

cpp_indie_studio 2018

Generated by Doxygen 1.8.11

Contents

1	Hierarchical Index	1
1.1	Class Hierarchy	1
2	Class Index	3
2.1	Class List	3
3	File Index	5
3.1	File List	5
4	Class Documentation	7
4.1	Bomberman Class Reference	7
4.1.1	Constructor & Destructor Documentation	8
4.1.1.1	Bomberman(bool firstPlayer, bool ia, int maxBomb)	8
4.1.1.2	~Bomberman()	8
4.1.2	Member Function Documentation	8
4.1.2.1	addBonus(std::shared_ptr< Map > Map)	8
4.1.2.2	deleteBomb(std::shared_ptr< Map > Map)	9
4.1.2.3	dropBomb()	9
4.1.2.4	findSpawn(std::shared_ptr< Map > Map, std::shared_ptr< Collision > Collision)	9
4.1.2.5	getItem()	9
4.1.2.6	getPlayerDead() const	10
4.1.2.7	getX() const	10
4.1.2.8	getZ() const	10
4.1.2.9	initBomberman(irr::scene::ISceneManager *smgr, irr::video::IVideoDriver *<=> Driver, std::shared_ptr< Graphical > Graphical)	10

4.1.2.10	initKeyMove()	10
4.1.2.11	invExplodeBomb(int f, int s, std::shared_ptr< Map > Map)	11
4.1.2.12	isRunning() const	11
4.1.2.13	moveBomberman(std::shared_ptr< Map > Map, std::shared_ptr< Collision > Collision)	11
4.1.2.14	realizeDisplacement()	12
4.1.2.15	setEnemy(std::shared_ptr< Bomberman > enemy)	12
4.1.2.16	setEnemy(std::shared_ptr< IA > ia)	12
4.1.2.17	setItem(enum Item)	12
4.1.2.18	setKeyMove(int key, int x, int z)	12
4.1.2.19	setMulti()	12
4.1.2.20	timeToExplode(std::shared_ptr< Map > Map)	12
4.2	BombStandard Class Reference	13
4.2.1	Constructor & Destructor Documentation	13
4.2.1.1	BombStandard(ISceneManager *smgr, IVideoDriver *driver, int range)	13
4.2.1.2	~BombStandard()	14
4.2.2	Member Function Documentation	14
4.2.2.1	addBonus(Cube::TypeBox type)	14
4.2.2.2	explodeBomb(int x, int y, std::shared_ptr< Map > Map, std::vector< std::pair< int, int >> playerPos)	14
4.2.2.3	getId()	14
4.2.2.4	getItemName()	15
4.2.2.5	getNbrBombBonus() const	15
4.2.2.6	getPlayerDead() const	15
4.2.2.7	getRangeBonus() const	15
4.2.2.8	getSpeedBonus() const	15
4.2.2.9	isUsed(float x, float z)	15
4.3	Button Class Reference	16
4.3.1	Constructor & Destructor Documentation	16
4.3.1.1	Button(const int pos[2], const int len[2], const std::string &name, int value)	16
4.3.1.2	~Button()	17

4.3.2	Member Function Documentation	17
4.3.2.1	getLenX() const	17
4.3.2.2	getLenY() const	17
4.3.2.3	getName() const	17
4.3.2.4	getPosX() const	18
4.3.2.5	getPosY() const	18
4.3.2.6	isClick(int clickX, int clickY) const	18
4.4	Collision Class Reference	18
4.4.1	Constructor & Destructor Documentation	18
4.4.1.1	Collision(int mapWidth, int mapHeight)	18
4.4.1.2	~Collision()	19
4.4.2	Member Function Documentation	19
4.4.2.1	checkCollision(int x, int y, std::shared_ptr< Map > Map)	19
4.4.2.2	getFarTen(int toFind)	19
4.5	Credits Class Reference	20
4.5.1	Constructor & Destructor Documentation	20
4.5.1.1	Credits()	20
4.5.1.2	~Credits()	20
4.5.2	Member Function Documentation	20
4.5.2.1	displayAll(std::shared_ptr< Graphical > Graphical)	20
4.5.2.2	displayLbyL(std::shared_ptr< Graphical > Graphical)	21
4.5.2.3	displayPic()	21
4.5.2.4	displayText(std::shared_ptr< Graphical > Graphical)	21
4.6	Cube Class Reference	21
4.6.1	Member Enumeration Documentation	22
4.6.1.1	TypeBox	22
4.6.2	Constructor & Destructor Documentation	22
4.6.2.1	Cube(int width, int height, irr::scene::ISceneManager *manager, irr::video::IVideoDriver *driver, bool destructable, TypeBox type)	22
4.6.2.2	~Cube()	23
4.6.3	Member Function Documentation	23

4.6.3.1	getCubeType() const	23
4.6.3.2	getMesh() const	23
4.6.3.3	getposx() const	23
4.6.3.4	getposy() const	24
4.6.3.5	isDestruct() const	24
4.6.3.6	isDestructable() const	24
4.6.3.7	setDestruct()	24
4.7	Floor Class Reference	24
4.7.1	Constructor & Destructor Documentation	25
4.7.1.1	Floor(int height, int width, irr::scene::ISceneManager *manager, irr::video::I← VideoDriver *driver)	25
4.7.1.2	~Floor()	26
4.7.2	Member Function Documentation	26
4.7.2.1	getHeight() const	26
4.7.2.2	getWidth() const	26
4.8	GetEvent Class Reference	26
4.8.1	Constructor & Destructor Documentation	27
4.8.1.1	GetEvent()	27
4.8.1.2	~GetEvent()	27
4.8.2	Member Function Documentation	27
4.8.2.1	getMousePos() const	27
4.8.2.2	isKeyPressed(int keyNum) const	27
4.8.2.3	isLeftClick() const	28
4.8.2.4	isRightClick() const	28
4.8.2.5	OnEvent(const irr::SEvent &event)	28
4.9	Graphical Class Reference	28
4.9.1	Constructor & Destructor Documentation	29
4.9.1.1	Graphical()	29
4.9.1.2	~Graphical()	29
4.9.2	Member Function Documentation	29
4.9.2.1	addCamera(const irr::core::vector3df &position, const irr::core::vector3df &lookat)	29

4.9.2.2	drawButton(const std::string &toDisplay, int pos[4], const std::string &backPic)	30
4.9.2.3	drawGUI()	30
4.9.2.4	drawMessageBox(const std::string &toDisplay, int pos[2], int len[2])	30
4.9.2.5	drawScene()	30
4.9.2.6	drawText(const std::string &toDisplay, int pos[4], bool adapToText)	30
4.9.2.7	drawTexture(const std::string &picPath, int origPos[2], int picLen[4])	30
4.9.2.8	end()	30
4.9.2.9	getDevice()	31
4.9.2.10	getEvent()	31
4.9.2.11	getMousePos()	31
4.9.2.12	getScene()	31
4.9.2.13	getVideo()	31
4.9.2.14	initCamera()	31
4.9.2.15	initGraphical()	31
4.9.2.16	isLeftClick()	31
4.9.2.17	isRightClick()	31
4.9.2.18	setCamera(float x, float y, float z)	31
4.9.2.19	start()	32
4.10	IA Class Reference	32
4.10.1	Member Enumeration Documentation	33
4.10.1.1	IaDir	33
4.10.2	Constructor & Destructor Documentation	33
4.10.2.1	IA(irr::scene::ISceneManager *smgr, irr::video::IVideoDriver *Driver, std::shared_ptr< Graphical > Graphical)	33
4.10.2.2	~IA()	34
4.10.3	Member Function Documentation	34
4.10.3.1	BetterPosAround(std::shared_ptr< Map > Map, std::shared_ptr< Collision > Collision)	34
4.10.3.2	calcBombDirst(float _x, float _z)	34
4.10.3.3	checkIfContinue(std::shared_ptr< Map > Map, std::shared_ptr< Collision > Collision)	35

4.10.3.4	<code>checkIfNotLast(std::shared_ptr< Map > Map, std::shared_ptr< Collision > Collision)</code>	35
4.10.3.5	<code>deleteBomb(std::shared_ptr< Map > Map)</code>	35
4.10.3.6	<code>dropBomb(std::shared_ptr< Map > Map, std::shared_ptr< Collision > Collision)</code>	35
4.10.3.7	<code>findPosNoSmash(std::shared_ptr< Map > Map, std::shared_ptr< Collision > Collision, std::pair< int, int > pos)</code>	36
4.10.3.8	<code>findSpawn(std::shared_ptr< Map > Map, std::shared_ptr< Collision > Collision)</code>	36
4.10.3.9	<code>getIA newPos(std::shared_ptr< Map > Map, std::shared_ptr< Collision > Collision)</code>	36
4.10.3.10	<code>getItem()</code>	37
4.10.3.11	<code>getPlayerDead() const</code>	37
4.10.3.12	<code>getX() const</code>	37
4.10.3.13	<code>getZ() const</code>	37
4.10.3.14	<code>invExplodeBomb(int f, int s, std::shared_ptr< Map > Map)</code>	37
4.10.3.15	<code>moveIA(std::shared_ptr< Map > Map, std::shared_ptr< Collision > Collision)</code>	38
4.10.3.16	<code>nearPlayer()</code>	38
4.10.3.17	<code>setPPos(float a, float b)</code>	38
4.10.3.18	<code>timeToExplode(std::shared_ptr< Map > Map)</code>	38
4.10.3.19	<code>wallAround(std::shared_ptr< Map > Map, std::shared_ptr< Collision > Collision)</code>	38
4.11	Item Class Reference	39
4.11.1	Constructor & Destructor Documentation	39
4.11.1.1	<code>~Item()=default</code>	39
4.11.2	Member Function Documentation	39
4.11.2.1	<code>explodeBomb(int x, int y, std::shared_ptr< Map > Map, std::vector< std::pair< int, int >> playerPos)=0</code>	39
4.11.2.2	<code>getId()=0</code>	39
4.11.2.3	<code>getItemName()=0</code>	40
4.11.2.4	<code>getPlayerDead() const =0</code>	40
4.11.2.5	<code>isUsed(float x, float z)=0</code>	40
4.12	Intro Class Reference	40
4.12.1	Constructor & Destructor Documentation	40
4.12.1.1	<code>Intro()</code>	40
4.12.1.2	<code>~Intro()</code>	41

4.12.2	Member Function Documentation	41
4.12.2.1	displayAll(std::shared_ptr< Graphical > Graphical)	41
4.12.2.2	displayLbyL(std::shared_ptr< Graphical > Graphical)	41
4.12.2.3	displayPic()	41
4.12.2.4	displayText(std::shared_ptr< Graphical > Graphical)	41
4.13	Map Class Reference	42
4.13.1	Constructor & Destructor Documentation	43
4.13.1.1	Map(irr::scene::ISceneManager *manager, irr::video::IVideoDriver *driver, std::map< std::string, int > vParam)	43
4.13.1.2	~Map()	43
4.13.2	Member Function Documentation	43
4.13.2.1	addBonus()	43
4.13.2.2	createFloor()	43
4.13.2.3	generateBorder()	43
4.13.2.4	generateHeightMap()	44
4.13.2.5	generateMap()	44
4.13.2.6	generateTypeBox()	44
4.13.2.7	generateTypeBoxRt(float rdm, float caseWSpeed, float caseWSRange, float caseWBoost)	44
4.13.2.8	generateWidthMap()	44
4.13.2.9	getHeight() const	44
4.13.2.10	getLastBonus() const	45
4.13.2.11	getNbrBomb() const	45
4.13.2.12	getRange() const	45
4.13.2.13	getSpeed() const	45
4.13.2.14	getWidth() const	45
4.13.2.15	isCubeNotEmpty(int x, int y, bool isToDestroy)	45
4.13.2.16	isInit() const	45
4.13.2.17	isValidPos(std::pair< int, int > pos)	46
4.13.2.18	setCubeDeleted(int x, int y)	46
4.14	Menu Class Reference	46

