

BattleRoyale

Generated by Doxygen 1.8.11

Contents

1	Class Index	1
1.1	Class List	1
2	File Index	3
2.1	File List	3
3	Class Documentation	5
3.1	Action Class Reference	5
3.1.1	Detailed Description	5
3.1.2	Member Function Documentation	5
3.1.2.1	command(const Movement &m)	5
3.2	Bike Struct Reference	5
3.2.1	Detailed Description	6
3.3	Board Class Reference	6
3.3.1	Detailed Description	7
3.3.2	Constructor & Destructor Documentation	7
3.3.2.1	Board()	7
3.3.3	Member Function Documentation	7
3.3.3.1	bike(int id)	7
3.3.3.2	bike_ok(int id) const	7
3.3.3.3	bikes(int player) const	7
3.3.3.4	bonus_round() const	7
3.3.3.5	bonus_vertices() const	7
3.3.3.6	ghost_duration() const	7

3.3.3.7	<code>is_neighbour(int vertex_a, int vertex_b) const</code>	8
3.3.3.8	<code>map() const</code>	8
3.3.3.9	<code>nb_bikes() const</code>	8
3.3.3.10	<code>nb_players() const</code>	8
3.3.3.11	<code>nb_rounds() const</code>	8
3.3.3.12	<code>nb_vertices() const</code>	8
3.3.3.13	<code>player_ok(int player) const</code>	8
3.3.3.14	<code>round() const</code>	8
3.3.3.15	<code>score(int player) const</code>	8
3.3.3.16	<code>score_bonus() const</code>	8
3.3.3.17	<code>secgame() const</code>	9
3.3.3.18	<code>status(int player)</code>	9
3.3.3.19	<code>turbo_duration() const</code>	9
3.3.3.20	<code>version()</code>	9
3.3.3.21	<code>vertex(int v) const</code>	9
3.3.3.22	<code>vertex_ok(int id) const</code>	9
3.4	Movement Struct Reference	9
3.4.1	Detailed Description	10
3.4.2	Constructor & Destructor Documentation	10
3.4.2.1	<code>Movement(int unit_id_)</code>	10
3.4.2.2	<code>Movement(int unit_id_, int next_vertex_, bool use_bonus_=false)</code>	10
3.5	Vertex Struct Reference	10
3.5.1	Detailed Description	10
3.5.2	Member Data Documentation	10
3.5.2.1	<code>wall</code>	10
4	File Documentation	11
4.1	Action.hh File Reference	11
4.1.1	Detailed Description	11
4.2	Board.hh File Reference	11
4.2.1	Detailed Description	12
4.2.2	Enumeration Type Documentation	12
4.2.2.1	<code>Bonus</code>	12
	Index	13

Chapter 1

Class Index

1.1 Class List

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

Action	5
Bike	5
Board	6
Movement	9
Vertex	10

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Action.hh	11
Board.hh	11

Chapter 3

Class Documentation

3.1 Action Class Reference

```
#include <Action.hh>
```

Public Member Functions

- bool [command](#) (const [Movement](#) &m)

3.1.1 Detailed Description

Class that stores the actions requested by a player in a round.

3.1.2 Member Function Documentation

3.1.2.1 bool [Action::command](#) (const [Movement](#) & *m*) [[inline](#)]

Adds a movement to the action list. Fails and returns false if a movement is already present for this unit.

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

- [Action.hh](#)

3.2 Bike Struct Reference

```
#include <Board.hh>
```

Public Attributes

- int [id](#)
The unique id for this bike in the board.
- int [player](#)
The id of the player that owns this bike.
- [Bonus](#) [bonus](#)
The bonus item this bike collected, if any.
- int [vertex](#)
The current position of this bike in the graph.
- int [turbo_duration](#)
Number of rounds this unit can move in turbo mode.
- int [ghost_duration](#)
Number of rounds this bike can move in ghost mode.
- bool [alive](#)
True until the bike crashes.

