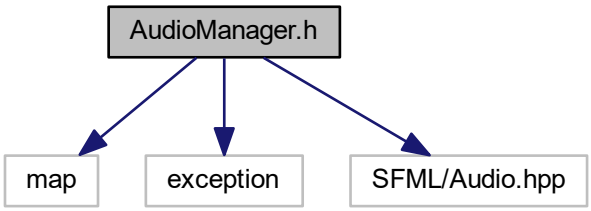
Namespaces

- RiD

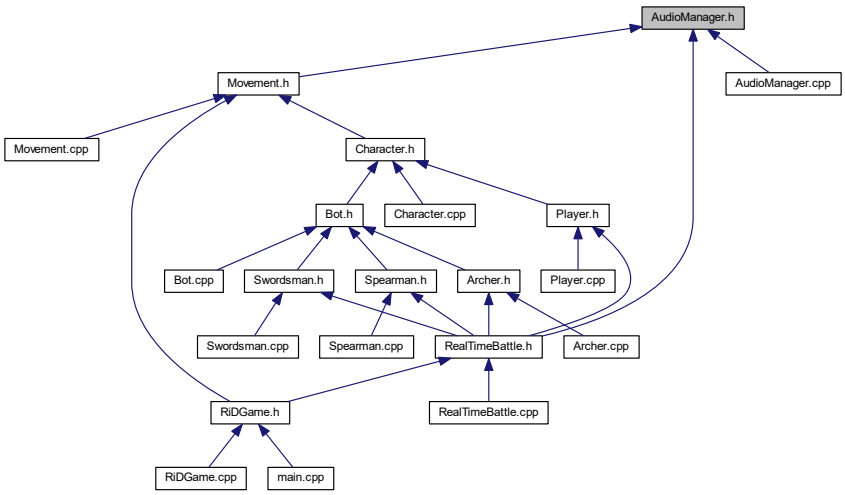
7.10 AudioManager.h File Reference

```
#include <map>
#include <exception>
#include "SFML/Audio.hpp"
```

Include dependency graph for AudioManager.h:



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



Classes

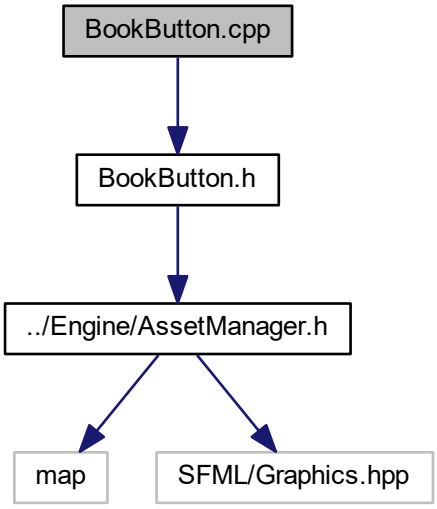
- class RiD::AudioManager

Namespaces

- RiD

7.11 BookButton.cpp File Reference

```
#include "BookButton.h"
```
Include dependency graph for BookButton.cpp:

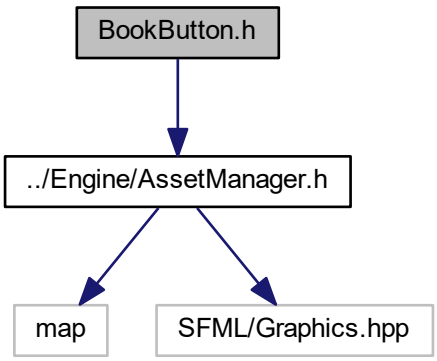


Namespaces

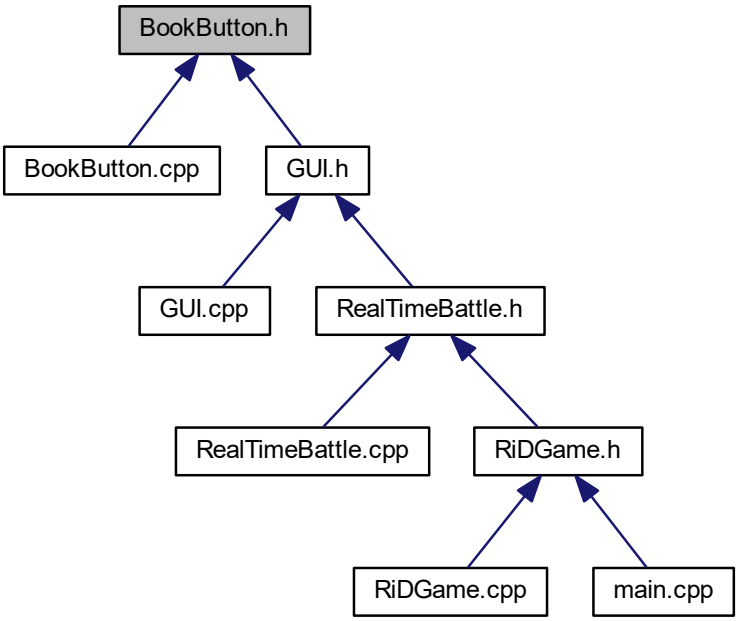
- RTBGUI

7.12 BookButton.h File Reference

```
#include "../Engine/AssetManager.h"
```
Include dependency graph for BookButton.h:



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



Classes

- class RTBGUI::BookButton

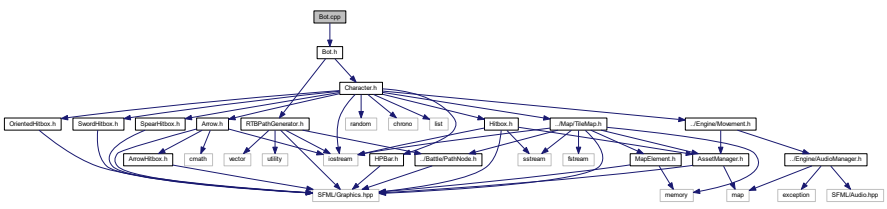
Namespaces

- RTBGUI

7.13 Bot.cpp File Reference

```
#include "Bot.h"
```

Include dependency graph for Bot.cpp:



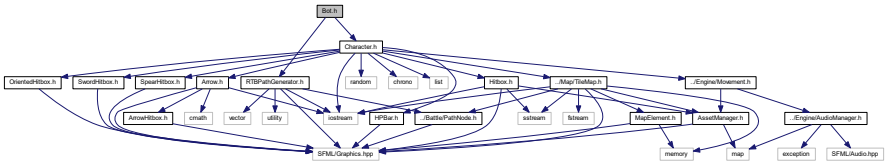
Namespaces

- RTB

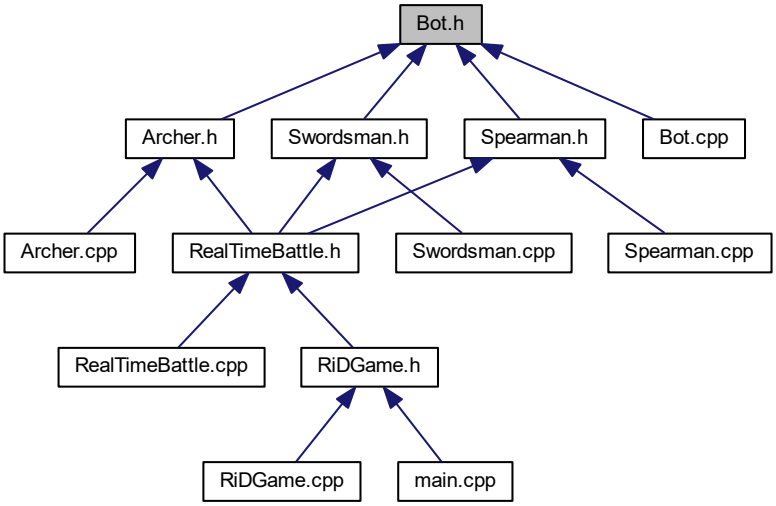
7.14 Bot.h File Reference

```
#include "Character.h"
#include "RTBPathGenerator.h"
```

Include dependency graph for Bot.h:



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



Classes

- class RTB::Bot

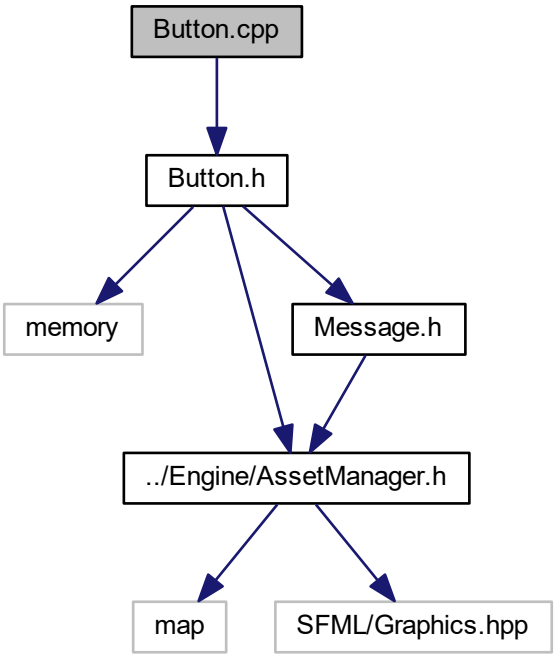
Namespaces

- RTB

7.15 Button.cpp File Reference

```
#include "Button.h"
```

Include dependency graph for Button.cpp:



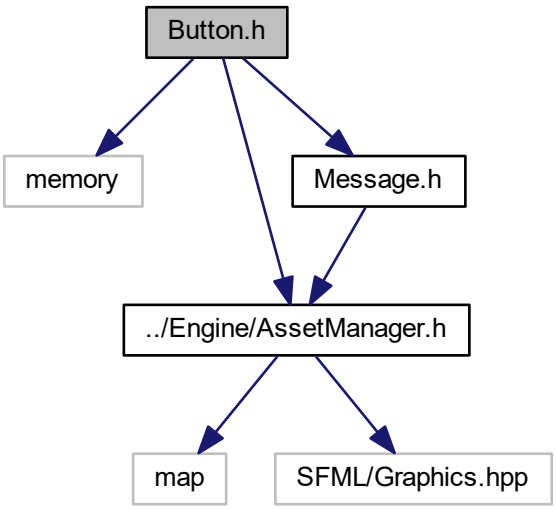
Namespaces

- RTBGUI

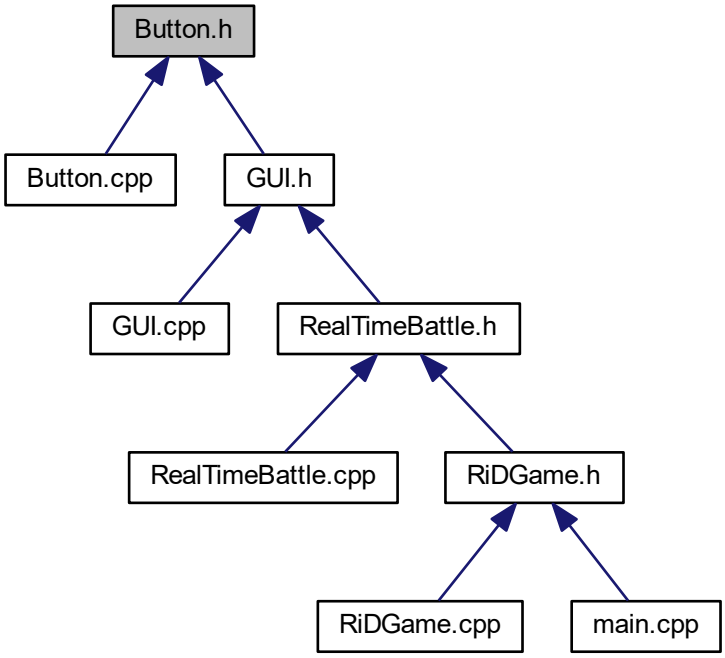
7.16 Button.h File Reference

```
#include <memory>
#include "../Engine/AssetManager.h"
#include "Message.h"
```