4.14.1	Constructor & Destructor Documentation	47
4.14.1.1	Menu()	47
4.14.1.2	~Menu()	47
4.14.2	Member Function Documentation	47
4.14.2.1	drawMenu(std::shared_ptr< Graphical > Graphical)	47
4.14.2.2	getButton() const	48
4.14.2.3	getClickButton(const int &clickX, const int &clickY, bool isClick)	48
4.14.2.4	initButton()	48
4.14.2.5	initEquivAction()	48
4.14.2.6	isRun() const	49
4.14.2.7	setEquivAction(const std::string &nameAction, actualState myAction)	49
4.14.2.8	setGame(bool game)	49
4.14.2.9	setMultiplayer(bool multi)	49
4.15	Param Class Reference	50
4.15.1	Constructor & Destructor Documentation	50
4.15.1.1	Param()	50
4.15.1.2	~Param()	51
4.15.2	Member Function Documentation	51
4.15.2.1	drawNameButton(std::shared_ptr< Graphical > Graphical)	51
4.15.2.2	drawParam(std::shared_ptr< Graphical > Graphical)	51
4.15.2.3	drawReturnButton(std::shared_ptr< Graphical > Graphical)	51
4.15.2.4	drawSideButton(std::shared_ptr< Graphical > Graphical)	52
4.15.2.5	getButton() const	52
4.15.2.6	getClickButton(const int &clickX, const int &clickY, bool isClick)	52
4.15.2.7	getFormattedStringButtonName(Button b1)	52
4.15.2.8	getParam() const	53
4.15.2.9	initButton()	53
4.15.2.10	initButtonLimits()	53
4.15.2.11	initEquivAction()	53
4.15.2.12	initIntoButton()	53

4.15.2.13 isRun() const	53
4.15.2.14 newStateButton(Button b1)	53
4.15.2.15 setButtonLimits(const std::string &nameButton, int lBottom, int lTop)	54
4.15.2.16 setEquivButton(const std::string &nameButton, const std::string &equivButton)	54
4.15.2.17 setIntoButton(const std::string &nameButton, int value)	54
4.16 Song Class Reference	54
4.16.1 Constructor & Destructor Documentation	55
4.16.1.1 Song()	55
4.16.1.2 ~Song()	55
4.16.2 Member Function Documentation	55
4.16.2.1 playLongSong(const char *song)	55
4.16.2.2 playShortSong(const char *song)	55
4.16.2.3 stopLongSong()	56
4.17 StudioCore Class Reference	56
4.17.1 Constructor & Destructor Documentation	57
4.17.1.1 StudioCore()	57
4.17.1.2 ~StudioCore()	57
4.17.2 Member Function Documentation	57
4.17.2.1 computeMapParam()	57
4.17.2.2 initBomberman()	58
4.17.2.3 initCore()	58
4.17.2.4 initMultiplayerBomberman()	58
4.17.2.5 initSelectState()	58
4.17.2.6 runCore()	58
4.17.2.7 runCredits()	58
4.17.2.8 runDeath()	59
4.17.2.9 runDifferentCases()	59
4.17.2.10 runGame()	59
4.17.2.11 runIntro()	59
4.17.2.12 runMenu()	59
4.17.2.13 runMultiplayerGame()	59
4.17.2.14 runNewGame()	60
4.17.2.15 runParam()	60

5 File Documentation	61
5.1 Collision/Collision.cpp File Reference	61
5.2 Core/StudioCore.cpp File Reference	61
5.3 Credits/Credits.cpp File Reference	61
5.4 Game/Bomberman.cpp File Reference	61
5.5 Graphical/GetEvent.cpp File Reference	61
5.6 Graphical/Graphical.cpp File Reference	62
5.7 IA/IA.cpp File Reference	62
5.8 include/Bomberman.hpp File Reference	62
5.8.1 Macro Definition Documentation	62
5.8.1.1 MAXBOMB	62
5.9 include/Button.hpp File Reference	62
5.10 include/Collision.hpp File Reference	63
5.11 include/Credits.hpp File Reference	63
5.11.1 Macro Definition Documentation	63
5.11.1.1 PERSO_PATH	63
5.12 include/Cube.hpp File Reference	63
5.13 include/Floor.hpp File Reference	64
5.14 include/GetEvent.hpp File Reference	64
5.15 include/Graphical.hpp File Reference	64
5.15.1 Macro Definition Documentation	65
5.15.1.1 FONT_PATH	65
5.15.1.2 FTB	65
5.15.1.3 S_TO_WS	65
5.16 include/IA.hpp File Reference	65
5.17 include/indieStudio.hpp File Reference	65
5.17.1 Macro Definition Documentation	66
5.17.1.1 WINDOW_HEIGHT	66
5.17.1.2 WINDOW_WIDTH	66
5.17.2 Enumeration Type Documentation	66

5.17.2.1	actualState	66
5.18	include/Intro.hpp File Reference	66
5.18.1	Macro Definition Documentation	67
5.18.1.1	PERSO_PATH	67
5.19	include/Item.hpp File Reference	67
5.19.1	Enumeration Type Documentation	67
5.19.1.1	Item	67
5.20	include/Map.hpp File Reference	67
5.20.1	Macro Definition Documentation	68
5.20.1.1	NBRBOMB	68
5.20.1.2	S_TO_WS	68
5.21	include/Menu.hpp File Reference	68
5.21.1	Macro Definition Documentation	68
5.21.1.1	FONT_MENU	68
5.22	include/Param.hpp File Reference	68
5.23	include/Song.hpp File Reference	68
5.24	include/StudioCore.hpp File Reference	69
5.24.1	Macro Definition Documentation	69
5.24.1.1	MULTI_P	69
5.25	Introduction/Intro.cpp File Reference	69
5.26	Item/BombStandard.cpp File Reference	69
5.27	Item/include/BombStandard.hpp File Reference	69
5.28	main.cpp File Reference	70
5.28.1	Function Documentation	70
5.28.1.1	main(int ac, char **av)	70
5.29	Map/Cube.cpp File Reference	70
5.30	Map/Floor.cpp File Reference	70
5.31	Map/Map.cpp File Reference	70
5.32	Menu/Button.cpp File Reference	70
5.33	Menu/Menu.cpp File Reference	70
5.34	Menu/Param.cpp File Reference	71
5.35	Song/Song.cpp File Reference	71

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Bomberman	7
Button	16
Collision	18
Credits	20
Cube	21
Floor	24
Graphical	28
IA	32
IEventReceiver	
GetEvent	26
Item	39
BombStandard	13
Intro	40
Map	42
Menu	46
Param	50
Song	54
StudioCore	56

Chapter 2

Class Index

2.1 Class List

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

Bomberman	7
BombStandard	13
Button	16
Collision	18
Credits	20
Cube	21
Floor	24
GetEvent	26
Graphical	28
IA	32
Item	39
Intro	40
Map	42
Menu	46
Param	50
Song	54
StudioCore	56

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

main.cpp	70
Collision/ Collision.cpp	61
Core/ StudioCore.cpp	61
Credits/ Credits.cpp	61
Game/ Bomberman.cpp	61
Graphical/ GetEvent.cpp	61
Graphical/ Graphical.cpp	62
IA/ IA.cpp	62
include/ Bomberman.hpp	62
include/ Button.hpp	62
include/ Collision.hpp	63
include/ Credits.hpp	63
include/ Cube.hpp	63
include/ Floor.hpp	64
include/ GetEvent.hpp	64
include/ Graphical.hpp	64
include/ IA.hpp	65
include/ indieStudio.hpp	65
include/ Intro.hpp	66
include/ Item.hpp	67
include/ Map.hpp	67
include/ Menu.hpp	68
include/ Param.hpp	68
include/ Song.hpp	68
include/ StudioCore.hpp	69
Introduction/ Intro.cpp	69
Item/ BombStandard.cpp	69
Item/include/ BombStandard.hpp	69
Map/ Cube.cpp	70
Map/ Floor.cpp	70
Map/ Map.cpp	70
Menu/ Button.cpp	70
Menu/ Menu.cpp	70
Menu/ Param.cpp	71
Song/ Song.cpp	71

Chapter 4

Class Documentation

4.1 Bomberman Class Reference

```
#include <Bomberman.hpp>
```

Public Member Functions

- [Bomberman](#) (bool firstPlayer, bool ia, int maxBomb)
construct the player with his parameters
- [~Bomberman](#) ()
destructor
- bool [isRunning](#) () const
check if bomberman is running
- std::shared_ptr< [Item](#) > [getItem](#) ()
get an item
- void [setItem](#) (enum [Item](#))
set an item
- int [initBomberman](#) (irr::scene::ISceneManager *smgr, irr::video::IVideoDriver *Driver, std::shared_ptr< [Graphical](#) > [Graphical](#))
Construction of the charcater bomberman.
- void [moveBomberman](#) (std::shared_ptr< [Map](#) > [Map](#), std::shared_ptr< [Collision](#) > [Collision](#))
realise moving and moving the camera
- void [timeToExplode](#) (std::shared_ptr< [Map](#) > [Map](#))
- void [deleteBomb](#) (std::shared_ptr< [Map](#) > [Map](#))
deleting bomb when she explodes
- void [invExplodeBomb](#) (int f, int s, std::shared_ptr< [Map](#) > [Map](#))
getting the positions and see who died
- void [findSpawn](#) (std::shared_ptr< [Map](#) > [Map](#), std::shared_ptr< [Collision](#) > [Collision](#))
searching the good spawn
- float [getZ](#) () const
get z
- float [getX](#) () const
get x
- void [setKeyMove](#) (int key, int x, int z)
set key moves

- void `initKeyMove` ()
initialisation of commands for player1 and player2
- void `realizeDeplacement` ()
realize the player deplacement with the keyboard inputs
- void `dropBomb` ()
dropping bomb by players
- void `setMulti` ()
set multiplayer
- void `setEnemy` (std::shared_ptr< `Bomberman` > enemy)
- void `setEnemy` (std::shared_ptr< `IA` > ia)
- std::string `getPlayerDead` () const
get player dead
- void `addBonus` (std::shared_ptr< `Map` > `Map`)
adding all the bonuses

4.1.1 Constructor & Destructor Documentation

4.1.1.1 `Bomberman::Bomberman (bool firstPlayer, bool ia, int maxBomb)`

construct the player with his parameters

Parameters

<i>ia</i>	is it ia or no
<i>firstPlayer</i>	is it the first player
<i>maxBomb</i>	nb max bomb to drop

Returns

nothing

4.1.1.2 `Bomberman::~~Bomberman ()`

destructor

Returns

nothing

4.1.2 Member Function Documentation

4.1.2.1 `void Bomberman::addBonus (std::shared_ptr< Map > Map)`

adding all the bonuses

Parameters

<i>Map</i>	represent the <i>Map</i>
------------	--------------------------

Returns

nothing

4.1.2.2 void Bomberman::deleteBomb (std::shared_ptr< *Map* > *Map*)

deleting bomb when she explodes

Parameters

<i>Map</i>	represent the <i>Map</i>
------------	--------------------------

Returns

nothing

4.1.2.3 void Bomberman::dropBomb ()

dropping bomb by players

Returns

nothing

4.1.2.4 void Bomberman::findSpawn (std::shared_ptr< *Map* > *Map*, std::shared_ptr< *Collision* > *Collision*)

searching the good spawn

Parameters

<i>Map</i>	represent the <i>Map</i>
<i>Collision</i>	handling collision

Returns

nothing

4.1.2.5 std::shared_ptr< *Item* > Bomberman::getItem ()

get an item

Returns

std::shared_ptr<Item> _inv[0]

4.1.2.6 std::string Bomberman::getPlayerDead () const

get player dead

4.1.2.7 float Bomberman::getX () const

get x

Returns

float _x

4.1.2.8 float Bomberman::getZ () const

get z

Returns

float _z

4.1.2.9 int Bomberman::initBomberman (irr::scene::ISceneManager * *smgr*, irr::video::IVideoDriver * *Driver*, std::shared_ptr<Graphical > *Graphical*)

Construction of the charcater bomberman.