3.2.1 Detailed Description

Defines a bike on the board and its properties

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

- [Board.hh](#)

3.3 Board Class Reference

```
#include <Board.hh>
```

Public Member Functions

- [Board](#) ()
- string [map](#) () const
- int [nb_players](#) () const
- int [nb_bikes](#) () const
- int [nb_rounds](#) () const
- int [bonus_round](#) () const
- int [turbo_duration](#) () const
- int [ghost_duration](#) () const
- int [score_bonus](#) () const
- int [nb_vertices](#) () const
- vector< int > [bonus_vertices](#) () const
- bool [secgame](#) () const
- int [round](#) () const
- bool [player_ok](#) (int player) const
- bool [bike_ok](#) (int id) const
- bool [vertex_ok](#) (int id) const
- bool [is_neighbour](#) (int vertex_a, int vertex_b) const
- int [score](#) (int player) const
- double [status](#) (int player)
- const [Vertex](#) & [vertex](#) (int v) const
- const [Bike](#) & [bike](#) (int id)
- vector< int > [bikes](#) (int player) const

Static Public Member Functions

- static string [version](#) ()

3.3.1 Detailed Description

Represents and manages the game board.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 `Board::Board ()` [inline]

Empty constructor.

3.3.3 Member Function Documentation

3.3.3.1 `const Bike& Board::bike (int id)` [inline]

Returns the bike with identifier *id*.

3.3.3.2 `bool Board::bike_ok (int id) const` [inline]

Return whether *id* is a valid bike identifier.

3.3.3.3 `vector<int> Board::bikes (int player) const` [inline]

Returns the bikes of a certain player

3.3.3.4 `int Board::bonus_round () const` [inline]

Returns the round number when the bonus item will appear in the vertices given by [bonus_vertices\(\)](#)

3.3.3.5 `vector<int> Board::bonus_vertices () const` [inline]

Returns the vertices where bonus items will appear

3.3.3.6 `int Board::ghost_duration () const` [inline]

Returns the duration of the ghost mode in rounds, when activated by a bike

3.3.3.7 `bool Board::is_neighbour (int vertex_a, int vertex_b) const` `[inline]`

Returns whether `vertex_b` can be reached from `vertex_a`. Does not take obstacles into account.

3.3.3.8 `string Board::map () const` `[inline]`

Returns the name of the map ("plane", "icosahedron"...)

3.3.3.9 `int Board::nb_bikes () const` `[inline]`

Returns the number of bikes per player

3.3.3.10 `int Board::nb_players () const` `[inline]`

Returns the number of players in the game

3.3.3.11 `int Board::nb_rounds () const` `[inline]`

Returns the number of rounds for the game

3.3.3.12 `int Board::nb_vertices () const` `[inline]`

Returns the size of the board graph

3.3.3.13 `bool Board::player_ok (int player) const` `[inline]`

Return whether `player` is a valid player identifier.

3.3.3.14 `int Board::round () const` `[inline]`

Returns the current round.

3.3.3.15 `int Board::score (int player) const` `[inline]`

Returns the current score of a player.

3.3.3.16 `int Board::score_bonus () const` `[inline]`

Returns the extra points given by collecting the "Points" bonus item

3.3.3.17 `bool Board::secgame () const [inline]`

Returns true when the game is running in a secure environment (i.e. the server)

3.3.3.18 `double Board::status (int player) [inline]`

Returns the percentage of cpu time used in the last round, in the range [0.0 - 1.0] or a value lesser than 0 if this player is dead. Note that this is only accessible if [secgame\(\)](#) is true

3.3.3.19 `int Board::turbo_duration () const [inline]`

Returns the duration of the turbo mode in rounds, when activated by a bike

3.3.3.20 `static string Board::version () [inline], [static]`

Return a string with the game name and version

3.3.3.21 `const Vertex& Board::vertex (int v) const [inline]`

Returns the vertex *v*.

3.3.3.22 `bool Board::vertex_ok (int id) const [inline]`

Return whether *id* is a valid vertex identifier.