Include dependency graph for Button.h:



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



Classes

- class RTBGUI::Button

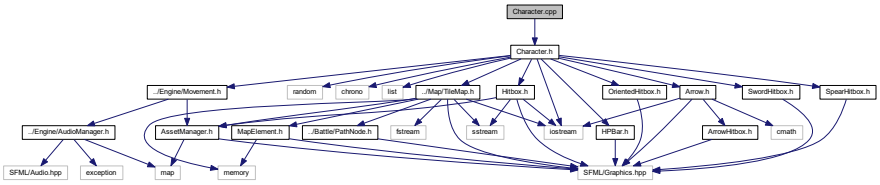
Namespaces

- RTBGUI

7.17 Character.cpp File Reference

```
#include "Character.h"
```

Include dependency graph for Character.cpp:



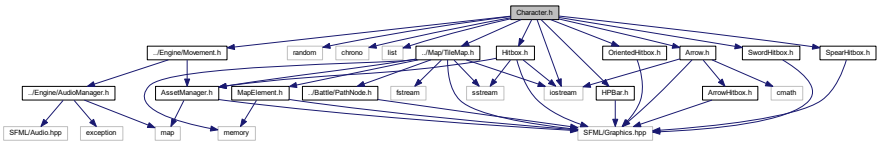
Namespaces

- RTB

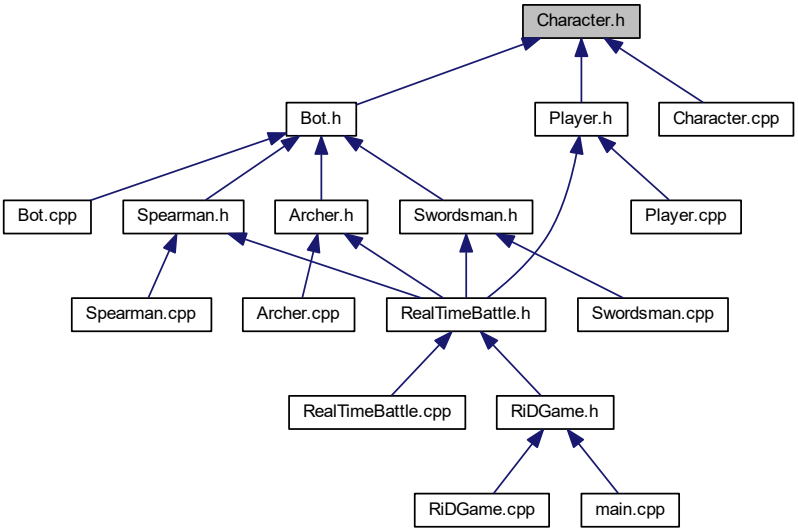
7.18 Character.h File Reference

```
#include <iostream>
#include <random>
#include <chrono>
#include <list>
#include "../Engine/Movement.h"
#include "Hitbox.h"
#include "HPBar.h"
#include "SwordHitbox.h"
#include "SpearHitbox.h"
#include "Arrow.h"
#include "../Map/TileMap.h"
#include "OrientedHitbox.h"
```

Include dependency graph for Character.h:



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



Classes

- class RTB::Character

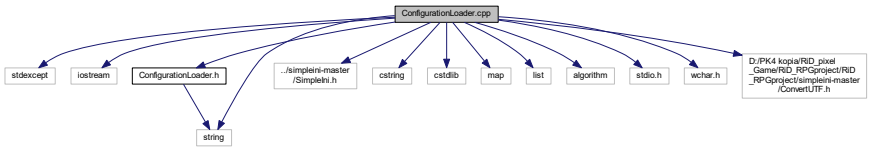
Namespaces

- RTB

7.19 ConfigurationLoader.cpp File Reference

```
#include <stdexcept>
#include <iostream>
#include "ConfigurationLoader.h"
#include "../simpleini-master/SimpleIni.h"
```

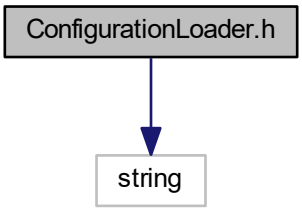
Include dependency graph for ConfigurationLoader.cpp:



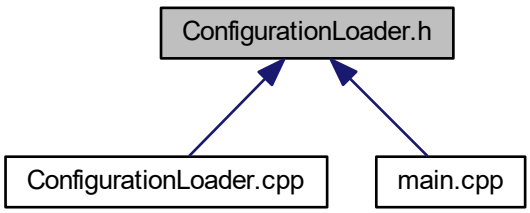
7.20 ConfigurationLoader.h File Reference

```
#include <string>
```

Include dependency graph for ConfigurationLoader.h:



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



Classes

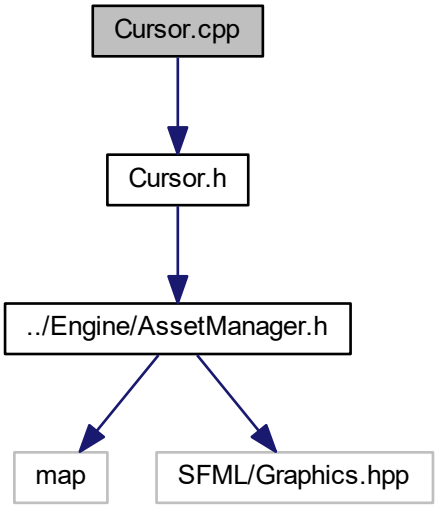
- class RiD::ConfigurationLoader

Namespaces

- RiD

7.21 Cursor.cpp File Reference

```
#include "Cursor.h"
```
Include dependency graph for Cursor.cpp:

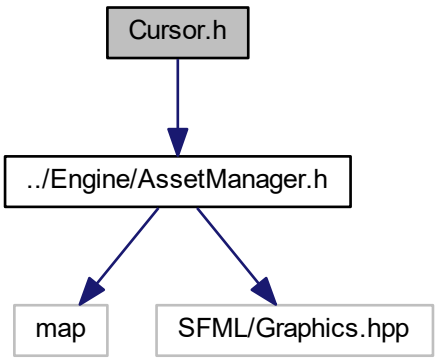


Namespaces

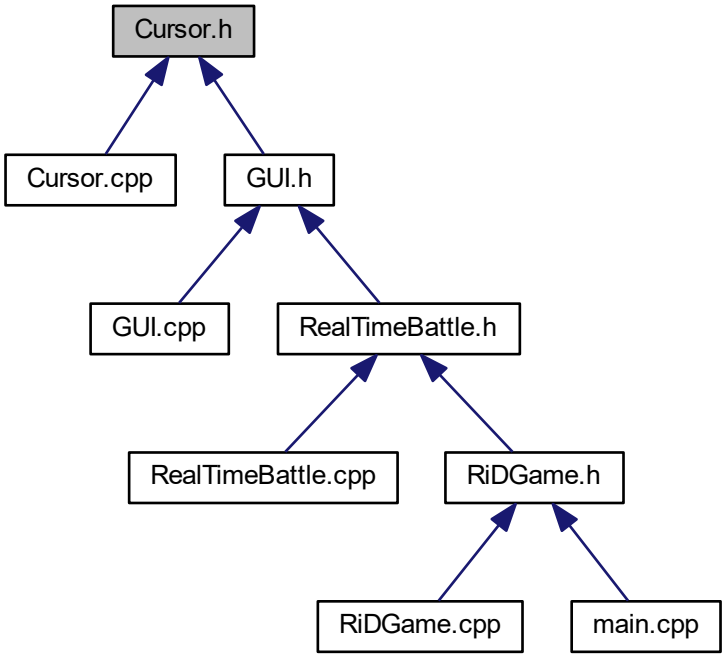
- RTBGUI

7.22 Cursor.h File Reference

```
#include "../Engine/AssetManager.h"
```
Include dependency graph for Cursor.h:



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



Classes

- class RTBGUI::Cursor

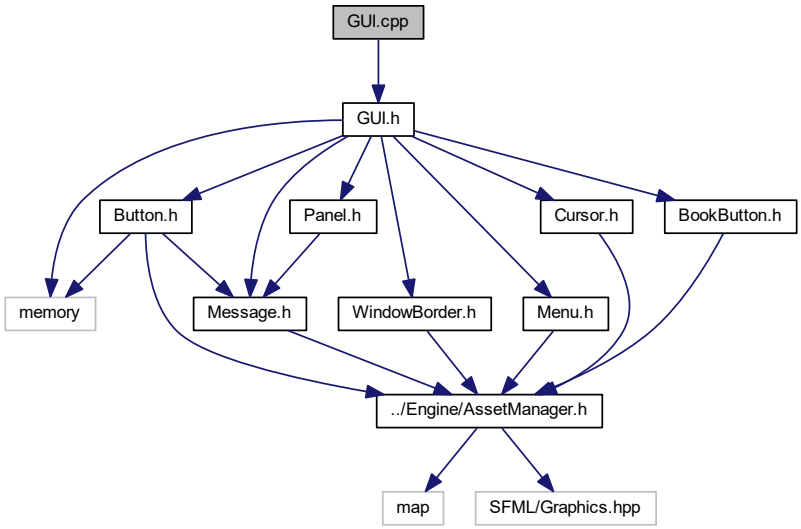
Namespaces

- RTBGUI

7.23 GUI.cpp File Reference

```
#include "GUI.h"
```

Include dependency graph for GUI.cpp:



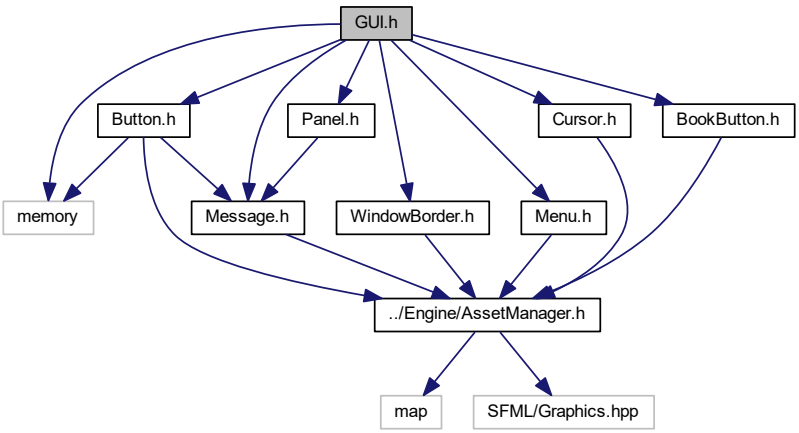
Namespaces

- RTBGUI

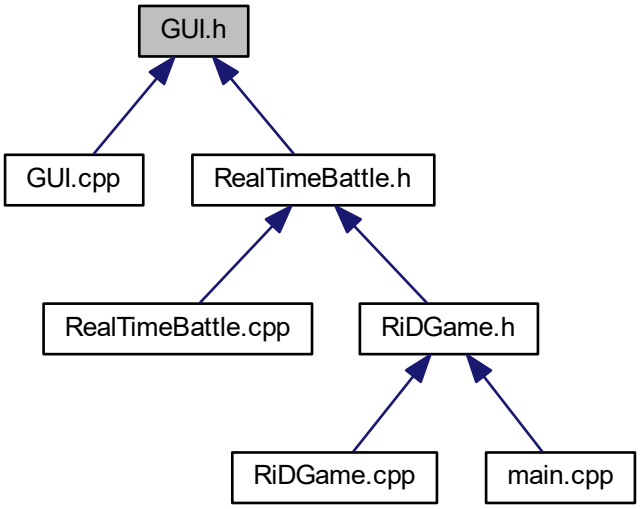
7.24 GUI.h File Reference

```
#include <memory>
#include "Button.h"
#include "WindowBorder.h"
#include "Menu.h"
#include "Message.h"
#include "Cursor.h"
#include "BookButton.h"
#include "Panel.h"
```

Include dependency graph for GUI.h:



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



Classes

- class RTBGUI::GUI

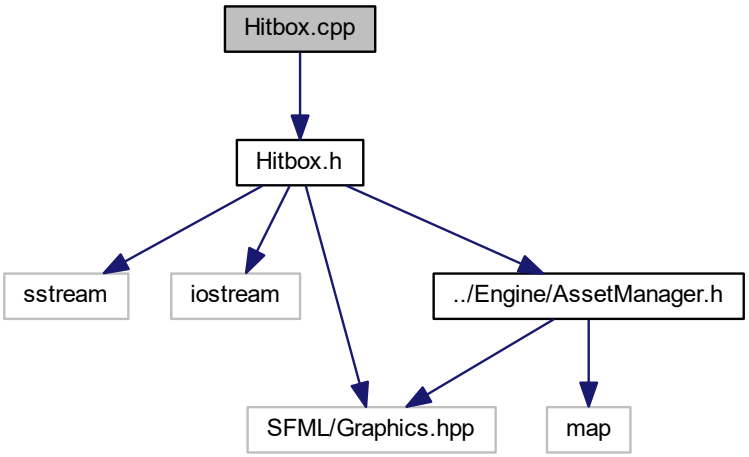
Namespaces

- RTBGUI

7.25 Hitbox.cpp File Reference

```
#include "Hitbox.h"
```

Include dependency graph for Hitbox.cpp:



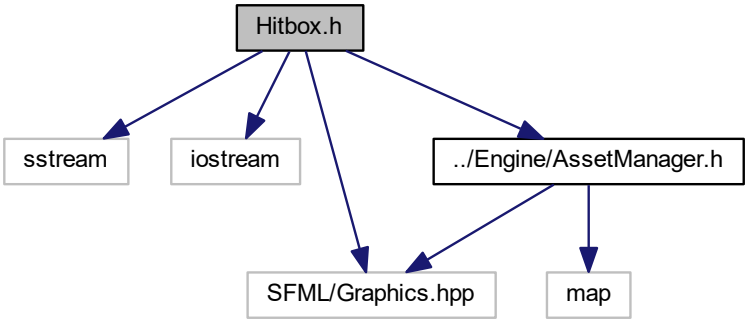
Namespaces

- RTB

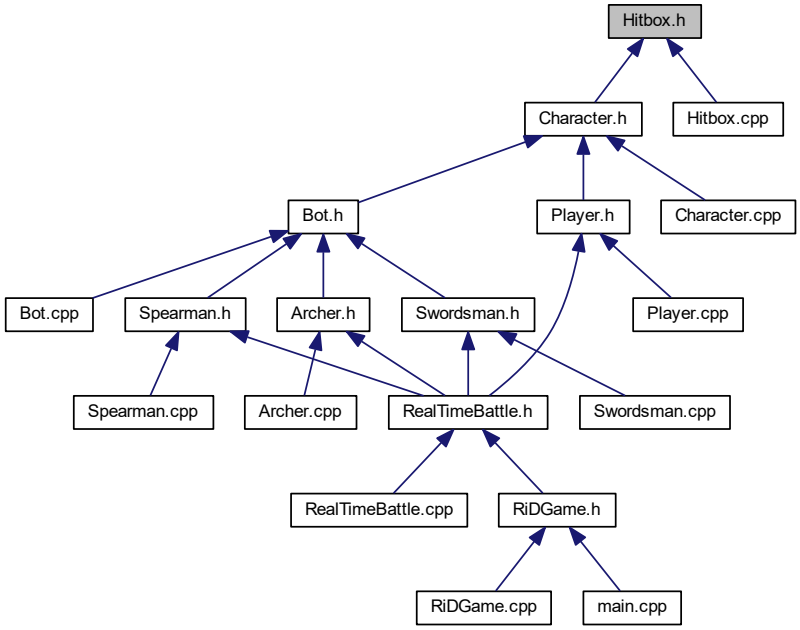
7.26 Hitbox.h File Reference

```
#include <sstream>
#include <iostream>
#include "SFML/Graphics.hpp"
#include "../Engine/AssetManager.h"
```

Include dependency graph for Hitbox.h:



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



Classes

- class RTB::Hitbox

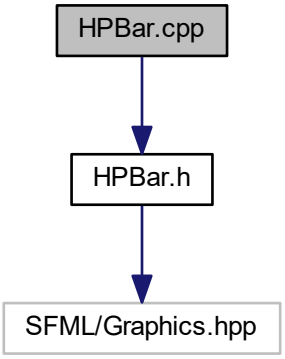
Namespaces

- RTB

7.27 HPBar.cpp File Reference

```
#include "HPBar.h"
```

Include dependency graph for HPBar.cpp:



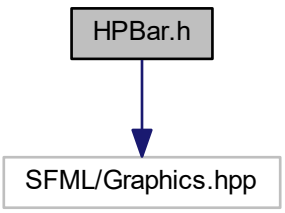
Namespaces

- RTB

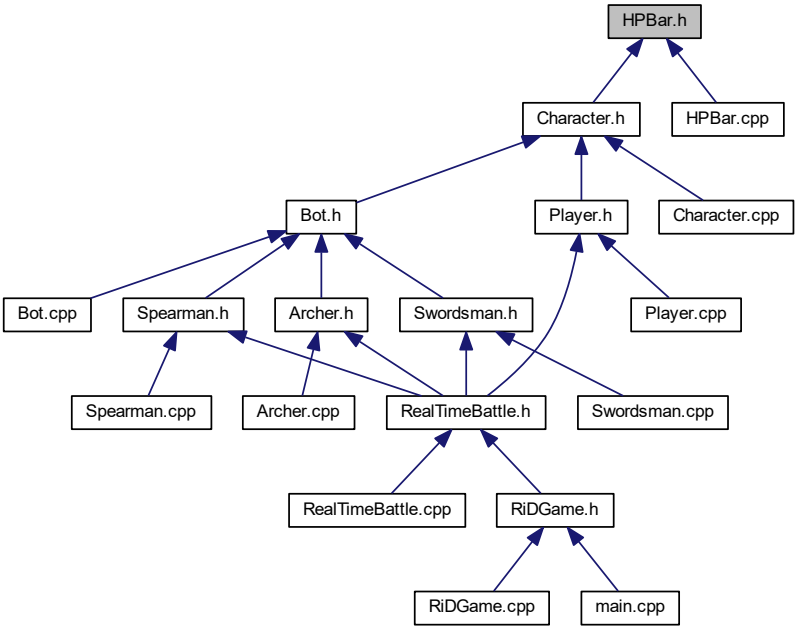
7.28 HPBar.h File Reference

```
#include "SFML/Graphics.hpp"
```

Include dependency graph for HPBar.h:



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



Classes

- class RTB::HPBar

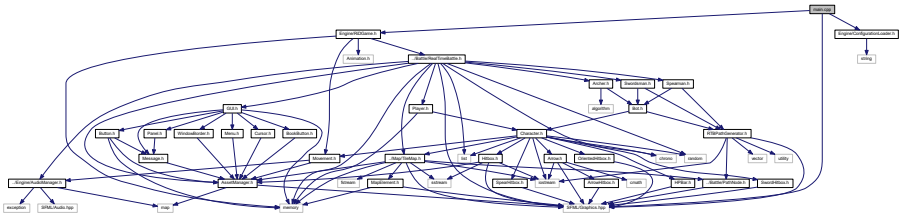
Namespaces

- RTB

7.29 main.cpp File Reference

```
#include "SFML/Graphics.hpp"
#include "Engine/RiDGame.h"
#include "Engine/ConfigurationLoader.h"
```

Include dependency graph for main.cpp:



Functions

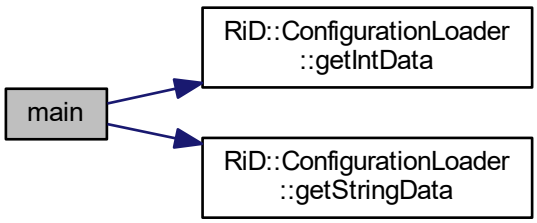
- int main ()

7.29.1 Function Documentation

7.29.1.1 main()

```
int main ( )
```

Here is the call graph for this function:



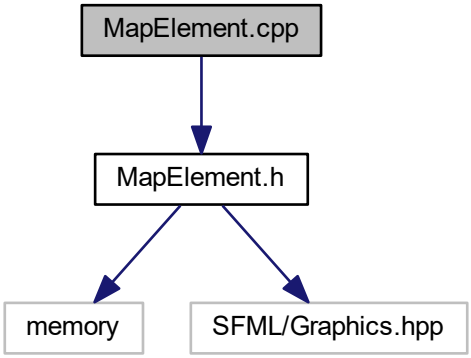
7.30 map.txt File Reference

7.31 map.txt File Reference

7.32 MapElement.cpp File Reference

```
#include "MapElement.h"
```

Include dependency graph for Map/MapElement.cpp:



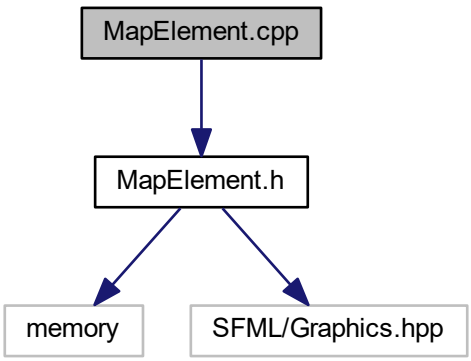
Namespaces

- RTB

7.33 MapElement.cpp File Reference

```
#include "MapElement.h"
```

Include dependency graph for Release/Map/MapElement.cpp:



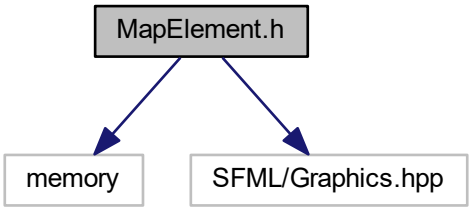
Namespaces

- RTB

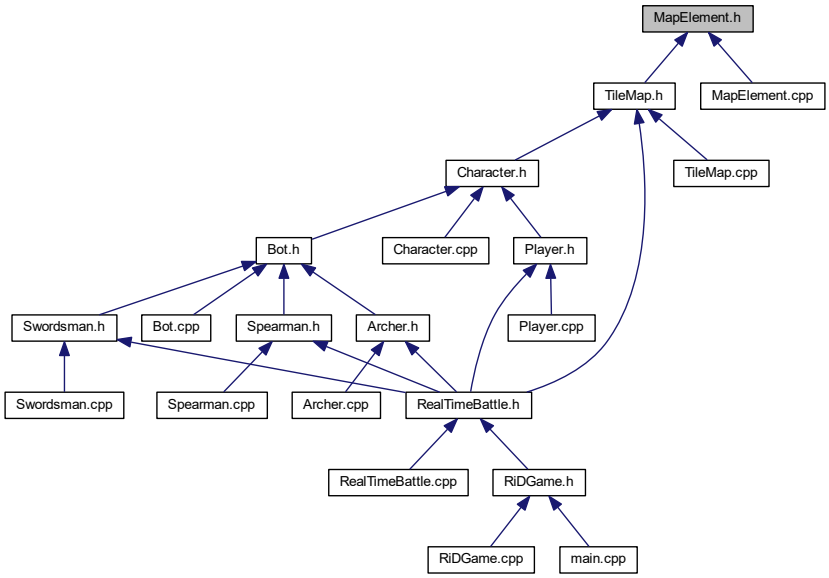
7.34 MapElement.h File Reference

```
#include <memory>
#include "SFML/Graphics.hpp"
```

Include dependency graph for Map/MapElement.h:



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



Classes

- class RTB::MapElement

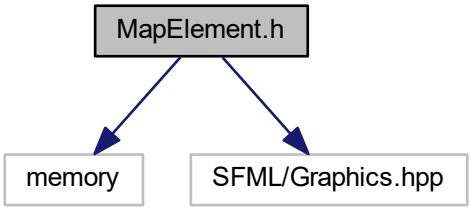
Namespaces

- RTB

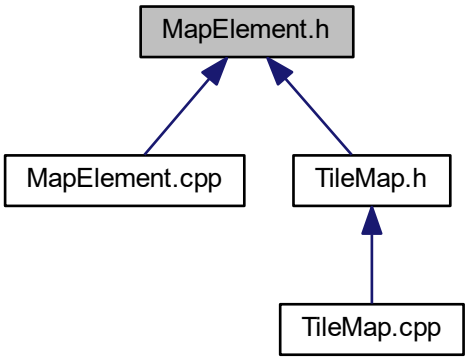
7.35 MapElement.h File Reference

```
#include <memory>
#include "SFML/Graphics.hpp"
```

Include dependency graph for Release/Map/MapElement.h:



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



Classes

- class RTB::MapElement

Namespaces

- RTB

7.36 mapFlora.txt File Reference

7.37 mapFlora.txt File Reference

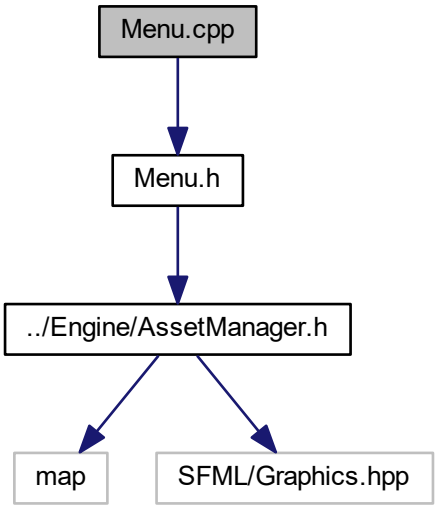
7.38 mapObjects.txt File Reference

7.39 mapObjects.txt File Reference

7.40 Menu.cpp File Reference

```
#include "Menu.h"
```

Include dependency graph for Menu.cpp:



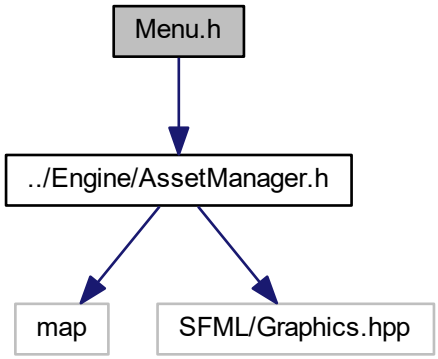
Namespaces

- RTBGUI

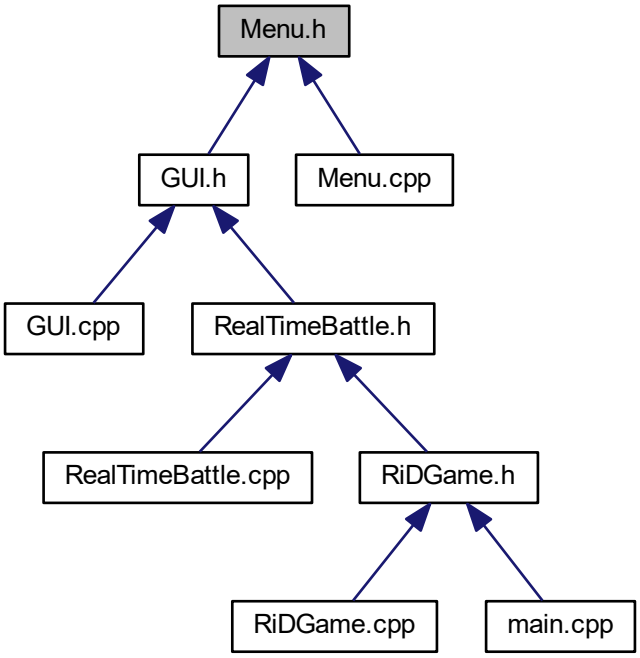
7.41 Menu.h File Reference

```
#include "../Engine/AssetManager.h"
```

Include dependency graph for Menu.h:



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



Classes

- class RTBGUI::Menu

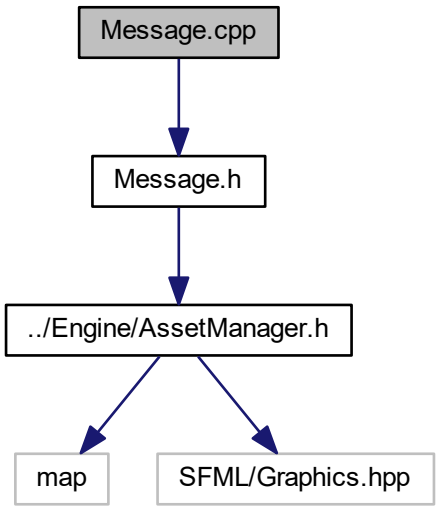
Namespaces

- RTBGUI

7.42 Message.cpp File Reference

```
#include "Message.h"
```

Include dependency graph for Message.cpp:



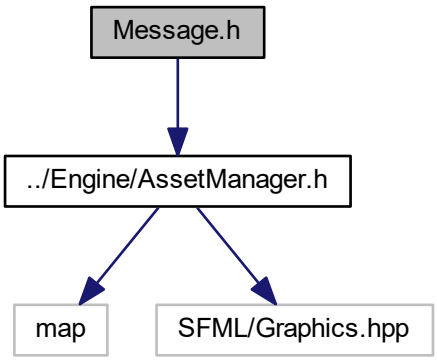
Namespaces

- RTBGUI

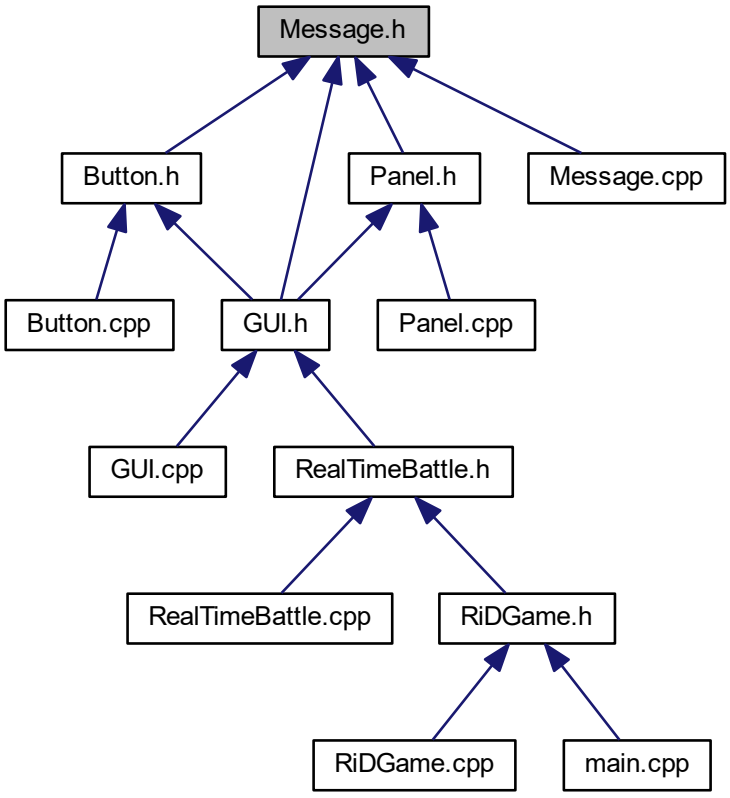
7.43 Message.h File Reference

```
#include "../Engine/AssetManager.h"
```

Include dependency graph for Message.h:



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



Classes

- class RTBGUI::Message

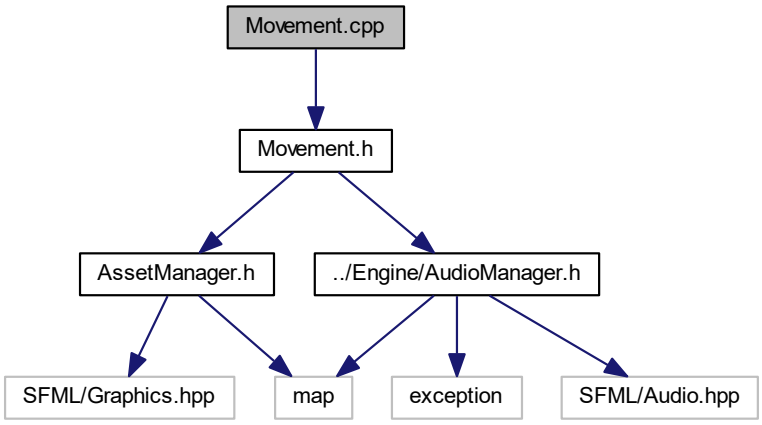
Namespaces

- RTBGUI

7.44 Movement.cpp File Reference

```
#include "Movement.h"
```

Include dependency graph for Movement.cpp:



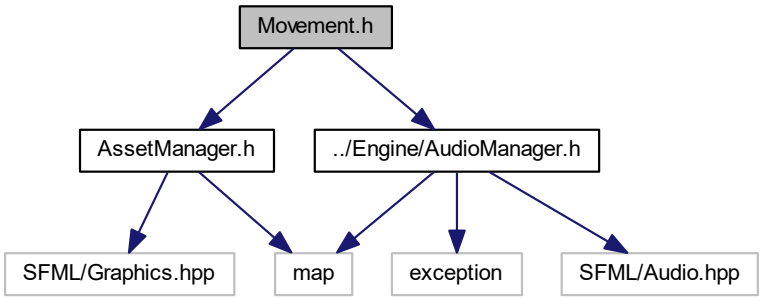
Namespaces

- RiD

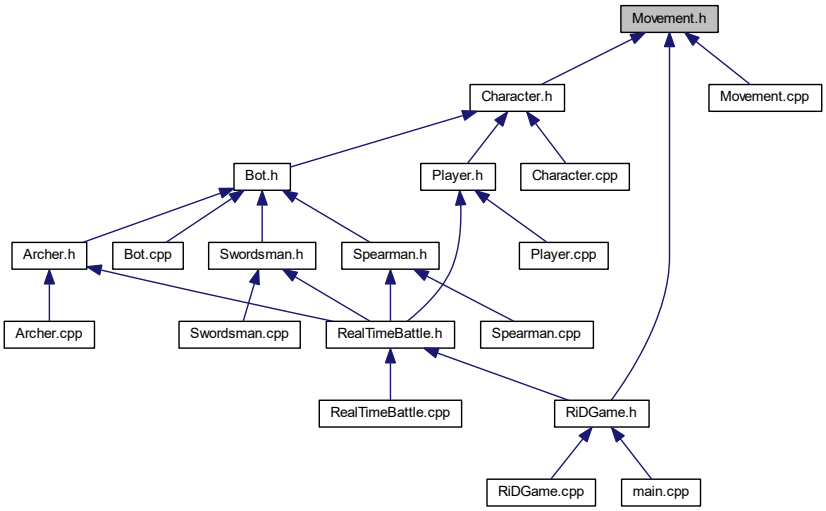
7.45 Movement.h File Reference

```
#include "AssetManager.h"
#include "../Engine/AudioManager.h"
```

Include dependency graph for Movement.h:



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



Classes

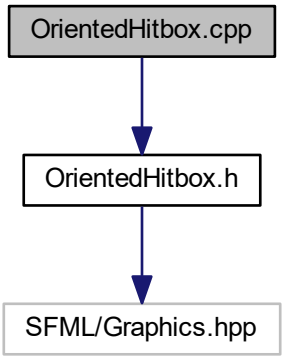
- class RiD::Movement
- struct RiD::Movement::animationDuration

Namespaces

- RiD

7.46 OrientedHitbox.cpp File Reference

```
#include "OrientedHitbox.h"
```
Include dependency graph for OrientedHitbox.cpp:

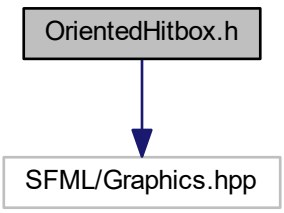


Namespaces

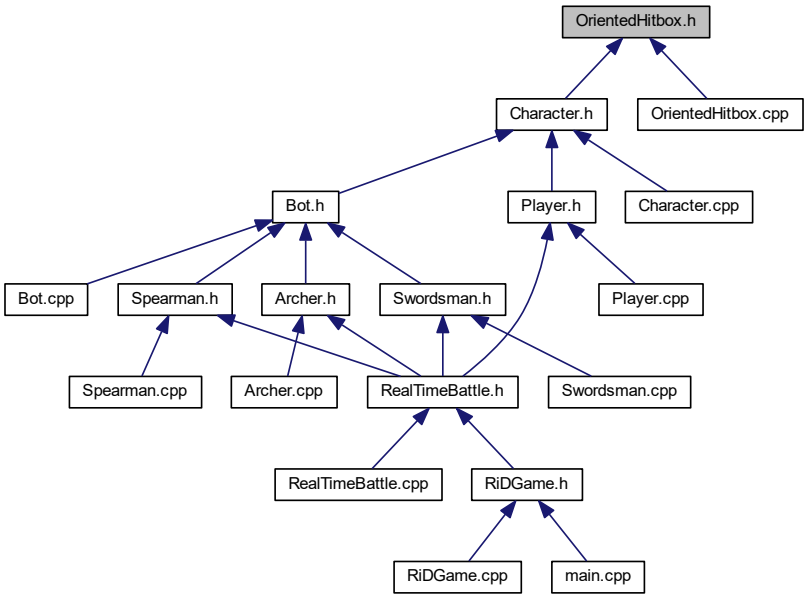
- RTB

7.47 OrientedHitbox.h File Reference

```
#include "SFML/Graphics.hpp"
```
Include dependency graph for OrientedHitbox.h:



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



Classes

- class RTB::OrientedHitbox

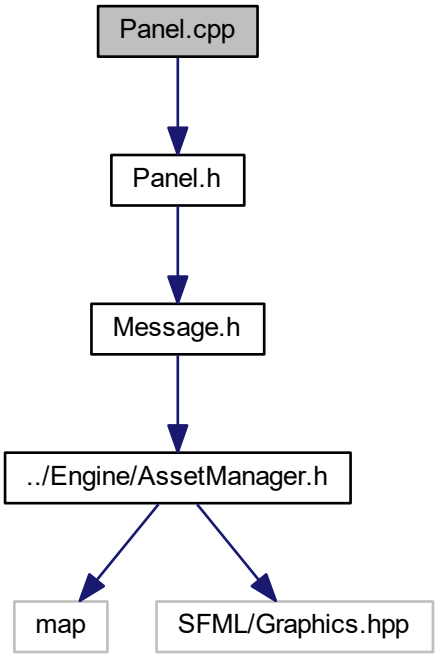
Namespaces

- RTB

7.48 Panel.cpp File Reference

```
#include "Panel.h"
```

Include dependency graph for Panel.cpp:



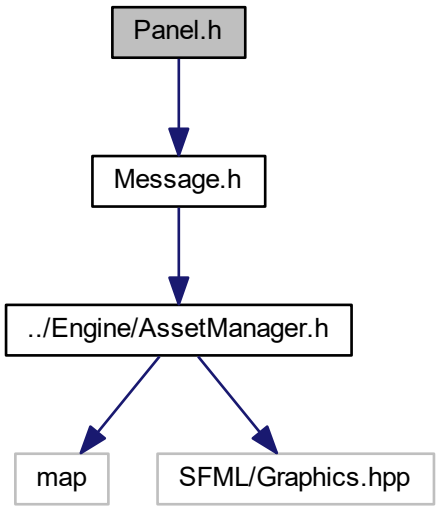
Namespaces

- RTBGUI

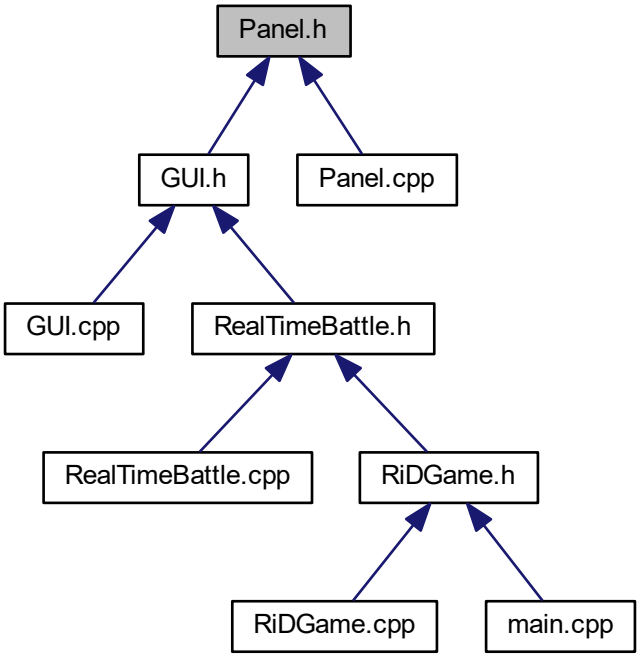
7.49 Panel.h File Reference

```
#include "Message.h"
```

Include dependency graph for Panel.h:



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



Classes

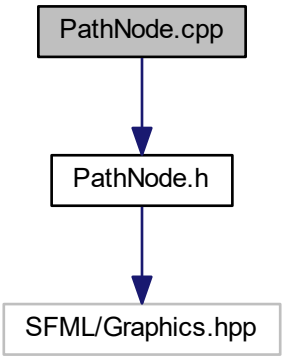
- class RTBGUI::Panel

Namespaces

- RTBGUI

7.50 PathNode.cpp File Reference

```
#include "PathNode.h"
```
Include dependency graph for PathNode.cpp:

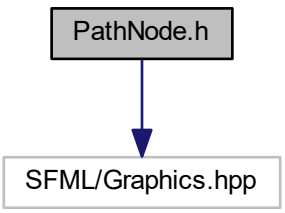


Namespaces

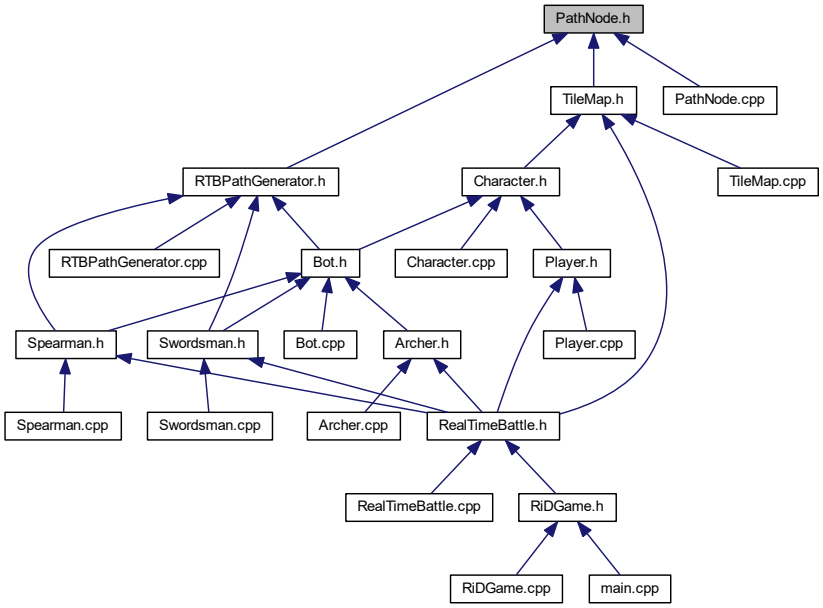
- AI

7.51 PathNode.h File Reference

```
#include "SFML/Graphics.hpp"
```
Include dependency graph for PathNode.h:



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



Classes

- class AI::PathNode

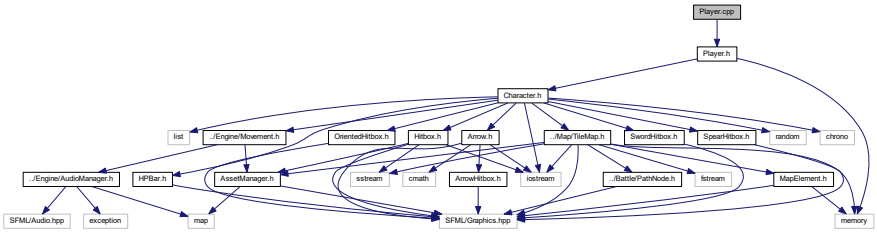
Namespaces

- AI

7.52 Player.cpp File Reference

```
#include "Player.h"
```

Include dependency graph for Player.cpp:



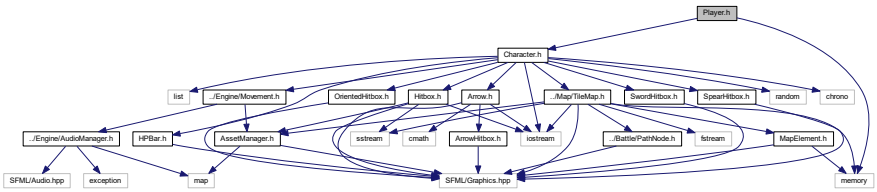
Namespaces

- RTB

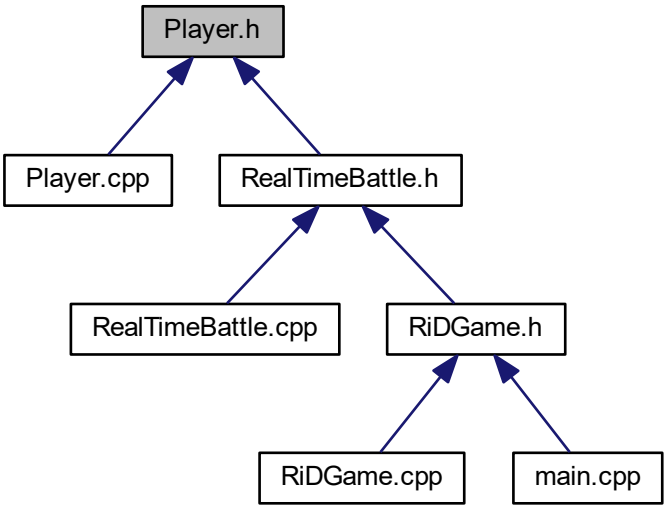
7.53 Player.h File Reference

```
#include "Character.h"
#include <memory>
```

Include dependency graph for Player.h:



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



Classes

- class RTB::Player

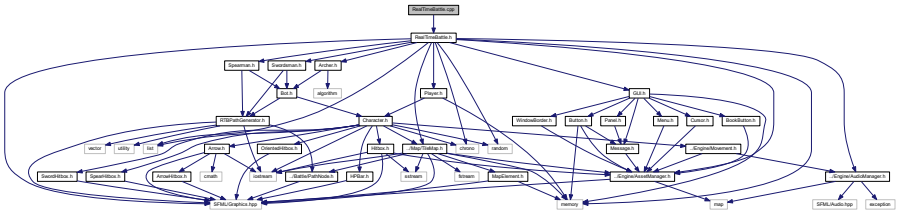
Namespaces

- RTB

7.54 RealTimeBattle.cpp File Reference

```
#include "RealTimeBattle.h"
```

Include dependency graph for RealTimeBattle.cpp:



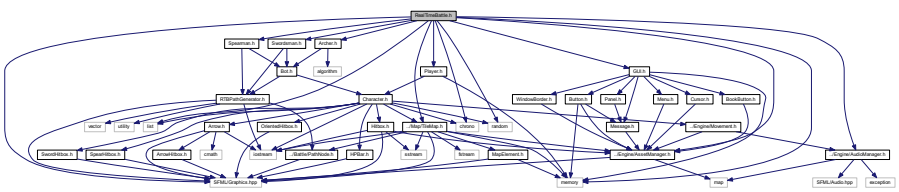
Namespaces

- RTB

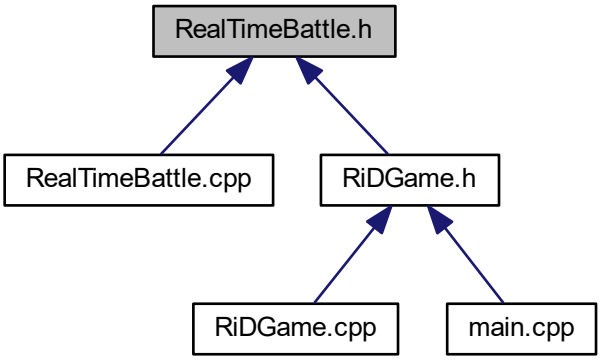
7.55 RealTimeBattle.h File Reference

```
#include <list>
#include <memory>
#include <chrono>
#include <random>
#include "SFML/Graphics.hpp"
#include "../Engine/AssetManager.h"
#include "../Map/TileMap.h"
#include "../Engine/AudioManager.h"
#include "Player.h"
#include "Swordsman.h"
#include "Archer.h"
#include "Spearman.h"
#include "GUI.h"
```

Include dependency graph for RealTimeBattle.h:



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



Classes

- class RTB::RealTimeBattle

Namespaces

- RTB

Macros

- #define ZOOM_UP 1.05
- #define ZOOM_DOWN 0.95

7.55.1 Macro Definition Documentation

7.55.1.1 ZOOM_DOWN

```
#define ZOOM_DOWN 0.95
```

7.55.1.2 ZOOM_UP

```
#define ZOOM_UP 1.05
```

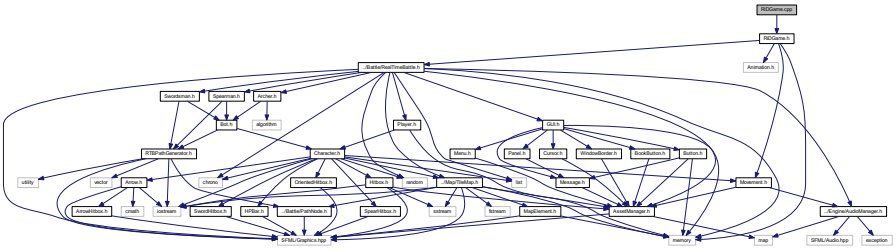
7.56 RiD_RPGproject.vcxproj.FileListAbsolute.txt File Reference

7.57 RiD_RPGproject.vcxproj.FileListAbsolute.txt File Reference

7.58 RiDGame.cpp File Reference

```
#include "RiDGame.h"
```

Include dependency graph for RiDGame.cpp:



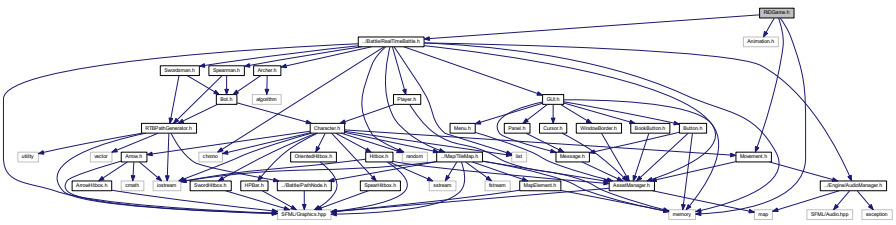
Namespaces

- RiD

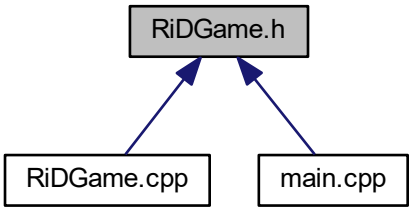
7.59 RiDGame.h File Reference

```
#include <memory>
#include "Animation.h"
#include "Movement.h"
#include "../Battle/RealTimeBattle.h"
```

Include dependency graph for RiDGame.h:



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



Classes

- struct RiD::gameDat
- class RiD::RiDGame

Namespaces

- RiD

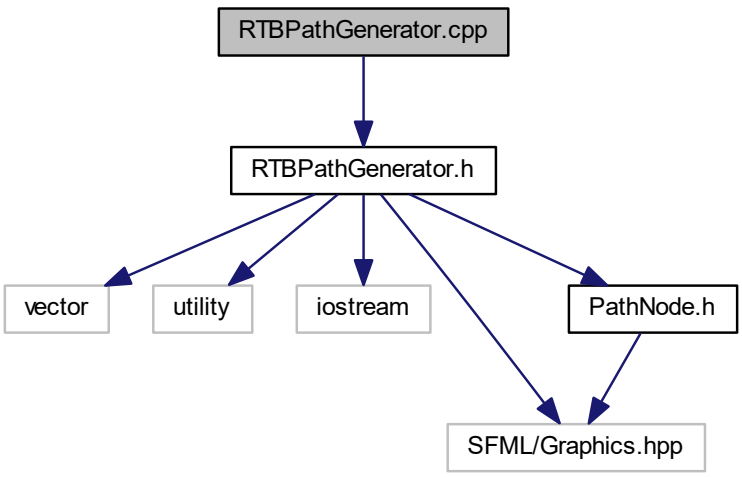
Typedefs

- typedef std::shared_ptr< gameDat > RiD::gameDatReference

7.60 RTBPathGenerator.cpp File Reference

```
#include "RTBPathGenerator.h"
```

Include dependency graph for RTBPathGenerator.cpp:



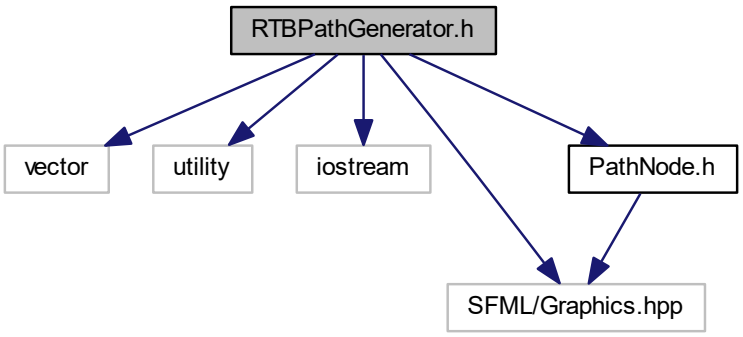
Namespaces

- AI

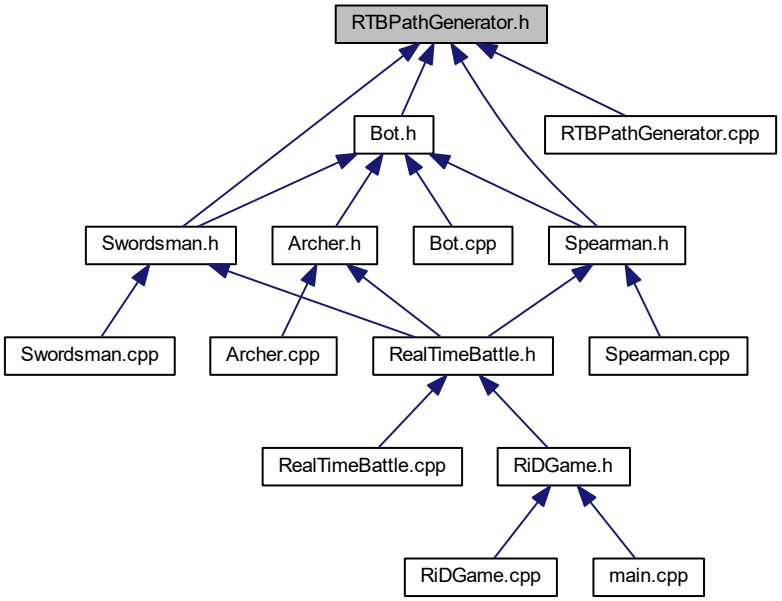
7.61 RTBPathGenerator.h File Reference

```
#include <vector>
#include <utility>
#include <iostream>
#include "SFML/Graphics.hpp"
#include "PathNode.h"
```

Include dependency graph for RTBPathGenerator.h:



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



Classes

- class AI::RTBPathGenerator

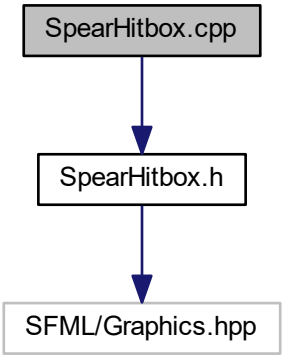
Namespaces

- AI

7.62 SpearHitbox.cpp File Reference

```
#include "SpearHitbox.h"
```

Include dependency graph for SpearHitbox.cpp:



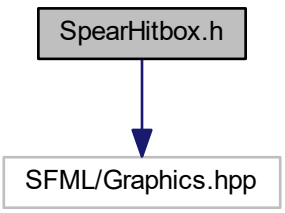
Namespaces

- RTB

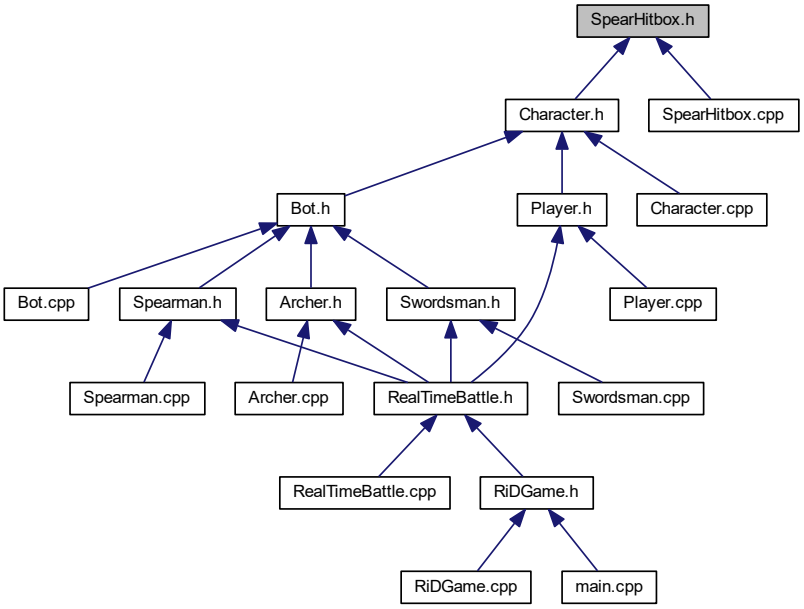
7.63 SpearHitbox.h File Reference

```
#include "SFML/Graphics.hpp"
```

Include dependency graph for SpearHitbox.h:



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



Classes

- class RTB::SpearHitbox

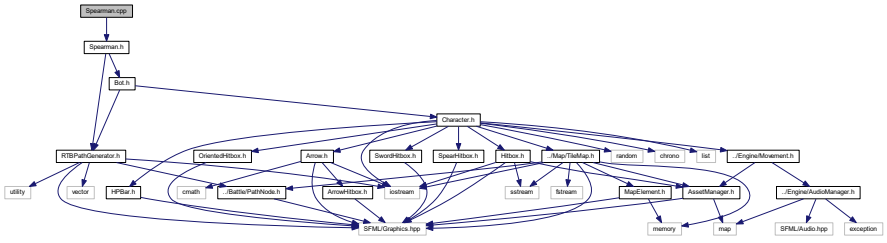
Namespaces

- RTB

7.64 Spearman.cpp File Reference

```
#include "Spearman.h"
```

Include dependency graph for Spearman.cpp:



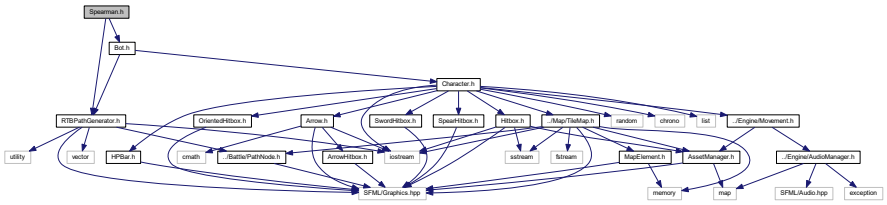
Namespaces

- RTB

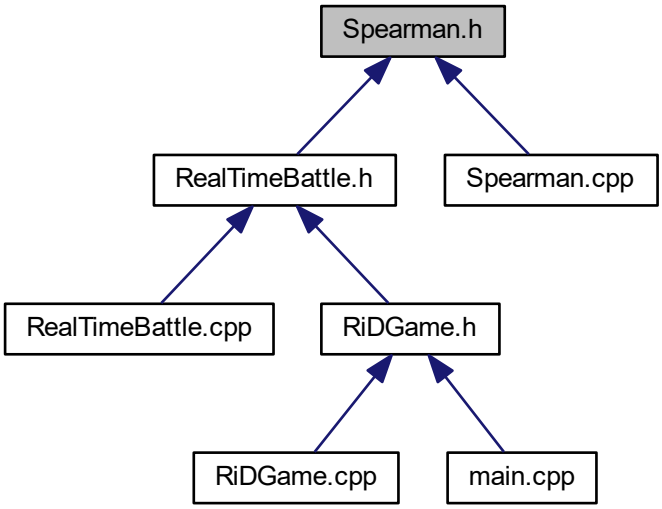
7.65 Spearman.h File Reference

```
#include "Bot.h"
#include "RTBPathGenerator.h"
```

Include dependency graph for Spearman.h:



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



Classes

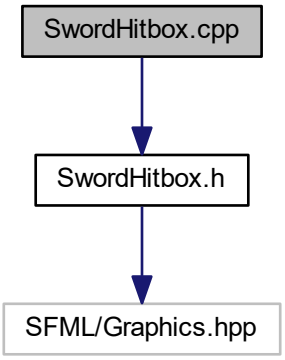
- class RTB::Spearman

Namespaces

- RTB

7.66 SwordHitbox.cpp File Reference

```
#include "SwordHitbox.h"
```
Include dependency graph for SwordHitbox.cpp:

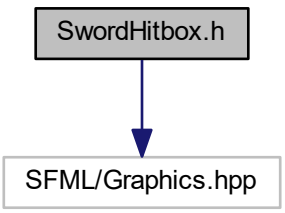


Namespaces

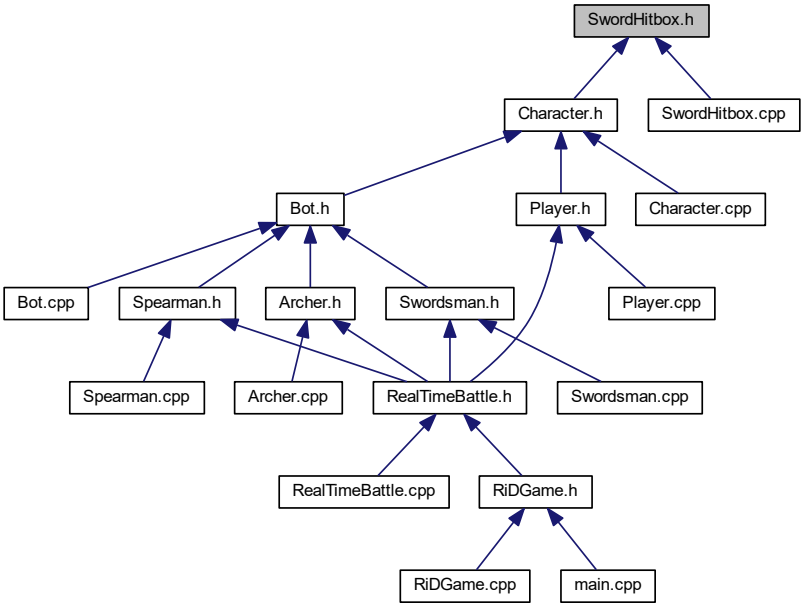
- RTB

7.67 SwordHitbox.h File Reference

```
#include "SFML/Graphics.hpp"
```
Include dependency graph for SwordHitbox.h:



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



Classes

- class RTB::SwordHitbox

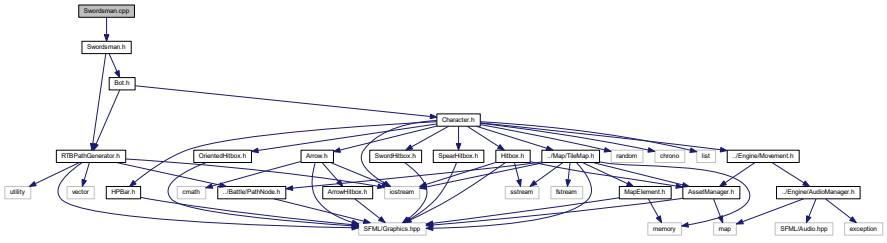
Namespaces

- RTB

7.68 Swordsman.cpp File Reference

```
#include "Swordsman.h"
```

Include dependency graph for Swordsman.cpp:



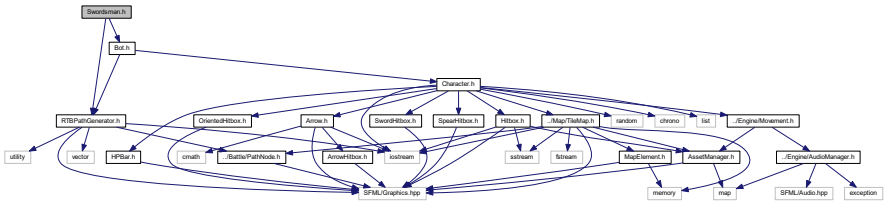
Namespaces

- RTB

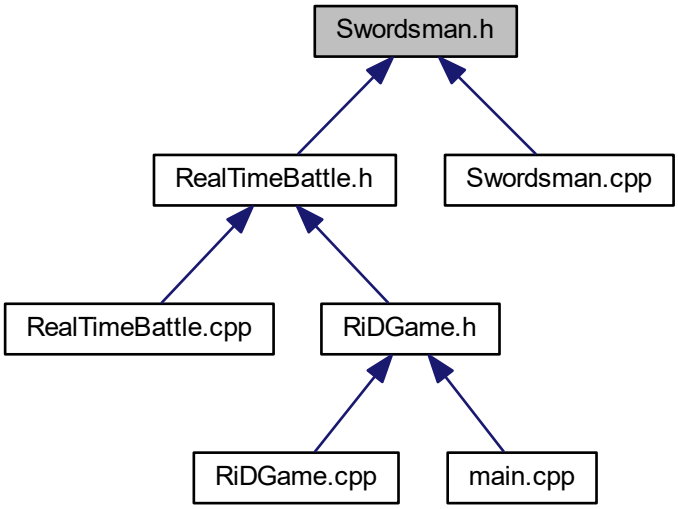
7.69 Swordsman.h File Reference

```
#include "Bot.h"
#include "RTBPathGenerator.h"
```

Include dependency graph for Swordsman.h:



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



Classes

- class RTB::Swordsman

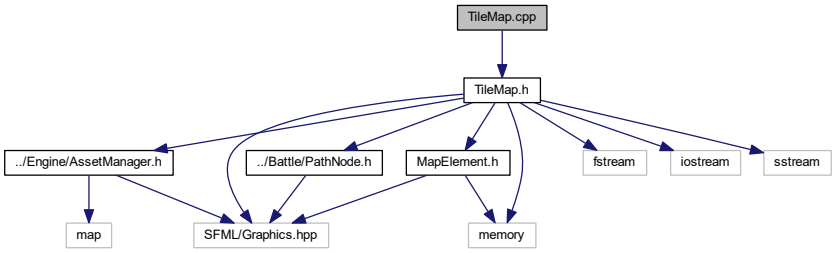
Namespaces

- RTB

7.70 TileMap.cpp File Reference

```
#include "TileMap.h"
```

Include dependency graph for Map/TileMap.cpp:



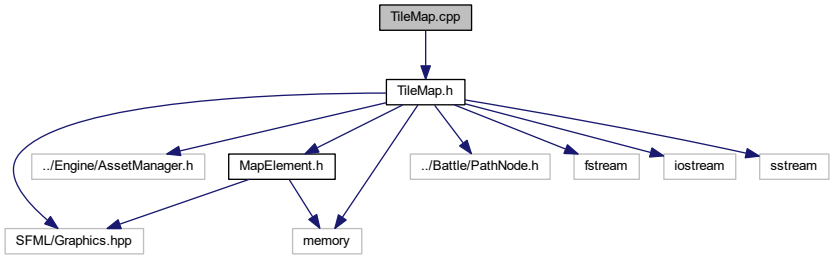
Namespaces

- RTB

7.71 TileMap.cpp File Reference

```
#include "TileMap.h"
```

Include dependency graph for Release/Map/TileMap.cpp:



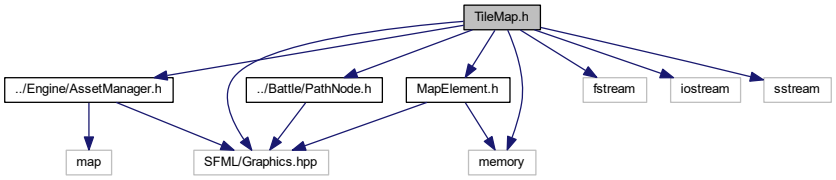
Namespaces

- RTB

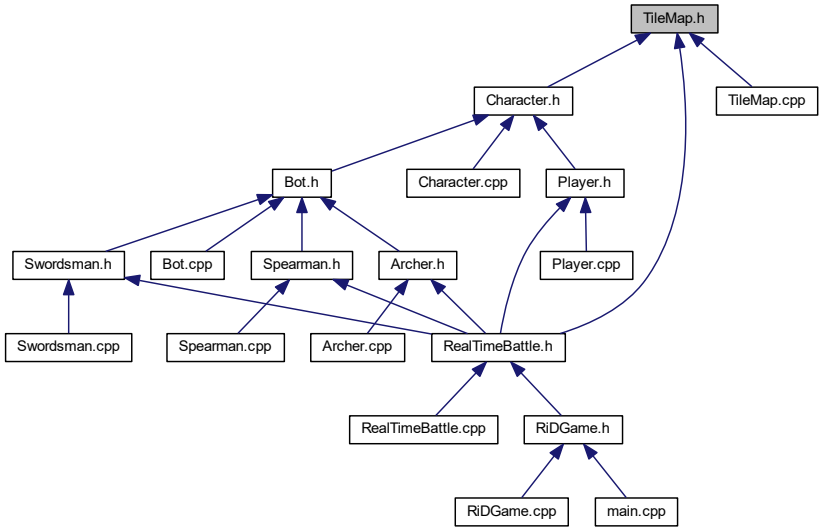
7.72 TileMap.h File Reference

```
#include "SFML/Graphics.hpp"
#include "../Engine/AssetManager.h"
#include "MapElement.h"
#include "../Battle/PathNode.h"
#include <fstream>
#include <iostream>
#include <sstream>
#include <memory>
```

Include dependency graph for Map/TileMap.h:



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



Classes

- class RTB::TileMap

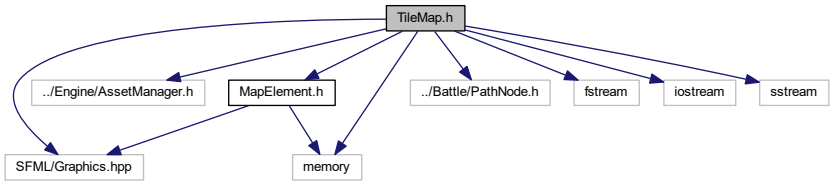
Namespaces

- RTB

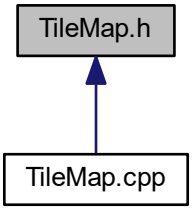
7.73 TileMap.h File Reference

```
#include "SFML/Graphics.hpp"
#include "../Engine/AssetManager.h"
#include "MapElement.h"
#include "../Battle/PathNode.h"
#include <fstream>
#include <iostream>
#include <sstream>
#include <memory>
```