Parameters

<i>smgr</i>	managing the scene
<i>Driver</i>	handling vidéo
<i>Graphical</i>	managing the graphics of character

Returns

int

4.1.2.10 void Bomberman::initKeyMove ()

initialisation of commands for player1 and player2

Returns

nothing

4.1.2.11 `void Bomberman::invExplodeBomb (int f, int s, std::shared_ptr< Map > Map)`

getting the positions and see who died

Parameters

<i>Map</i>	represent the <i>Map</i>
<i>f</i>	position first
<i>s</i>	position second

Returns

nothing

4.1.2.12 `bool Bomberman::isRunning () const`

check if bomberman is running

Returns

true or false

4.1.2.13 `void Bomberman::moveBomberman (std::shared_ptr< Map > Map, std::shared_ptr< Collision > Collision)`

realise moving and moving the camera

handling the duration of the explodes of bomb

Parameters

<i>Map</i>	represent the <i>Map</i>
<i>Collision</i>	handling the collision

Returns

nothing

Parameters

<i>Map</i>	represent the <i>Map</i>
------------	--------------------------

Returns

nothing

4.1.2.14 void Bomberman::realizeDeplacement ()

realize the player deplacement withe keyboard inputs

Returns

nothing

4.1.2.15 void Bomberman::setEnemy (std::shared_ptr< Bomberman > *enemy*)

4.1.2.16 void Bomberman::setEnemy (std::shared_ptr< IA > *ia*)

4.1.2.17 void Bomberman::setItem (enum *Item*)

set an item

Returns

nothing

4.1.2.18 void Bomberman::setKeyMove (int *key*, int *x*, int *y*)

set key moves

Parameters

<i>key</i>	key nbr
<i>x</i>	x index
<i>y</i>	y index

Returns

nothing

4.1.2.19 void Bomberman::setMulti ()

set multiplayer

Returns

nothing

4.1.2.20 void Bomberman::timeToExplode (std::shared_ptr< Map > *Map*)

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

- include/[Bomberman.hpp](#)
- Game/[Bomberman.cpp](#)

4.2 BombStandard Class Reference

```
#include <BombStandard.hpp>
```

Inherits [IItem](#).

Public Member Functions

- [BombStandard](#) (ISceneManager *smgr, IVideoDriver *driver, int range)
constructor (init standard bomb)
- [~BombStandard](#) ()
destructor
- IAnimatedMeshSceneNode * [isUsed](#) (float x, float z)
plant the bomb at (x, y) position.
- std::string [getItemName](#) ()
get item name
- int [getId](#) ()
- bool [explodeBomb](#) (int x, int y, std::shared_ptr< [Map](#) > [Map](#), std::vector< std::pair< int, int >> playerPos)
Bomb explosion gestion (with bonus)
- std::pair< int, int > [getPlayerDead](#) () const
get player dead
- int [getSpeedBonus](#) () const
get speed bonus
- int [getRangeBonus](#) () const
get range bonus
- int [getNbrBombBonus](#) () const
get number of bombs bonus
- void [addBonus](#) ([Cube::TypeBox](#) type)
add bonus

4.2.1 Constructor & Destructor Documentation

4.2.1.1 BombStandard::BombStandard (ISceneManager * smgr, IVideoDriver * driver, int range)

constructor (init standard bomb)

Parameters

<i>ISceneManager</i>	*smgr: scene manager
<i>IVideoDriver</i>	*driver: driver
<i>int</i>	range: range

Returns

nothing

4.2.1.2 BombStandard::~~BombStandard ()

destructor

Returns

nothing

4.2.2 Member Function Documentation

4.2.2.1 void BombStandard::addBonus (Cube::TypeBox type)

add bonus

Parameters

Cube::TypeBox	type: bonus type
-------------------------------	------------------

Returns

nothing

4.2.2.2 bool BombStandard::explodeBomb (int x, int y, std::shared_ptr< Map > Map, std::vector< std::pair< int, int >> playerPos) [virtual]

Bomb explosion gestion (with bonus)

Parameters

<i>int</i>	x: position on x
<i>int</i>	y: position on y
<i>std::shared_ptr<Map></i>	Map : map
<i>std::vector<std::pair<int,int>></i>	playerPos: vector of player positions

Returns

true or false

Implements [IItem](#).

4.2.2.3 int BombStandard::getId () [virtual]

Implements [IItem](#).

4.2.2.4 `std::string BombStandard::getItemName () [virtual]`

get item name

Returns

`std::string _itemName`

Implements [IItem](#).

4.2.2.5 `int BombStandard::getNbrBombBonus () const`

get number of bombs bonus

Returns

`int _nbrBombBonus`

4.2.2.6 `std::pair< int, int > BombStandard::getPlayerDead () const [virtual]`

get player dead

Returns

`std::pair<int, int> _playerDead`

Implements [IItem](#).

4.2.2.7 `int BombStandard::getRangeBonus () const`

get range bonus

Returns

`int _rangeBonus`

4.2.2.8 `int BombStandard::getSpeedBonus () const`

get speed bonus

Returns

`int _speedBonus`

4.2.2.9 `IAnimatedMeshSceneNode * BombStandard::isUsed (float x, float y) [virtual]`

plant the bomb at (x, y) position.

Parameters

<i>float</i>	x: position on x
<i>float</i>	y: position on y

Returns

IAnimatedSceneNode *node

Implements [IItem](#).

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

- Item/include/BombStandard.hpp
- Item/BombStandard.cpp

4.3 Button Class Reference

```
#include <Button.hpp>
```

Public Member Functions

- [Button](#) (const int pos[2], const int len[2], const std::string &name, int value)
constructor of the object button
- [~Button](#) ()
destructor of the object button
- bool [isClick](#) (int clickX, int clickY) const
- std::string [getName](#) () const
get the button name
- int [getLenX](#) () const
get the x len
- int [getLenY](#) () const
get the y len
- int [getPosX](#) () const
get the x pos
- int [getPosY](#) () const
get the y pos

4.3.1 Constructor & Destructor Documentation

4.3.1.1 [Button::Button](#) (const int *pos*[2], const int *len*[2], const std::string & *name*, int *value*)

constructor of the object button

Parameters

<i>pos[2]</i>	pos x and y
<i>len[2]</i>	len x and y
<i>&name</i>	name of button
<i>value</i>	value of button

Returns

nothing

4.3.1.2 Button::~~Button ()

destructor of the object button

Returns

nothing

4.3.2 Member Function Documentation

4.3.2.1 int Button::getLenX () const

get the x len

Returns

int _lenX

4.3.2.2 int Button::getLenY () const

get the y len

Returns

int _lenY

4.3.2.3 std::string Button::getName () const

get the button name

Returns

std::string _name

4.3.2.4 int Button::getPosX () const

get the x pos

Returns

int _posX

4.3.2.5 int Button::getPosY () const

get the y pos

Returns

int _posY

4.3.2.6 bool Button::isClick (int clickX, int clickY) const

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

- [include/Button.hpp](#)
- [Menu/Button.cpp](#)

4.4 Collision Class Reference

```
#include <Collision.hpp>
```

Public Member Functions

- [Collision](#) (int mapWidth, int mapHeight)
constructor (set the map size)
- [~Collision](#) ()
destructor
- bool [checkCollision](#) (int x, int y, std::shared_ptr< [Map](#) > [Map](#))
search for an empty cube or not
- int [getFarTen](#) (int toFind)
round to ten

4.4.1 Constructor & Destructor Documentation

4.4.1.1 Collision::Collision (int mapWidth, int mapHeight)

constructor (set the map size)

Parameters

<i>mapWidth</i>	width of the map
<i>mapHeight</i>	height of the map

Returns

nothing

4.4.1.2 Collision::~~Collision ()

destructor

Returns

nothing

4.4.2 Member Function Documentation

4.4.2.1 bool Collision::checkCollision (int x, int y, std::shared_ptr< Map > Map)

search for an empty cube or not

Parameters

<i>x</i>	position on x
<i>y</i>	position on y
<i>std::shared_ptr< Map ></i>	Map: Map

Returns

true or false

4.4.2.2 int Collision::getFarTen (int toFind)

round to ten

Parameters

<i>int</i>	toFind: number to round
------------	-------------------------

Returns

i or 0

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

- [include/Collision.hpp](#)
- [Collision/Collision.cpp](#)

4.5 Credits Class Reference

```
#include <Credits.hpp>
```

Public Member Functions

- [Credits](#) ()
credits constructor (init credits ressources)
- [~Credits](#) ()
credits destructor
- [actualState displayAll](#) (std::shared_ptr< [Graphical](#) > [Graphical](#))
display the credits
- bool [displayLbyL](#) (std::shared_ptr< [Graphical](#) > [Graphical](#))
do the credits movement
- void [displayText](#) (std::shared_ptr< [Graphical](#) > [Graphical](#))
draw the text
- void [displayPic](#) ()

4.5.1 Constructor & Destructor Documentation

4.5.1.1 Credits::Credits ()

credits constructor (init credits ressources)

Returns

nothing

4.5.1.2 Credits::~~Credits ()

credits destructor

Returns

nothing

4.5.2 Member Function Documentation

4.5.2.1 actualState Credits::displayAll (std::shared_ptr< [Graphical](#) > [Graphical](#))

display the credits

Parameters

<code>std::shared_ptr<Graphical></code>	Graphical : graphical object
---	--

Returns

actualState::CREDITS

4.5.2.2 bool Credits::displayLbyL (std::shared_ptr< Graphical > Graphical)

do the credits movement

Parameters

<code>std::shared_ptr<Graphical></code>	Graphical : graphical object
---	--

Returns

true or false

4.5.2.3 void Credits::displayPic ()

4.5.2.4 void Credits::displayText (std::shared_ptr< Graphical > Graphical)

draw the text

Parameters

<code>std::shared_ptr<Graphical></code>	Graphical : graphical object
---	--

Returns

nothing

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

- include/[Credits.hpp](#)
- Credits/[Credits.cpp](#)

4.6 Cube Class Reference

```
#include <Cube.hpp>
```

Public Types

- enum [TypeBox](#) {
[NORMAL](#), [BORDER](#), [SPEED](#), [RANGE](#),
[BOMB](#), [NOBOX](#), [FLOOR](#) }

Public Member Functions

- [Cube](#) (int width, int height, irr::scene::ISceneManager *manager, irr::video::IVideoDriver *driver, bool destructable, [TypeBox](#) type)
Construction of the object cube.
- [~Cube](#) ()
destruction of the cube object.
- int [getposx](#) () const
get x pos.
- int [getposy](#) () const
get y pos.
- irr::scene::IMeshSceneNode * [getMesh](#) () const
get the mesh
- bool [isDestruct](#) () const
know if the block is destroyed
- void [setDestruct](#) ()
destroy box
- bool [isDestructable](#) () const
know if i can destroy a block
- [Cube::TypeBox](#) [getCubeType](#) () const
get the cube type

4.6.1 Member Enumeration Documentation

4.6.1.1 enum [Cube::TypeBox](#)

Enumerator

NORMAL
BORDER
SPEED
RANGE
BOMB
NOBOX
FLOOR

4.6.2 Constructor & Destructor Documentation

- 4.6.2.1 [Cube::Cube](#) (int width, int height, irr::scene::ISceneManager * manager, irr::video::IVideoDriver * driver, bool destructable, [TypeBox](#) type)

Construction of the object cube.

Parameters

<i>height</i>	height of the cube
<i>width</i>	width of the cube
<i>manager</i>	manager of the Scene
<i>driver</i>	vidéo handling
<i>destructable</i>	Cube destructable or no
<i>TypeBox</i>	the type of box

Returns

nothing

4.6.2.2 `Cube::~~Cube ()`

destruction of the cube object.

Returns

nothing

4.6.3 Member Function Documentation

4.6.3.1 `Cube::TypeBox Cube::getCubeType () const`

get the cube type

Returns

[Cube::TypeBox](#) _type

4.6.3.2 `irr::scene::IMeshSceneNode * Cube::getMesh () const`

get the mesh

Returns

`irr::scene::IMeshSceneNode *_cube`

4.6.3.3 `int Cube::getposx () const`

get x pos.

Returns

`int _width`

4.6.3.4 int Cube::getposy () const

get y pos.