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

- [Board.hh](#)

3.4 Movement Struct Reference

```
#include <Action.hh>
```

Public Member Functions

- [Movement](#) (int unit_id_)
- [Movement](#) (int unit_id_, int next_vertex_, bool use_bonus_=false)

Public Attributes

- int **unit_id**
- int **next_vertex**
- bool **use_bonus**

3.4.1 Detailed Description

A movement defines to which vertex will a particular unit move

3.4.2 Constructor & Destructor Documentation

3.4.2.1 Movement::Movement (int *unit_id_*) [inline]

Constructor, given only a unit id

3.4.2.2 Movement::Movement (int *unit_id_*, int *next_vertex_*, bool *use_bonus_* = false) [inline]

Constructor, given a unit id, the vertex to move to and whether to use or not the item in the bike's inventory

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

- [Action.hh](#)

3.5 Vertex Struct Reference

```
#include <Board.hh>
```

Public Attributes

- int [id](#)
The unique id for this vertex in the board.
- int [bike](#)
The id of the bike in this vertex if present, -1 otherwise.
- int [wall](#)
- [Bonus](#) [bonus](#)
The bonus item that this vertex contains.
- vector< int > [neighbours](#)
The ids of the vertices that can be reached from this one.

3.5.1 Detailed Description

Describes a vertex in the board graph, and its contents

3.5.2 Member Data Documentation

3.5.2.1 int Vertex::wall

Will be -1 if there is no wall or the id of the bike that created the wall otherwise. Pre-generated obstacles, if any, will be represented with a number greater than the number of bikes.

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

- [Board.hh](#)

Chapter 4

File Documentation

4.1 Action.hh File Reference

```
#include "Utils.hh"
```

Classes

- struct [Movement](#)
- class [Action](#)

4.1.1 Detailed Description

Contains the class [Action](#) and the struct [Movement](#) that it uses.

4.2 Board.hh File Reference

```
#include "Utils.hh"
```

Classes

- struct [Vertex](#)
- struct [Bike](#)
- class [Board](#)

Enumerations

- enum [Bonus](#) {
 None = 0,
 Turbo,
 Ghost,
 Points,
 BonusEnumSize }

Functions

- char **bonus2char** ([Bonus](#) b)
- [Bonus](#) **char2bonus** (char c)

4.2.1 Detailed Description

Contains the [Board](#) class itself and structs to represent all the elements that can be on the board.

4.2.2 Enumeration Type Documentation

4.2.2.1 enum **Bonus**

Defines the collectable bonus that can be on a vertex

Index

Action, [5](#)
 command, [5](#)
Action.hh, [11](#)

Bike, [5](#)
bike
 Board, [7](#)
bike_ok
 Board, [7](#)
bikes
 Board, [7](#)
Board, [6](#)
 bike, [7](#)
 bike_ok, [7](#)
 bikes, [7](#)
 Board, [7](#)
 bonus_round, [7](#)
 bonus_vertices, [7](#)
 ghost_duration, [7](#)
 is_neighbour, [7](#)
 map, [8](#)
 nb_bikes, [8](#)
 nb_players, [8](#)
 nb_rounds, [8](#)
 nb_vertices, [8](#)
 player_ok, [8](#)
 round, [8](#)
 score, [8](#)
 score_bonus, [8](#)
 secgame, [8](#)
 status, [9](#)
 turbo_duration, [9](#)
 version, [9](#)
 vertex, [9](#)
 vertex_ok, [9](#)
Board.hh, [11](#)
 Bonus, [12](#)
Bonus
 Board.hh, [12](#)
bonus_round
 Board, [7](#)
bonus_vertices
 Board, [7](#)

command
 Action, [5](#)

ghost_duration
 Board, [7](#)

is_neighbour
 Board, [7](#)

map
 Board, [8](#)
Movement, [9](#)
 Movement, [10](#)

nb_bikes
 Board, [8](#)
nb_players
 Board, [8](#)
nb_rounds
 Board, [8](#)
nb_vertices
 Board, [8](#)

player_ok
 Board, [8](#)

round
 Board, [8](#)

score
 Board, [8](#)
score_bonus
 Board, [8](#)
secgame
 Board, [8](#)
status
 Board, [9](#)

turbo_duration
 Board, [9](#)

version
 Board, [9](#)
Vertex, [10](#)
 wall, [10](#)
vertex
 Board, [9](#)
vertex_ok
 Board, [9](#)

wall
 Vertex, [10](#)