Include dependency graph for Release/Map/TileMap.h:



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



Classes

- class RTB::TileMap

Namespaces

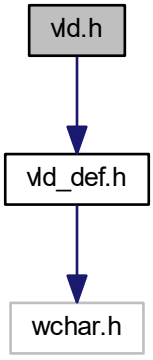
- RTB

7.74 tiles_positions.txt File Reference

7.75 vld.h File Reference

```
#include "vld_def.h"
```

Include dependency graph for vld.h:



Macros

- #define VLDEnable()
- #define VLDDisable()
- #define VLDRestore()
- #define VLDGlobalDisable()
- #define VLDGlobalEnable()
- #define VLDReportLeaks() (0)
- #define VLDReportThreadLeaks() (0)
- #define VLDGetLeaksCount() (0)
- #define VLDGetThreadLeaksCount() (0)
- #define VLDMarkAllLeaksAsReported()
- #define VLDMarkThreadLeaksAsReported(a)
- #define VLDRefreshModules()
- #define VLDEnableModule(a)
- #define VLDDisableModule(b)
- #define VLDGetOptions() (0)
- #define VLDGetReportFilename(a)
- #define VLDSetOptions(a, b, c)
- #define VLDSetReportHook(a, b)
- #define VLDSetModulesList(a)
- #define VLDGetModulesList(a, b) (FALSE)
- #define VLDSetReportOptions(a, b)
- #define VLDResolveCallstacks() (0)

Typedefs

- typedef int VLD_BOOL
- typedef unsigned int VLD_UINT
- typedef size_t VLD_SIZET
- typedef void * VLD_HMODULE

7.75.1 Macro Definition Documentation

7.75.1.1 VLDDisable

```
#define VLDDisable( )
```

7.75.1.2 VLDDisableModule

```
#define VLDDisableModule(
              b )
```

7.75.1.3 VLDEnable

```
#define VLDEnable( )
```

7.75.1.4 VLDEnableModule

```
#define VLDEnableModule(
              a )
```

7.75.1.5 VLDGetLeaksCount

```
#define VLDGetLeaksCount( ) (0)
```

7.75.1.6 VLDGetModulesList

```
#define VLDGetModulesList(
              a,
              b ) (FALSE)
```

7.75.1.7 VLDGetOptions

```
#define VLDGetOptions( ) (0)
```

7.75.1.8 VLDGetReportFilename

```
#define VLDGetReportFilename(
              a )
```

7.75.1.9 VLDGetThreadLeaksCount

```
#define VLDGetThreadLeaksCount( ) (0)
```

7.75.1.10 VLDGlobalDisable

```
#define VLDGlobalDisable( )
```

7.75.1.11 VLDGlobalEnable

```
#define VLDGlobalEnable( )
```

7.75.1.12 VLDMarkAllLeaksAsReported

```
#define VLDMarkAllLeaksAsReported( )
```

7.75.1.13 VLDMarkThreadLeaksAsReported

```
#define VLDMarkThreadLeaksAsReported(
              a )
```

7.75.1.14 VLDRefreshModules

```
#define VLDRefreshModules( )
```

7.75.1.15 VLDReportLeaks

```
#define VLDReportLeaks( ) (0)
```

7.75.1.16 VLDReportThreadLeaks

```
#define VLDReportThreadLeaks( ) (0)
```

7.75.1.17 VLDResolveCallstacks

```
#define VLDResolveCallstacks( ) (0)
```

7.75.1.18 VLDRestore

```
#define VLDRestore( )
```

7.75.1.19 VLDSetModulesList

```
#define VLDSetModulesList(
              a )
```

7.75.1.20 VLDSetOptions

```
#define VLDSetOptions(
              a,
              b,
              c )
```

7.75.1.21 VLDSetReportHook

```
#define VLDSetReportHook(
              a,
              b )
```

7.75.1.22 VLDSetReportOptions

```
#define VLDSetReportOptions(
              a,
              b )
```

7.75.2 Typedef Documentation

7.75.2.1 VLD_BOOL

```
typedef int VLD_BOOL
```

7.75.2.2 VLD_HMODULE

```
typedef void* VLD_HMODULE
```

7.75.2.3 VLD_SIZET

```
typedef size_t VLD_SIZET
```

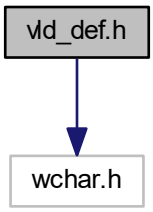
7.75.2.4 VLD_UINT

```
typedef unsigned int VLD_UINT
```

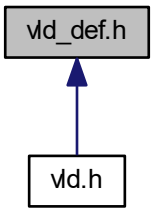
7.76 vld_def.h File Reference

```
#include <wchar.h>
```

Include dependency graph for vld_def.h:



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



Macros

- #define VLD_OPT_AGGREGATE_DUPLICATES 0x0001
- #define VLD_OPT_MODULE_LIST_INCLUDE 0x0002
- #define VLD_OPT_REPORT_TO_DEBUGGER 0x0004
- #define VLD_OPT_REPORT_TO_FILE 0x0008
- #define VLD_OPT_SAFE_STACK_WALK 0x0010
- #define VLD_OPT_SELF_TEST 0x0020
- #define VLD_OPT_SLOW_DEBUGGER_DUMP 0x0040
- #define VLD_OPT_START_DISABLED 0x0080
- #define VLD_OPT_TRACE_INTERNAL_FRAMES 0x0100

- #define VLD_OPT_UNICODE_REPORT 0x0200
- #define VLD_OPT_VLDOFF 0x0400
- #define VLD_OPT_REPORT_TO_STDOUT 0x0800
- #define VLD_OPT_SKIP_HEAPFREE_LEAKS 0x1000
- #define VLD_OPT_VALIDATE_HEAPFREE 0x2000
- #define VLD_OPT_SKIP_CRTSTARTUP_LEAKS 0x4000
- #define VLD_RPTHOOK_INSTALL 0
- #define VLD_RPTHOOK_REMOVE 1

Typedefs

- typedef int(__cdecl * VLD_REPORT_HOOK) (int reportType, wchar_t *message, int *returnValue)

7.76.1 Macro Definition Documentation

7.76.1.1 VLD_OPT_AGGREGATE_DUPLICATES

```
#define VLD_OPT_AGGREGATE_DUPLICATES 0x0001
```

7.76.1.2 VLD_OPT_MODULE_LIST_INCLUDE

```
#define VLD_OPT_MODULE_LIST_INCLUDE 0x0002
```

7.76.1.3 VLD_OPT_REPORT_TO_DEBUGGER

```
#define VLD_OPT_REPORT_TO_DEBUGGER 0x0004
```

7.76.1.4 VLD_OPT_REPORT_TO_FILE

```
#define VLD_OPT_REPORT_TO_FILE 0x0008
```

7.76.1.5 VLD_OPT_REPORT_TO_STDOUT

```
#define VLD_OPT_REPORT_TO_STDOUT 0x0800
```

7.76.1.6 VLD_OPT_SAFE_STACK_WALK

```
#define VLD_OPT_SAFE_STACK_WALK 0x0010
```

7.76.1.7 VLD_OPT_SELF_TEST

```
#define VLD_OPT_SELF_TEST 0x0020
```

7.76.1.8 VLD_OPT_SKIP_CRTSTARTUP_LEAKS

```
#define VLD_OPT_SKIP_CRTSTARTUP_LEAKS 0x4000
```

7.76.1.9 VLD_OPT_SKIP_HEAPFREE_LEAKS

```
#define VLD_OPT_SKIP_HEAPFREE_LEAKS 0x1000
```

7.76.1.10 VLD_OPT_SLOW_DEBUGGER_DUMP

```
#define VLD_OPT_SLOW_DEBUGGER_DUMP 0x0040
```

7.76.1.11 VLD_OPT_START_DISABLED

```
#define VLD_OPT_START_DISABLED 0x0080
```

7.76.1.12 VLD_OPT_TRACE_INTERNAL_FRAMES

```
#define VLD_OPT_TRACE_INTERNAL_FRAMES 0x0100
```

7.76.1.13 VLD_OPT_UNICODE_REPORT

```
#define VLD_OPT_UNICODE_REPORT 0x0200
```

7.76.1.14 VLD_OPT_VALIDATE_HEAPFREE

```
#define VLD_OPT_VALIDATE_HEAPFREE 0x2000
```

7.76.1.15 VLD_OPT_VLDOFF

```
#define VLD_OPT_VLDOFF 0x0400
```

7.76.1.16 VLD_RPTHOOK_INSTALL

```
#define VLD_RPTHOOK_INSTALL 0
```

7.76.1.17 VLD_RPTHOOK_REMOVE

```
#define VLD_RPTHOOK_REMOVE 1
```

7.76.2 Typedef Documentation

7.76.2.1 VLD_REPORT_HOOK

```
typedef int(__cdecl * VLD_REPORT_HOOK) (int reportType, wchar_t *message, int *returnValue)
```

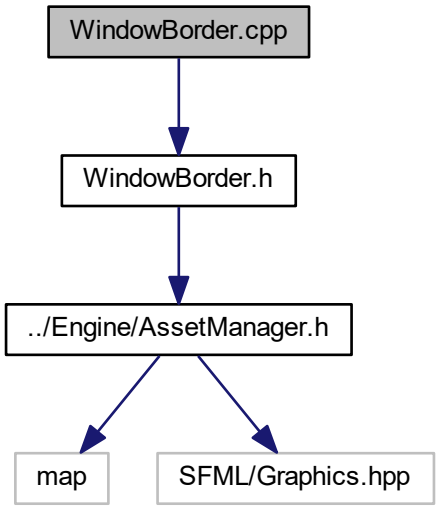
7.77 walkable.txt File Reference

7.78 walkable.txt File Reference

7.79 WindowBorder.cpp File Reference

```
#include "WindowBorder.h"
```

Include dependency graph for WindowBorder.cpp:



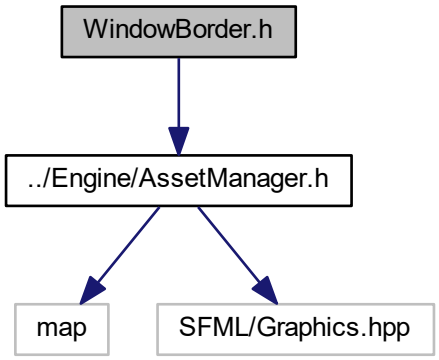
Namespaces

- RTBGUI

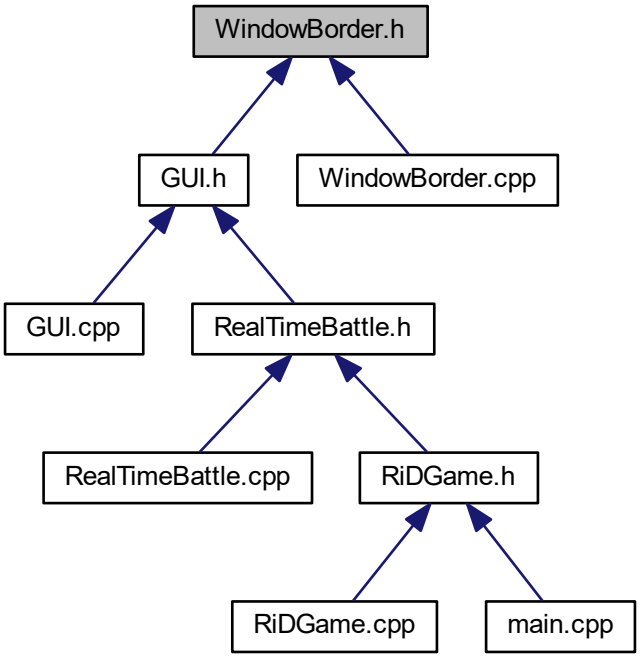
7.80 WindowBorder.h File Reference

```
#include "../Engine/AssetManager.h"
```

Include dependency graph for WindowBorder.h:



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



Classes

- class RTBGUI::WindowBorder

Namespaces

- RTBGUI