Returns

int _height

4.6.3.5 bool Cube::isDestruct () const

know if the block is destroyed

Returns

true or false

4.6.3.6 bool Cube::isDestructable () const

know if i can destroy a block

Returns

true or false

4.6.3.7 void Cube::setDestruct ()

destroy box

Returns

nothing

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

- [include/Cube.hpp](#)
- [Map/Cube.cpp](#)

4.7 Floor Class Reference

```
#include <Floor.hpp>
```

Public Member Functions

- [Floor](#) (int height, int width, irr::scene::ISceneManager *manager, irr::video::IVideoDriver *driver)
Construction of the map floor.
- [~Floor](#) ()
destructor
- int [getHeight](#) () const
get the width of the floor
- int [getWidth](#) () const
get the width of the floor

4.7.1 Constructor & Destructor Documentation

4.7.1.1 Floor::Floor (int *height*, int *width*, irr::scene::ISceneManager * *manager*, irr::video::IVideoDriver * *driver*)

Construction of the map floor.

Parameters

<i>height</i>	height of the floor
<i>width</i>	width of the floor
<i>manager</i>	manager of the Scene
<i>driver</i>	vidéo handling

Returns

nothing

4.7.1.2 Floor::~~Floor ()

destructor

Returns

nothing

4.7.2 Member Function Documentation**4.7.2.1 int Floor::getHeight () const**

get the width of the floor

Returns

int _height

4.7.2.2 int Floor::getWidth () const

get the width of the floor

Returns

int _width

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

- [include/Floor.hpp](#)
- [Map/Floor.cpp](#)

4.8 GetEvent Class Reference

```
#include <GetEvent.hpp>
```

Inherits IEventReceiver.

Public Member Functions

- [GetEvent](#) ()
constructor (init event values)
- [~GetEvent](#) ()
- bool [OnEvent](#) (const irr::SEvent &event)
getting the event clicked and the mouse position
- bool [isLeftClick](#) () const
catch left click
- bool [isRightClick](#) () const
catch right click
- irr::core::position2di [getMousePos](#) () const
get the mouse position
- bool [isKeyPressed](#) (int keyNum) const
know if a key is pressed

4.8.1 Constructor & Destructor Documentation

4.8.1.1 GetEvent::GetEvent ()

constructor (init event values)

destructor

Returns

nothing

4.8.1.2 GetEvent::~~GetEvent ()

4.8.2 Member Function Documentation

4.8.2.1 position2di GetEvent::getMousePos () const

get the mouse position

Returns

position2di _mousePos

4.8.2.2 bool GetEvent::isKeyPressed (int keyNum) const

know if a key is pressed

Returns

true or false

4.8.2.3 bool GetEvent::isLeftClick () const

catch left click

Returns

nothing

4.8.2.4 bool GetEvent::isRightClick () const

catch right click

Returns

nothing

4.8.2.5 bool GetEvent::OnEvent (const irr::SEvent & *event*)

getting the event clicked and the mouse position

Parameters

<i>event</i>	Handle event
--------------	--------------

Returns

true or false

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

- [include/GetEvent.hpp](#)
- [Graphical/GetEvent.cpp](#)

4.9 Graphical Class Reference

```
#include <Graphical.hpp>
```

Public Member Functions

- [Graphical](#) ()
constructor
- [~Graphical](#) ()
destructor
- bool [initGraphical](#) ()

initialisation of graphics

- void [drawText](#) (const std::string &toDisplay, int pos[4], bool adapToText)
- void [addCamera](#) (const irr::core::vector3df &position, const irr::core::vector3df &lookat)

add a camera

- void [drawButton](#) (const std::string &toDisplay, int pos[4], const std::string &backPic)
- void [drawMessageBox](#) (const std::string &toDisplay, int pos[2], int len[2])
- void [drawTexture](#) (const std::string &picPath, int origPos[2], int picLen[4])
- void [start](#) ()

start the scene

- void [end](#) ()

end the scene

- void [drawGUI](#) ()

draw the gui

- void [drawScene](#) ()

drew the scene

- irr::core::position2di [getMousePos](#) ()
- [GetEvent](#) [getEvent](#) ()

get the event

- irr::scene::ISceneManager * [getScene](#) ()

get the scene

- irr::video::IVideoDriver * [getVideo](#) ()

get the video driver

- irr::IrrlichtDevice * [getDevice](#) ()

get the device

- bool [isRightClick](#) ()
- bool [isLeftClick](#) ()
- void [initCamera](#) ()
- void [setCamera](#) (float x, float y, float z)

set the camera position then add it

4.9.1 Constructor & Destructor Documentation

4.9.1.1 Graphical::Graphical ()

constructor

Returns

nothing

4.9.1.2 Graphical::~~Graphical ()

destructor

Returns

nothing

4.9.2 Member Function Documentation

4.9.2.1 void Graphical::addCamera (const irr::core::vector3df & *position*, const irr::core::vector3df & *lookat*)

add a camera

Parameters

<i>const</i>	irr::core::vector3df &position: (x,y,z) position of the eye
<i>const</i>	irr::core::vector3df &lookat: (x,y,z) position of the point of view

Returns

nothing

4.9.2.2 void Graphical::drawButton (const std::string & *toDisplay*, int *pos*[4], const std::string & *backPic*)

4.9.2.3 void Graphical::drawGUI ()

draw the gui

Returns

nothing

4.9.2.4 void Graphical::drawMessageBox (const std::string & *toDisplay*, int *pos*[2], int *len*[2])

4.9.2.5 void Graphical::drawScene ()

drew the scene

Returns

nothing

4.9.2.6 void Graphical::drawText (const std::string & *toDisplay*, int *pos*[4], bool *adapToText*)

4.9.2.7 void Graphical::drawTexture (const std::string & *picPath*, int *origPos*[2], int *picLen*[4])

4.9.2.8 void Graphical::end ()

end the scene

Returns

nothing

4.9.2.9 IrrlichtDevice * Graphical::getDevice ()

get the device

Returns

IrrlichtDevice *_device

4.9.2.10 GetEvent Graphical::getEvent ()

get the event

Returns

GetEvent _event

4.9.2.11 irr::core::position2di Graphical::getMousePos ()

4.9.2.12 ISceneManager * Graphical::getScene ()

get the scene

Returns

ISceneManager *_scene

4.9.2.13 IVideoDriver * Graphical::getVideo ()

get the video driver

Returns

IVideodriver *_video

4.9.2.14 void Graphical::initCamera ()

4.9.2.15 bool Graphical::initGraphical ()

initialisation of graphics

Returns

true or false

4.9.2.16 bool Graphical::isLeftClick ()

4.9.2.17 bool Graphical::isRightClick ()

4.9.2.18 void Graphical::setCamera (float x, float y, float z)

set the camera position then add it

Parameters

<i>float</i>	x: position on x
<i>float</i>	y: position on y
<i>float</i>	z: position on z

Returns

nothing

4.9.2.19 void Graphical::start ()

start the scene

Returns

nothing

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

- [include/Graphical.hpp](#)
- [Graphical/Graphical.cpp](#)

4.10 IA Class Reference

```
#include <IA.hpp>
```

Public Types

- enum [IaDir](#) {
[UP](#), [DOWN](#), [LEFT](#), [RIGHT](#),
[LAST](#) }

Public Member Functions

- [IA](#) (irr::scene::ISceneManager *smgr, irr::video::IVideoDriver *Driver, std::shared_ptr< [Graphical](#) > [Graphical](#))
Construction of the [IA](#) and the character [IA](#).
- [~IA](#) ()
destructor of [IA](#)
- void [findSpawn](#) (std::shared_ptr< [Map](#) > [Map](#), std::shared_ptr< [Collision](#) > [Collision](#))
searching for the good spawn
- void [moveIA](#) (std::shared_ptr< [Map](#) > [Map](#), std::shared_ptr< [Collision](#) > [Collision](#))
move the [IA](#) int the better direction
- float [getZ](#) () const
get Z position

- float `getX` () const
get Z position
- std::pair< int, int > `getIaNewPos` (std::shared_ptr< Map > Map, std::shared_ptr< Collision > Collision)
Getting a new good position for Ia.
- IaDir `checkIfContinue` (std::shared_ptr< Map > Map, std::shared_ptr< Collision > Collision)
helping the ia to be more smart and running in a single direction
- IaDir `checkIfNotLast` (std::shared_ptr< Map > Map, std::shared_ptr< Collision > Collision)
check if not last
- void `dropBomb` (std::shared_ptr< Map > Map, std::shared_ptr< Collision > Collision)
drop a bomb
- void `timeToExplode` (std::shared_ptr< Map > Map)
timer before explosion
- void `deleteBomb` (std::shared_ptr< Map > Map)
delete the bomb after explosion
- std::shared_ptr< IItem > `getItem` ()
get the items
- void `invExplodeBomb` (int f, int s, std::shared_ptr< Map > Map)
get the positions and see who died
- void `setPPos` (float a, float b)
- bool `findPosNoSmash` (std::shared_ptr< Map > Map, std::shared_ptr< Collision > Collision, std::pair< int, int > pos)
find a position out of bomb range
- bool `nearPlayer` ()
see where is the player
- bool `wallAround` (std::shared_ptr< Map > Map, std::shared_ptr< Collision > Collision)
checking if there is wall around IA
- std::string `getPlayerDead` () const
get player dead
- std::pair< int, int > `BetterPosAround` (std::shared_ptr< Map > Map, std::shared_ptr< Collision > Collision)
check the collision and the position and know what's the best pos for IA
- float `calcBombDirst` (float _x, float _z)
calcul the distance between bomb and position

4.10.1 Member Enumeration Documentation

4.10.1.1 enum Ia::IaDir

Enumerator

UP
DOWN
LEFT
RIGHT
LAST

4.10.2 Constructor & Destructor Documentation

4.10.2.1 Ia::Ia (irr::scene::ISceneManager * smgr, irr::video::IVideoDriver * Driver, std::shared_ptr< Graphical > Graphical)

Construction of the `Ia` and the character `Ia`.

Parameters

<i>smgr</i>	managing the scene
<i>Driver</i>	handling vidéo
<i>Graphical</i>	managing the graphics of character

Returns

nothing

4.10.2.2 `IA::~~IA ()`

destructor of [IA](#)

Returns

nothing

4.10.3 Member Function Documentation

4.10.3.1 `std::pair< int, int > IA::BetterPosAround (std::shared_ptr< Map > Map, std::shared_ptr< Collision > Collision)`

check the collision and the position and know what's the best pos for [IA](#)

Parameters

<i>Map</i>	a map that contain map
<i>Collision</i>	handle the collision

Returns

`std::pair<int, int>` position

4.10.3.2 `float IA::calcBombDirst (float _x, float _s)`

calcul the distance between bomb and position

Parameters

\leftrightarrow _ \leftrightarrow x	position z
\leftrightarrow _ \leftrightarrow z	position x

Returns

float

4.10.3.3 IA::laDir IA::checkIfContinue (std::shared_ptr< Map > Map, std::shared_ptr< Collision > Collision)

helping the ia to be more smart and running in a single direction

Parameters

<i>Map</i>	map that contain the map
<i>Collision</i>	handle the collision of character

Returns

laDir

4.10.3.4 IA::laDir IA::checkIfNotLast (std::shared_ptr< Map > Map, std::shared_ptr< Collision > Collision)

check if not last

Parameters

<i>Map</i>	map that contain the map
<i>Collision</i>	handle the collision of character

Returns

laDir

4.10.3.5 void IA::deleteBomb (std::shared_ptr< Map > Map)

delete the bomb after explosion

Parameters

<i>Map</i>	a map that contain map
------------	------------------------

Returns

nothing

4.10.3.6 void IA::dropBomb (std::shared_ptr< Map > Map, std::shared_ptr< Collision > Collision)

drop a bomb

Parameters

<i>Map</i>	a map that contain map
<i>Collision</i>	handle the collision

Returns

nothing

4.10.3.7 `bool IA::findPosNoSmash (std::shared_ptr< Map > Map, std::shared_ptr< Collision > Collision, std::pair< int, int > pos)`

find a position out of bomb range

Parameters

<i>Map</i>	map that contain the map
<i>Collision</i>	handle the collision of character
<i>pos</i>	positions of <i>IA</i> in the map

Returns

true or false

4.10.3.8 `void IA::findSpawn (std::shared_ptr< Map > Map, std::shared_ptr< Collision > Collision)`

searching for the good spawn

Parameters

<i>Map</i>	map that contain the map
<i>Collision</i>	handle the collision of the character

Returns

nothing

4.10.3.9 `std::pair< int, int > IA::getIaNewPos (std::shared_ptr< Map > Map, std::shared_ptr< Collision > Collision)`

Getting a new good position for Ia.

Parameters

<i>Map</i>	map that contain the map
<i>Collision</i>	handle the collision of character

Returns

`std::pair<int, int>` position

4.10.3.10 `std::shared_ptr< IItem > IA::getItem ()`

get the items

Returns

`std::shared_ptr<IItem>`

4.10.3.11 `std::string IA::getPlayerDead () const`

get player dead

Returns

return `std::string _playerDead`

4.10.3.12 `float IA::getX () const`

get Z position

Returns

`float _x`

4.10.3.13 `float IA::getZ () const`

get Z position

Returns

`float _z`

4.10.3.14 `void IA::invExplodeBomb (int f, int s, std::shared_ptr< Map > Map)`

get the positions and see who died

Parameters

<i>Map</i>	map type that contain the map
<i>f</i>	first pos
<i>s</i>	second pos

Returns

nothing

4.10.3.15 void IA::moveIA (std::shared_ptr< Map > Map, std::shared_ptr< Collision > Collision)

move the IA int the better direction

Parameters

Map	map that contain the map
Collision	handle the collision of character

Returns

nothing

4.10.3.16 bool IA::nearPlayer ()

see where is the player

Returns

true or false

4.10.3.17 void IA::setPPos (float a, float b)

4.10.3.18 void IA::timeToExplode (std::shared_ptr< Map > Map)

timer before explosion

Parameters

Map	map type that contain the map
-----	-------------------------------

Returns

nothing

4.10.3.19 bool IA::wallAround (std::shared_ptr< Map > Map, std::shared_ptr< Collision > Collision)

checking if there is wall around IA

Parameters

Map	Map that contain the map
Collision	handle the collision of the character

Returns

true or false

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

- [include/IA.hpp](#)
- [IA/IA.cpp](#)

4.11 Item Class Reference

```
#include <Item.hpp>
```

Inherited by [BombStandard](#).

Public Member Functions

- virtual [~Item](#) ()=default
- virtual IAnimatedMeshSceneNode * [isUsed](#) (float x, float z)=0
- virtual std::string [getItemName](#) ()=0
- virtual int [getId](#) ()=0
- virtual bool [explodeBomb](#) (int x, int y, std::shared_ptr< [Map](#) > [Map](#), std::vector< std::pair< int, int >> [playerPos](#))=0
- virtual std::pair< int, int > [getPlayerDead](#) () const =0

4.11.1 Constructor & Destructor Documentation

4.11.1.1 virtual Item::~Item () [virtual],[default]

4.11.2 Member Function Documentation

4.11.2.1 virtual bool Item::explodeBomb (int x, int y, std::shared_ptr< [Map](#) > [Map](#), std::vector< std::pair< int, int >> [playerPos](#)) [pure virtual]

Implemented in [BombStandard](#).

4.11.2.2 virtual int Item::getId () [pure virtual]

Implemented in [BombStandard](#).

4.11.2.3 `virtual std::string lItem::getItemName () [pure virtual]`

Implemented in [BombStandard](#).

4.11.2.4 `virtual std::pair<int, int> lItem::getPlayerDead () const [pure virtual]`

Implemented in [BombStandard](#).

4.11.2.5 `virtual IAnimatedMeshSceneNode* lItem::isUsed (float x, float z) [pure virtual]`

Implemented in [BombStandard](#).

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

- [include/lItem.hpp](#)

4.12 Intro Class Reference

```
#include <Intro.hpp>
```

Public Member Functions

- [Intro \(\)](#)
introduction constructor (init pos and text)
- [~Intro \(\)](#)
introduction destructor
- [actualState displayAll](#) (std::shared_ptr< [Graphical](#) > [Graphical](#))
display the introduction
- bool [displayLbyL](#) (std::shared_ptr< [Graphical](#) > [Graphical](#))
do the introduction move
- void [displayText](#) (std::shared_ptr< [Graphical](#) > [Graphical](#))
draw the title
- void [displayPic](#) ()

4.12.1 Constructor & Destructor Documentation

4.12.1.1 `Intro::Intro ()`

introduction constructor (init pos and text)

Returns

nothing

4.12.1.2 Intro::~Intro ()

introduction destructor

Returns

nothing

4.12.2 Member Function Documentation

4.12.2.1 actualState Intro::displayAll (std::shared_ptr< Graphical > *Graphical*)

display the introduction

Parameters

<code>std::shared_ptr< Graphical ></code>	Graphical : graphical object
---	--

Returns

actualState::INTRO

4.12.2.2 bool Intro::displayLbyL (std::shared_ptr< Graphical > *Graphical*)

do the introduction move

Parameters

<code>std::shared_ptr< Graphical ></code>	Graphical : graphical object
---	--

Returns

true or false

4.12.2.3 void Intro::displayPic ()

4.12.2.4 void Intro::displayText (std::shared_ptr< Graphical > *Graphical*)

draw the title

Parameters

<code>std::shared_ptr< Graphical ></code>	Graphical : graphical object
---	--

Returns

nothing

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

- include/[Intro.hpp](#)
- Introduction/[Intro.cpp](#)

4.13 Map Class Reference

```
#include <Map.hpp>
```

Public Member Functions

- [Map](#) (irr::scene::ISceneManager *manager, irr::video::IVideoDriver *driver, std::map< std::string, int > v↔ Param)
build and initialise all value the class map need to draw it
- [~Map](#) ()
- bool [isValidPos](#) (std::pair< int, int > pos)
check if the position is valid or no
- void [createFloor](#) ()
call to [Floor](#) constructor who create the floor
- void [generateBorder](#) ()
generating all the border by calling [generateWidthMap\(\)](#) and [generatWidthMap\(\)](#).
- void [generateWidthMap](#) ()
generating the width map with border [Cube](#).
- void [generateHeightMap](#) ()
- void [generateMap](#) ()
generate the map and the boxes with calling [generatingTypeBox\(\)](#)
- [Cube::TypeBox](#) [generateTypeBox](#) ()
generating the type of box we need and call to [generateTypeBoxRt\(\)](#)
- [Cube::TypeBox](#) [generateTypeBoxRt](#) (float rdm, float caseWSpeed, float caseWSRange, float caseWBoost)
generating the type of box we need
- bool [isInit](#) () const
know if the map is initialized
- bool [setCubeDeleted](#) (int x, int y)
this function set if a cube is possible to delete
- bool [isCubeNotEmpty](#) (int x, int y, bool isToDestroy)
this function check if is it ok to destroy a cube
- int [getHeight](#) () const
get the height of map
- int [getWidth](#) () const
get the width of map
- int [getNbrBomb](#) () const
get the bomb number
- void [addBonus](#) ()
add the bonus
- int [getRange](#) () const
- int [getSpeed](#) () const
- [Cube::TypeBox](#) [getLastBonus](#) () const
get the last bonus

4.13.1 Constructor & Destructor Documentation

4.13.1.1 `Map::Map (irr::scene::ISceneManager * manager, irr::video::IVideoDriver * driver, std::map< std::string, int > vParam)`

build and initialise all value the class map need to draw it

Parameters

<i>manager</i>	the manager of the scene
<i>driver</i>	for handling video
<i>vParam</i>	the parameters of the map

Returns

nothing

4.13.1.2 `Map::~~Map ()`

4.13.2 Member Function Documentation

4.13.2.1 `void Map::addBonus ()`

add the bonus

Returns

nothing

4.13.2.2 `void Map::createFloor ()`

call to [Floor](#) constructor who create the floor

Returns

nothing

4.13.2.3 `void Map::generateBorder ()`

generating all the border by calling [generateWidthMap\(\)](#) and [generatWidthMap\(\)](#).

Returns

nothing

4.13.2.4 void Map::generateHeightMap ()

4.13.2.5 void Map::generateMap ()

generate the map and the boxes with calling generatingTypeBox()

Returns

nothing

4.13.2.6 Cube::TypeBox Map::generateTypeBox ()

generating the type of box we need and call to [generateTypeBoxRt\(\)](#)

Returns

a [Cube::TypeBox](#)

4.13.2.7 Cube::TypeBox Map::generateTypeBoxRt (float *rdm*, float *caseWSpeed*, float *caseWSRange*, float *caseWBoost*)

generating the type of box we need

Parameters

<i>rdm</i>	random
<i>caseWSpeed</i>	case box of speed
<i>caseWSRange</i>	case box of range
<i>caseWBoost</i>	case box boost

Returns

a [Cube::TypeBox](#)

4.13.2.8 void Map::generateWidthMap ()

generating the width map with border [Cube](#).

Returns

nothing

4.13.2.9 int Map::getHeight () const

get the height of map

Returns

int _height

4.13.2.10 `Cube::TypeBox Map::getLastBonus () const`

get the last bonus

Returns

`Cube::TypeBox _lastBonus`

4.13.2.11 `int Map::getNbrBomb () const`

get the bomb number

Returns

`int _nbrbomb`

4.13.2.12 `int Map::getRange () const`

4.13.2.13 `int Map::getSpeed () const`

4.13.2.14 `int Map::getWidth () const`

get the width of map

Returns

`int _width`

4.13.2.15 `bool Map::isCubeNotEmpty (int x, int y, bool isToDestroy)`

this function check if is it ok to destroy a cube

Parameters

<code>x</code>	the position x of cube
<code>y</code>	position y of the <code>Cube</code>
<code>isToDestroy</code>	give if is it to destroy a cube or no

Returns

true or false

4.13.2.16 `bool Map::isInit () const`

know if the map is initialized

Returns

true or false

4.13.2.17 `bool Map::isValidPos (std::pair< int, int > pos)`

check if the position is valid or no

Parameters

<i>pos</i>	positions to check
------------	--------------------

Returns

true or false

4.13.2.18 `bool Map::setCubeDeleted (int x, int y)`

this function set if a cube is possible to delete

Parameters

<i>x</i>	position x of the Cube
<i>y</i>	position y of the Cube

Returns

true or false

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

- [include/Map.hpp](#)
- [Map/Map.cpp](#)

4.14 Menu Class Reference

```
#include <Menu.hpp>
```

Public Member Functions

- [Menu](#) ()
constructor of [Menu](#)
- [~Menu](#) ()
destructor of [Menu](#)
- bool [initButton](#) ()

- initialize the buttons*
- void [initEquivAction](#) ()
- initialisation action with state*
- void [setEquivAction](#) (const std::string &nameAction, [actualState](#) myAction)
- assign a string to enumeration*
- [actualState](#) [getClickButton](#) (const int &clickX, const int &clickY, bool isClick)
- get the action did by the user and handle the next*
- bool [isRun](#) () const
- check if the menu is running*
- std::vector< [Button](#) > [getButton](#) () const
- get the button*
- void [drawMenu](#) (std::shared_ptr< [Graphical](#) > [Graphical](#))
- drawing the menu*
- void [setMultiplayer](#) (bool multi)
- set multiplayer mode*
- void [setGame](#) (bool game)

4.14.1 Constructor & Destructor Documentation

4.14.1.1 [Menu::Menu](#) ()

constructor of [Menu](#)

Returns

nothing

4.14.1.2 [Menu::~~Menu](#) ()

destructor of [Menu](#)

Returns

nothing

4.14.2 Member Function Documentation

4.14.2.1 void [Menu::drawMenu](#) (std::shared_ptr< [Graphical](#) > [Graphical](#))

drawing the menu

Parameters

Graphical	handle the graphic side
---------------------------	-------------------------

Returns

nothing

4.14.2.2 `std::vector< Button > Menu::getButton () const`

get the button

Returns

`std::vector<Button> _button`

4.14.2.3 `actualState Menu::getClickButton (const int & clickX, const int & clickY, bool isClick)`

get the action did by the user and handle the next

Parameters

<i>clickX</i>	position X of click
<i>clickY</i>	position Y of click
<i>isClick</i>	is it clicked

Returns

actualstate of menu

4.14.2.4 `bool Menu::initButton ()`

initialize the buttons

Returns

true or false

4.14.2.5 `void Menu::initEquivAction ()`

initalsion action with state

Returns

nothing

4.14.2.6 `bool Menu::isRun () const`

check if the menu is running

Returns

true or false

4.14.2.7 `void Menu::setEquivAction (const std::string & nameAction, actualState myAction)`

assign a string to enumeration

Returns

nothing

4.14.2.8 `void Menu::setGame (bool game)`

4.14.2.9 `void Menu::setMultiplayer (bool game)`

set multiplayer mode

set game mode

Parameters

<i>bool</i>	multi: multi state
-------------	--------------------

Returns

nothing

Parameters

<i>bool</i>	game: game state
-------------	------------------

Returns

nothing

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

- [include/Menu.hpp](#)
- [Menu/Menu.cpp](#)

4.15 Param Class Reference

```
#include <Param.hpp>
```

Public Member Functions

- [Param](#) ()
building of all parameters
- [~Param](#) ()
destructor of [Param](#)
- bool [initButton](#) ()
initialization of all buttons in param
- void [initIntoButton](#) ()
inititalize the value of paramaters into their buttons
- void [initEquivAction](#) ()
initialization of events with button into parameter
- void [initButtonLimits](#) ()
init the limits of button with setters
- void [setButtonLimits](#) (const std::string &nameButton, int lBottom, int lTop)
set the buttons limit
- void [setIntoButton](#) (const std::string &nameButton, int value)
assign a string to an enumeration
- void [setEquivButton](#) (const std::string &nameButton, const std::string &equivButton)
- bool [isRun](#) () const
- std::vector< [Button](#) > [getButton](#) () const
get the button
- void [drawParam](#) (std::shared_ptr< [Graphical](#) > [Graphical](#))
draw parameters
- void [drawSideButton](#) (std::shared_ptr< [Graphical](#) > [Graphical](#))
draw + or -
- void [drawNameButton](#) (std::shared_ptr< [Graphical](#) > [Graphical](#))
draw a name button
- void [drawReturnButton](#) (std::shared_ptr< [Graphical](#) > [Graphical](#))
draw the button return
- std::string [getFormatedStringButtonName](#) ([Button](#) b1)
get a formatted string of button name
- [actualState](#) [getClickButton](#) (const int &clickX, const int &clickY, bool isClick)
get the button clicked and and check what state we are
- void [newStateButton](#) ([Button](#) b1)
give the new state of button
- std::map< std::string, int > [getParam](#) () const
get param

4.15.1 Constructor & Destructor Documentation

4.15.1.1 Param::Param ()

building of all parameters

Returns

nothing

4.15.1.2 Param::~~Param ()

destructor of [Param](#)

Returns

nothing

4.15.2 Member Function Documentation

4.15.2.1 void Param::drawNameButton (std::shared_ptr< [Graphical](#) > *Graphical*)

draw a name button

Parameters

Graphical	handle the graphics to display
---------------------------	--------------------------------

Returns

nothing

4.15.2.2 void Param::drawParam (std::shared_ptr< [Graphical](#) > *Graphical*)

draw parameters

Parameters

<i>std::shared_ptr<Graphical></i>	Graphical : handle the graphical side
---	---

Returns

void

4.15.2.3 void Param::drawReturnButton (std::shared_ptr< [Graphical](#) > *Graphical*)

draw the button return

Parameters

Graphical	handle the graphics to display
---------------------------	--------------------------------

Returns

void

4.15.2.4 void Param::drawSideButton (std::shared_ptr< Graphical > Graphical)

draw + or -

Parameters

<i>Graphical</i>	handle the graphics to display
------------------	--------------------------------

Returns

void

4.15.2.5 std::vector< Button > Param::getButton () const

get the button

Returns

void

4.15.2.6 actualState Param::getClickButton (const int & clickX, const int & clickY, bool isClick)

get the button clicked and and check what state we are

Parameters

<i>clickX</i>	position X of click
<i>clickY</i>	position X of click
<i>isClick</i>	is it clicked or no

Returns

actualState

4.15.2.7 std::string Param::getFormattedStringButtonName (Button b1)

get a formatted string of button name

Parameters

<i>b1</i>	<i>Button</i> type
-----------	--------------------

Returns

std::string fButtonString

4.15.2.8 `std::map< std::string, int > Param::getParam () const`

get param

Returns

`std::map<std::string, int> _intoButton`

4.15.2.9 `bool Param::initButton ()`

initialization of all buttons in param

Returns

true or false

4.15.2.10 `void Param::initButtonLimits ()`

init the limits of button with setters

Returns

nothing

4.15.2.11 `void Param::initEquivAction ()`

initialization of events with button into parameter

Returns

void

4.15.2.12 `void Param::initIntoButton ()`

initialize the value of paramaters into their buttons

Returns

void

4.15.2.13 `bool Param::isRun () const`**4.15.2.14** `void Param::newStateButton (Button b1)`

give the new state of button

Parameters

<i>b1</i>	Button type
-----------	-----------------------------

Returns

void

4.15.2.15 void Param::setButtonLimits (const std::string & *nameButton*, int *lBottom*, int *lTop*)

set the buttons limit

Parameters

<i>const::std::string</i>	&nameButton: button name
<i>int</i>	lBottom: bottom limit
<i>int</i>	lTop: top limit

Returns

void

4.15.2.16 void Param::setEquivButton (const std::string & *nameButton*, const std::string & *equivButton*)

4.15.2.17 void Param::setIntoButton (const std::string & *nameButton*, int *value*)

assign a string to an enumeration

Returns

void

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

- include/[Param.hpp](#)
- Menu/[Param.cpp](#)

4.16 Song Class Reference

```
#include <Song.hpp>
```

Public Member Functions

- [Song](#) ()
: constructor of [Song](#)
- [~Song](#) ()
- void [playShortSong](#) (const char *song)
creating a engine and playing a bref song one time
- void [playLongSong](#) (const char *song)
creating the Engine and play song while the fn stop long song is not called
- void [stopLongSong](#) ()
destroying the engine and stoping a long song like menu or the song of the game

4.16.1 Constructor & Destructor Documentation

4.16.1.1 [Song::Song](#) ()

: constructor of [Song](#)

: destructor of [Song](#)

Returns

void

4.16.1.2 [Song::~~Song](#) ()

4.16.2 Member Function Documentation

4.16.2.1 void [Song::playLongSong](#) (const char * *song*)

creating the Engine and play song while the fn stop long song is not called

Parameters

<i>song</i>	name of song
-------------	--------------

Returns

void

4.16.2.2 void [Song::playShortSong](#) (const char * *song*)

creating a engine and playing a bref song one time

Parameters

<i>song</i>	name of song
-------------	--------------

Returns

void

4.16.2.3 void Song::stopLongSong ()

destroying the engine and stoping a long song like menu or the song of the game

Returns

void

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

- include/[Song.hpp](#)
- Song/[Song.cpp](#)

4.17 StudioCore Class Reference

```
#include <StudioCore.hpp>
```

Public Member Functions

- [StudioCore](#) ()
construction of the core of the game
- [~StudioCore](#) ()
destructor of the core of the game
- void [runCore](#) ()
run the graphics and draw the scene and the GUI
- bool [initCore](#) ()
initialisation of menu and parameters
- bool [initSelectState](#) ()
initialisation of states
- void [computeMapParam](#) ()
computing the parameter of map
- void [initBomberman](#) ()
initialisation of the bomberman character
- void [initMultiplayerBomberman](#) ()
initialisation of the bomberman for the multiplayer
- void [runDifferentCases](#) ()
select the action associated to the state
- void [runMenu](#) ()

- run the menu with graphics and song*
- void `runCredits` ()
run the credits
- void `runParam` ()
get events and run to parameters
- void `runGame` ()
playing song starting and running a game
- void `runMultiplayerGame` ()
play songs, start and run the multiplayer
- void `runIntro` ()
run the intro
- void `runNewGame` ()
restart a new game
- void `runDeath` ()
end the game when you die and run a new game

4.17.1 Constructor & Destructor Documentation

4.17.1.1 StudioCore::StudioCore ()

construction of the core of the game

Returns

nothing

4.17.1.2 StudioCore::~~StudioCore ()

destructor of the core of the game

Returns

nothing

4.17.2 Member Function Documentation

4.17.2.1 void StudioCore::computeMapParam ()

computing the parameter of map

Returns

nothing

4.17.2.2 void StudioCore::initBomberman ()

initialisation of the bomberman character

Returns

nothing

4.17.2.3 bool StudioCore::initCore ()

initialisation of menu and parameters

Returns

true or false

4.17.2.4 void StudioCore::initMultiplayerBomberman ()

initialisation of the bomberman for the multiplayer

Returns

nothing

4.17.2.5 bool StudioCore::initSelectState ()

initialisation of states

Returns

true or false

4.17.2.6 void StudioCore::runCore ()

run the graphics and draw the scene and the GUI

Returns

nothing

4.17.2.7 void StudioCore::runCredits ()

run the credits

Returns

nothing

4.17.2.8 void StudioCore::runDeath ()

end the game when you die and run a new game

Returns

nothing

4.17.2.9 void StudioCore::runDifferentCases ()

select the action associated to the state

Returns

nothing

4.17.2.10 void StudioCore::runGame ()

playing song starting and running a game

Returns

nothing

4.17.2.11 void StudioCore::runIntro ()

run the intro

Returns

nothing

4.17.2.12 void StudioCore::runMenu ()

run the menu with graphics and song

Returns

nothing

4.17.2.13 void StudioCore::runMultiplayerGame ()

play songs, start and run the multiplayer

Returns

nothing

4.17.2.14 void StudioCore::runNewGame ()

restart a new game

Returns

nothing

4.17.2.15 void StudioCore::runParam ()

get events and run to parameters

Returns

nothing

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

- [include/StudioCore.hpp](#)
- [Core/StudioCore.cpp](#)

Chapter 5

File Documentation

5.1 Collision/Collision.cpp File Reference

```
#include "Collision.hpp"
```

5.2 Core/StudioCore.cpp File Reference

```
#include "StudioCore.hpp"
```

5.3 Credits/Credits.cpp File Reference

```
#include "Credits.hpp"
```

5.4 Game/Bomberman.cpp File Reference

```
#include "Bomberman.hpp"
```

5.5 Graphical/GetEvent.cpp File Reference

```
#include "GetEvent.hpp"
```

5.6 Graphical/Graphical.cpp File Reference

```
#include "Graphical.hpp"
```

5.7 IA/IA.cpp File Reference

```
#include "IA.hpp"
```

5.8 include/Bomberman.hpp File Reference

```
#include <vector>
#include <chrono>
#include <iostream>
#include <ctime>
#include <cstdlib>
#include "Item.hpp"
#include "BombStandard.hpp"
#include "indieStudio.hpp"
#include "Graphical.hpp"
#include "Map.hpp"
#include "Collision.hpp"
#include "Song.hpp"
#include "IA.hpp"
```

Classes

- class [Bomberman](#)

Macros

- `#define` [MAXBOMB](#) 3

5.8.1 Macro Definition Documentation

5.8.1.1 `#define` MAXBOMB 3

5.9 include/Button.hpp File Reference

```
#include <string>
#include <iostream>
```

Classes

- class [Button](#)

5.10 include/Collision.hpp File Reference

```
#include <map>
#include <memory>
#include "Map.hpp"
```

Classes

- class [Collision](#)

5.11 include/Credits.hpp File Reference

```
#include "Graphical.hpp"
```

Classes

- class [Credits](#)

Macros

- `#define PERSO_PATH "introPerso.jpg"`

5.11.1 Macro Definition Documentation

5.11.1.1 `#define PERSO_PATH "introPerso.jpg"`

5.12 include/Cube.hpp File Reference

```
#include <iostream>
#include <map>
#include "irrlicht.h"
```

Classes

- class [Cube](#)

5.13 include/Floor.hpp File Reference

```
#include <map>
#include <memory>
#include "Cube.hpp"
#include "irrlicht.h"
```

Classes

- class [Floor](#)

5.14 include/GetEvent.hpp File Reference

```
#include "indieStudio.hpp"
```

Classes

- class [GetEvent](#)

5.15 include/Graphical.hpp File Reference

```
#include "indieStudio.hpp"
#include "GetEvent.hpp"
```

Classes

- class [Graphical](#)

Macros

- #define [FONT_PATH](#) `"/Ressource/bigfont.png"`
- #define [FTB](#) `"/Ressource/ftbW.png"`
- #define [S_TO_WS](#)(path) `std::wstring(path.begin(), path.end()).c_str()`

5.15.1 Macro Definition Documentation

5.15.1.1 `#define FONT_PATH "/Ressource/bigfont.png"`

5.15.1.2 `#define FTB "/Ressource/ftbW.png"`

5.15.1.3 `#define S_TO_WS(path) std::wstring(path.begin(), path.end()).c_str()`

5.16 include/IA.hpp File Reference

```
#include <vector>
#include <chrono>
#include <iostream>
#include <ctime>
#include <cstdlib>
#include <cmath>
#include "Item.hpp"
#include "indieStudio.hpp"
#include "BombStandard.hpp"
#include "Graphical.hpp"
#include "Map.hpp"
#include "Collision.hpp"
#include "Song.hpp"
```

Classes

- class [IA](#)

5.17 include/indieStudio.hpp File Reference

```
#include <iostream>
#include <cstdlib>
#include <memory>
#include <sstream>
#include <vector>
#include <map>
#include <functional>
#include "irrlicht.h"
```

Macros

- `#define WINDOW_WIDTH 1200`
- `#define WINDOW_HEIGHT 800`

Enumerations

- enum [actualState](#) {
 [INTRO](#), [MENU](#), [GAME](#), [CREDITS](#),
 [PARAM](#), [EXIT](#), [MULTIPLAYER](#), [NEWGAME](#),
 [DEATH](#) }

5.17.1 Macro Definition Documentation

5.17.1.1 `#define WINDOW_HEIGHT 800`

5.17.1.2 `#define WINDOW_WIDTH 1200`

5.17.2 Enumeration Type Documentation

5.17.2.1 `enum actualState`

Enumerator

INTRO
MENU
GAME
CREDITS
PARAM
EXIT
MULTIPLAYER
NEWGAME
DEATH

5.18 include/Intro.hpp File Reference

```
#include "Graphical.hpp"
```

Classes

- class [Intro](#)

Macros

- `#define` [PERSO_PATH](#) "introPerso.jpg"

5.18.1 Macro Definition Documentation

5.18.1.1 #define PERSO_PATH "introPerso.jpg"

5.19 include/Item.hpp File Reference

```
#include <cstdio>
#include <iostream>
#include <memory>
#include "indieStudio.hpp"
#include "Map.hpp"
```

Classes

- class [Item](#)

Enumerations

- enum [Item](#) { [BOMBS](#), [BOMBM](#) }

5.19.1 Enumeration Type Documentation

5.19.1.1 enum Item

Enumerator

BOMBS

BOMBM

5.20 include/Map.hpp File Reference

```
#include <iostream>
#include <map>
#include <memory>
#include <ctime>
#include <cstdlib>
#include <utility>
#include "irrlicht.h"
#include "Cube.hpp"
#include "Floor.hpp"
```

Classes

- class [Map](#)

Macros

- `#define S_TO_WS(path) std::wstring(path.begin(), path.end()).c_str()`
- `#define NBRBOMB 10000`

5.20.1 Macro Definition Documentation

5.20.1.1 `#define NBRBOMB 10000`

5.20.1.2 `#define S_TO_WS(path) std::wstring(path.begin(), path.end()).c_str()`

5.21 include/Menu.hpp File Reference

```
#include "Graphical.hpp"  
#include "Button.hpp"
```

Classes

- class [Menu](#)

Macros

- `#define FONT_MENU "/Ressource/disp.png"`

5.21.1 Macro Definition Documentation

5.21.1.1 `#define FONT_MENU "/Ressource/disp.png"`

5.22 include/Param.hpp File Reference

```
#include "Graphical.hpp"  
#include "Button.hpp"
```

Classes

- class [Param](#)

5.23 include/Song.hpp File Reference

```
#include "irrKlang.h"  
#include "indieStudio.hpp"
```

Classes

- class [Song](#)

5.24 include/StudioCore.hpp File Reference

```
#include <thread>
#include <chrono>
#include "Graphical.hpp"
#include "Menu.hpp"
#include "Intro.hpp"
#include "Credits.hpp"
#include "Param.hpp"
#include "Map.hpp"
#include "Collision.hpp"
#include "Bomberman.hpp"
#include "Song.hpp"
#include "IA.hpp"
```

Classes

- class [StudioCore](#)

Macros

- #define [MULTI_P](#) 2

5.24.1 Macro Definition Documentation

5.24.1.1 #define MULTI_P 2

5.25 Introduction/Intro.cpp File Reference

```
#include "Intro.hpp"
```

5.26 Item/BombStandard.cpp File Reference

```
#include "BombStandard.hpp"
```

5.27 Item/include/BombStandard.hpp File Reference

```
#include "Item.hpp"
```

Classes

- class [BombStandard](#)

5.28 main.cpp File Reference

```
#include "StudioCore.hpp"
```

Functions

- int [main](#) (int ac, char **av)

5.28.1 Function Documentation

5.28.1.1 int main (int *ac*, char ** *av*)

5.29 Map/Cube.cpp File Reference

```
#include "Cube.hpp"
```

5.30 Map/Floor.cpp File Reference

```
#include "Floor.hpp"
```

5.31 Map/Map.cpp File Reference

```
#include "Map.hpp"
```

5.32 Menu/Button.cpp File Reference

```
#include "Button.hpp"
```

5.33 Menu/Menu.cpp File Reference

```
#include "Menu.hpp"
```

5.34 Menu/Param.cpp File Reference

```
#include "Param.hpp"
```

5.35 Song/Song.cpp File Reference

```
#include "Song.hpp"
```


Index

- ~BombStandard
 - BombStandard, [13](#)
- ~Bomberman
 - Bomberman, [8](#)
- ~Button
 - Button, [17](#)
- ~Collision
 - Collision, [19](#)
- ~Credits
 - Credits, [20](#)
- ~Cube
 - Cube, [23](#)
- ~Floor
 - Floor, [26](#)
- ~GetEvent
 - GetEvent, [27](#)
- ~Graphical
 - Graphical, [29](#)
- ~IA
 - IA, [34](#)
- ~Item
 - Item, [39](#)
- ~Intro
 - Intro, [40](#)
- ~Map
 - Map, [43](#)
- ~Menu
 - Menu, [47](#)
- ~Param
 - Param, [50](#)
- ~Song
 - Song, [55](#)
- ~StudioCore
 - StudioCore, [57](#)
- actualState
 - indieStudio.hpp, [66](#)
- addBonus
 - BombStandard, [14](#)
 - Bomberman, [8](#)
 - Map, [43](#)
- addCamera
 - Graphical, [29](#)
- BOMBM
 - Item.hpp, [67](#)
- BOMBS
 - Item.hpp, [67](#)
- BOMB
 - Cube, [22](#)

- BORDER
 - Cube, [22](#)
- BetterPosAround
 - IA, [34](#)
- BombStandard, [13](#)
 - ~BombStandard, [13](#)
 - addBonus, [14](#)
 - BombStandard, [13](#)
 - explodeBomb, [14](#)
 - getId, [14](#)
 - getItemName, [14](#)
 - getNbrBombBonus, [15](#)
 - getPlayerDead, [15](#)
 - getRangeBonus, [15](#)
 - getSpeedBonus, [15](#)
 - isUsed, [15](#)
- Bomberman, [7](#)
 - ~Bomberman, [8](#)
 - addBonus, [8](#)
 - Bomberman, [8](#)
 - deleteBomb, [9](#)
 - dropBomb, [9](#)
 - findSpawn, [9](#)
 - getItem, [9](#)
 - getPlayerDead, [10](#)
 - getX, [10](#)
 - getZ, [10](#)
 - initBomberman, [10](#)
 - initKeyMove, [10](#)
 - invExplodeBomb, [10](#)
 - isRunning, [11](#)
 - moveBomberman, [11](#)
 - realizeDisplacement, [11](#)
 - setEnemy, [12](#)
 - setItem, [12](#)
 - setKeyMove, [12](#)
 - setMulti, [12](#)
 - timeToExplode, [12](#)
- Bomberman.hpp
 - MAXBOMB, [62](#)
- Button, [16](#)
 - ~Button, [17](#)
 - Button, [16](#)
 - getLenX, [17](#)
 - getLenY, [17](#)
 - getName, [17](#)
 - getPosX, [17](#)
 - getPosY, [18](#)
 - isClick, [18](#)

- CREDITS
 - indieStudio.hpp, 66
- calcBombDirst
 - IA, 34
- checkCollision
 - Collision, 19
- checkIfContinue
 - IA, 35
- checkIfNotLast
 - IA, 35
- Collision, 18
 - ~Collision, 19
 - checkCollision, 19
 - Collision, 18
 - getFarTen, 19
- Collision/Collision.cpp, 61
- computeMapParam
 - StudioCore, 57
- Core/StudioCore.cpp, 61
- createFloor
 - Map, 43
- Credits, 20
 - ~Credits, 20
 - Credits, 20
 - displayAll, 20
 - displayLbyL, 21
 - displayPic, 21
 - displayText, 21
- Credits.hpp
 - PERSO_PATH, 63
- Credits/Credits.cpp, 61
- Cube, 21
 - ~Cube, 23
 - BOMB, 22
 - BORDER, 22
 - Cube, 22
 - FLOOR, 22
 - getCubeType, 23
 - getMesh, 23
 - getposx, 23
 - getposy, 23
 - isDestruct, 24
 - isDestructable, 24
 - NOBOX, 22
 - NORMAL, 22
 - RANGE, 22
 - SPEED, 22
 - setDestruct, 24
 - TypeBox, 22
- DEATH
 - indieStudio.hpp, 66
- DOWN
 - IA, 33
- deleteBomb
 - Bombberman, 9
 - IA, 35
- displayAll
 - Credits, 20
- Intro, 41
- displayLbyL
 - Credits, 21
 - Intro, 41
- displayPic
 - Credits, 21
 - Intro, 41
- displayText
 - Credits, 21
 - Intro, 41
- drawButton
 - Graphical, 30
- drawGUI
 - Graphical, 30
- drawMenu
 - Menu, 47
- drawMessageBox
 - Graphical, 30
- drawNameButton
 - Param, 51
- drawParam
 - Param, 51
- drawReturnButton
 - Param, 51
- drawScene
 - Graphical, 30
- drawSideButton
 - Param, 51
- drawText
 - Graphical, 30
- drawTexture
 - Graphical, 30
- dropBomb
 - Bombberman, 9
 - IA, 35
- EXIT
 - indieStudio.hpp, 66
- end
 - Graphical, 30
- explodeBomb
 - BombStandard, 14
 - Item, 39
- FLOOR
 - Cube, 22
- FONT_MENU
 - Menu.hpp, 68
- FONT_PATH
 - Graphical.hpp, 65
- FTB
 - Graphical.hpp, 65
- findPosNoSmash
 - IA, 36
- findSpawn
 - Bombberman, 9
 - IA, 36
- Floor, 24
 - ~Floor, 26

- Floor, 25
- getHeight, 26
- getWidth, 26
- GAME
 - indieStudio.hpp, 66
- Game/Bombberman.cpp, 61
- generateBorder
 - Map, 43
- generateHeightMap
 - Map, 43
- generateMap
 - Map, 44
- generateTypeBox
 - Map, 44
- generateTypeBoxRt
 - Map, 44
- generateWidthMap
 - Map, 44
- getButton
 - Menu, 48
 - Param, 52
- getClickButton
 - Menu, 48
 - Param, 52
- getCubeType
 - Cube, 23
- getDevice
 - Graphical, 30
- GetEvent, 26
 - ~GetEvent, 27
 - GetEvent, 27
 - getMousePos, 27
 - isKeyPressed, 27
 - isLeftClick, 27
 - isRightClick, 28
 - OnEvent, 28
- getEvent
 - Graphical, 31
- getFarTen
 - Collision, 19
- getFormattedStringButtonName
 - Param, 52
- getHeight
 - Floor, 26
 - Map, 44
- getIaNewPos
 - IA, 36
- getId
 - BombStandard, 14
 - Item, 39
- getItem
 - Bombberman, 9
 - IA, 37
- getItemName
 - BombStandard, 14
 - Item, 39
- getLastBonus
 - Map, 44
- getLenX
 - Button, 17
- getLenY
 - Button, 17
- getMesh
 - Cube, 23
- getMousePos
 - GetEvent, 27
 - Graphical, 31
- getName
 - Button, 17
- getNbrBomb
 - Map, 45
- getNbrBombBonus
 - BombStandard, 15
- getParam
 - Param, 52
- getPlayerDead
 - BombStandard, 15
 - Bombberman, 10
 - Item, 40
 - IA, 37
- getPosX
 - Button, 17
- getPosY
 - Button, 18
- getRange
 - Map, 45
- getRangeBonus
 - BombStandard, 15
- getScene
 - Graphical, 31
- getSpeed
 - Map, 45
- getSpeedBonus
 - BombStandard, 15
- getVideo
 - Graphical, 31
- getWidth
 - Floor, 26
 - Map, 45
- getposx
 - Cube, 23
- getposy
 - Cube, 23
- getX
 - Bombberman, 10
 - IA, 37
- getZ
 - Bombberman, 10
 - IA, 37
- Graphical, 28
 - ~Graphical, 29
 - addCamera, 29
 - drawButton, 30
 - drawGUI, 30
 - drawMessageBox, 30
 - drawScene, 30

- drawText, 30
- drawTexture, 30
- end, 30
- getDevice, 30
- getEvent, 31
- getMousePos, 31
- getScene, 31
- getVideo, 31
- Graphical, 29
- initCamera, 31
- initGraphical, 31
- isLeftClick, 31
- isRightClick, 31
- setCamera, 31
- start, 32
- Graphical.hpp
 - FONT_PATH, 65
 - FTB, 65
 - S_TO_WS, 65
- Graphical/GetEvent.cpp, 61
- Graphical/Graphical.cpp, 62
- IA/IA.cpp, 62
- Item, 39
 - ~Item, 39
 - explodeBomb, 39
 - getId, 39
 - getItemName, 39
 - getPlayerDead, 40
 - isUsed, 40
- INTRO
 - indieStudio.hpp, 66
- IA, 32
 - ~IA, 34
 - BetterPosAround, 34
 - calcBombDir, 34
 - checkIfContinue, 35
 - checkIfNotLast, 35
 - DOWN, 33
 - deleteBomb, 35
 - dropBomb, 35
 - findPosNoSmash, 36
 - findSpawn, 36
 - getIaNewPos, 36
 - getItem, 37
 - getPlayerDead, 37
 - getX, 37
 - getZ, 37
 - IA, 33
 - IaDir, 33
 - invExplodeBomb, 37
 - LAST, 33
 - LEFT, 33
 - moveIA, 38
 - nearPlayer, 38
 - RIGHT, 33
 - setPPos, 38
 - timeToExplode, 38
 - UP, 33
 - wallAround, 38
- IaDir
 - IA, 33
- include/Bomberman.hpp, 62
- include/Button.hpp, 62
- include/Collision.hpp, 63
- include/Credits.hpp, 63
- include/Cube.hpp, 63
- include/Floor.hpp, 64
- include/GetEvent.hpp, 64
- include/Graphical.hpp, 64
- include/IA.hpp, 65
- include/Intro.hpp, 66
- include/Item.hpp, 67
- include/Map.hpp, 67
- include/Menu.hpp, 68
- include/Param.hpp, 68
- include/Song.hpp, 68
- include/StudioCore.hpp, 69
- include/indieStudio.hpp, 65
- indieStudio.hpp
 - actualState, 66
 - CREDITS, 66
 - DEATH, 66
 - EXIT, 66
 - GAME, 66
 - INTRO, 66
 - MENU, 66
 - MULTIPLAYER, 66
 - NEWGAME, 66
 - PARAM, 66
 - WINDOW_HEIGHT, 66
 - WINDOW_WIDTH, 66
- initBomberman
 - Bomberman, 10
 - StudioCore, 57
- initButton
 - Menu, 48
 - Param, 53
- initButtonLimits
 - Param, 53
- initCamera
 - Graphical, 31
- initCore
 - StudioCore, 58
- initEquivAction
 - Menu, 48
 - Param, 53
- initGraphical
 - Graphical, 31
- initIntoButton
 - Param, 53
- initKeyMove
 - Bomberman, 10
- initMultiplayerBomberman
 - StudioCore, 58
- initSelectState
 - StudioCore, 58

- Intro, [40](#)
 - ~Intro, [40](#)
 - displayAll, [41](#)
 - displayLbyL, [41](#)
 - displayPic, [41](#)
 - displayText, [41](#)
 - Intro, [40](#)
- Intro.hpp
 - PERSO_PATH, [67](#)
- Introduction/Intro.cpp, [69](#)
- invExplodeBomb
 - Bombberman, [10](#)
 - IA, [37](#)
- isClick
 - Button, [18](#)
- isCubeNotEmpty
 - Map, [45](#)
- isDestruct
 - Cube, [24](#)
- isDestructable
 - Cube, [24](#)
- isInit
 - Map, [45](#)
- isKeyPressed
 - GetEvent, [27](#)
- isLeftClick
 - GetEvent, [27](#)
 - Graphical, [31](#)
- isRightClick
 - GetEvent, [28](#)
 - Graphical, [31](#)
- isRun
 - Menu, [48](#)
 - Param, [53](#)
- isRunning
 - Bombberman, [11](#)
- isUsed
 - BombStandard, [15](#)
 - Item, [40](#)
- isValidPos
 - Map, [46](#)
- Item
 - Item.hpp, [67](#)
- Item.hpp
 - BOMBM, [67](#)
 - BOMBS, [67](#)
 - Item, [67](#)
- Item/BombStandard.cpp, [69](#)
- Item/include/BombStandard.hpp, [69](#)
- LAST
 - IA, [33](#)
- LEFT
 - IA, [33](#)
- MAXBOMB
 - Bombberman.hpp, [62](#)
- MENU
 - indieStudio.hpp, [66](#)

- MULTI_P
 - StudioCore.hpp, [69](#)
- MULTIPLAYER
 - indieStudio.hpp, [66](#)
- main
 - main.cpp, [70](#)
- main.cpp, [70](#)
 - main, [70](#)
- Map, [42](#)
 - ~Map, [43](#)
 - addBonus, [43](#)
 - createFloor, [43](#)
 - generateBorder, [43](#)
 - generateHeightMap, [43](#)
 - generateMap, [44](#)
 - generateTypeBox, [44](#)
 - generateTypeBoxRt, [44](#)
 - generateWidthMap, [44](#)
 - getHeight, [44](#)
 - getLastBonus, [44](#)
 - getNbrBomb, [45](#)
 - getRange, [45](#)
 - getSpeed, [45](#)
 - getWidth, [45](#)
 - isCubeNotEmpty, [45](#)
 - isInit, [45](#)
 - isValidPos, [46](#)
 - Map, [43](#)
 - setCubeDeleted, [46](#)
- Map.hpp
 - NBRBOMB, [68](#)
 - S_TO_WS, [68](#)
- Map/Cube.cpp, [70](#)
- Map/Floor.cpp, [70](#)
- Map/Map.cpp, [70](#)
- Menu, [46](#)
 - ~Menu, [47](#)
 - drawMenu, [47](#)
 - getButton, [48](#)
 - getClickButton, [48](#)
 - initButton, [48](#)
 - initEquivAction, [48](#)
 - isRun, [48](#)
 - Menu, [47](#)
 - setEquivAction, [49](#)
 - setGame, [49](#)
 - setMultiplayer, [49](#)
- Menu.hpp
 - FONT_MENU, [68](#)
- Menu/Button.cpp, [70](#)
- Menu/Menu.cpp, [70](#)
- Menu/Param.cpp, [71](#)
- moveBombberman
 - Bombberman, [11](#)
- moveIA
 - IA, [38](#)
- NBRBOMB
 - Map.hpp, [68](#)

- NEWGAME
 - indieStudio.hpp, 66
- NOBOX
 - Cube, 22
- NORMAL
 - Cube, 22
- nearPlayer
 - IA, 38
- newStateButton
 - Param, 53
- OnEvent
 - GetEvent, 28
- PARAM
 - indieStudio.hpp, 66
- PERSO_PATH
 - Credits.hpp, 63
 - Intro.hpp, 67
- Param, 50
 - ~Param, 50
 - drawNameButton, 51
 - drawParam, 51
 - drawReturnButton, 51
 - drawSideButton, 51
 - getButton, 52
 - getClickButton, 52
 - getFormattedStringButtonName, 52
 - getParam, 52
 - initButton, 53
 - initButtonLimits, 53
 - initEquivAction, 53
 - initIntroButton, 53
 - isRun, 53
 - newStateButton, 53
 - Param, 50
 - setButtonLimits, 54
 - setEquivButton, 54
 - setIntroButton, 54
- playLongSong
 - Song, 55
- playShortSong
 - Song, 55
- RANGE
 - Cube, 22
- RIGHT
 - IA, 33
- realizeDisplacement
 - Bombberman, 11
- runCore
 - StudioCore, 58
- runCredits
 - StudioCore, 58
- runDeath
 - StudioCore, 58
- runDifferentCases
 - StudioCore, 59
- runGame
 - StudioCore, 59
- runIntro
 - StudioCore, 59
- runMenu
 - StudioCore, 59
- runMultiplayerGame
 - StudioCore, 59
- runNewGame
 - StudioCore, 59
- runParam
 - StudioCore, 60
- S_TO_WS
 - Graphical.hpp, 65
 - Map.hpp, 68
- SPEED
 - Cube, 22
- setButtonLimits
 - Param, 54
- setCamera
 - Graphical, 31
- setCubeDeleted
 - Map, 46
- setDestruct
 - Cube, 24
- setEnemy
 - Bombberman, 12
- setEquivAction
 - Menu, 49
- setEquivButton
 - Param, 54
- setGame
 - Menu, 49
- setIntroButton
 - Param, 54
- setItem
 - Bombberman, 12
- setKeyMove
 - Bombberman, 12
- setMulti
 - Bombberman, 12
- setMultiplayer
 - Menu, 49
- setPPos
 - IA, 38
- Song, 54
 - ~Song, 55
 - playLongSong, 55
 - playShortSong, 55
 - Song, 55
 - stopLongSong, 56
- Song/Song.cpp, 71
- start
 - Graphical, 32
- stopLongSong
 - Song, 56
- StudioCore, 56
 - ~StudioCore, 57
 - computeMapParam, 57

- initBomberman, [57](#)
- initCore, [58](#)
- initMultiplayerBomberman, [58](#)
- initSelectState, [58](#)
- runCore, [58](#)
- runCredits, [58](#)
- runDeath, [58](#)
- runDifferentCases, [59](#)
- runGame, [59](#)
- runIntro, [59](#)
- runMenu, [59](#)
- runMultiplayerGame, [59](#)
- runNewGame, [59](#)
- runParam, [60](#)
- StudioCore, [57](#)
- StudioCore.hpp
 - MULTI_P, [69](#)
- timeToExplode
 - Bomberman, [12](#)
 - IA, [38](#)
- TypeBox
 - Cube, [22](#)
- UP
 - IA, [33](#)
- WINDOW_HEIGHT
 - indieStudio.hpp, [66](#)
- WINDOW_WIDTH
 - indieStudio.hpp, [66](#)
- wallAround
 - IA, [38](#)