cpp_indie_studio 2018

Generated by Doxygen 1.8.11

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

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< IItem > 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 ()

  initialision of commands for player1 and player2
- void realizeDeplacement ()

  realize the player deplacement withe keyboard inputs
- void dropBomb ()

  droping 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

ia	is it ia or no
firstPlayer	is it the first player
maxBomb	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

Map	represent the Map

Returns

nothing

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

deleting bomb when she explodes

Parameters

Map	represent the Map

Returns

nothing

4.1.2.3 void Bomberman::dropBomb ( )

droping 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

Map	represent the Map
Collision	handling collision

Returns

nothing

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

get an item

Returns

std::shared_ptr<IItem> _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

smgr	managing the scene
Driver	handling vidéo
Graphical	managing the graphics of character

Returns

int

4.1.2.10 void Bomberman::initKeyMove ( )

initialision 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

Map	represent the Map
f	position first
s	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

Map	represent the Map
Collision	handling the collision

Returns

nothing

Parameters

Map	represent the Map

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

key	key nbr
x	x index
y	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

ISceneManager	∗smgr: scene manager
IVideoDriver	∗driver: driver
int	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

int	x: position on x
int	y: position on y
std::shared_ptr<Map>	Map: map
std::vector<std::pair<int,int>>	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

float	x: position on x
float	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

pos[2]	pos x and y
len[2]	len x and y
&name	name of button
value	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

mapWidth	width of the map
mapHeight	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

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

Returns

true or false

4.4.2.2 int Collision::getFarTen (int *toFind*)

round to ten

Parameters

int	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

std::shared_ptr<Graphical>	Graphical: graphical object

Returns

actualState::CREDITS

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

do the credits movement

Parameters

std::shared_ptr<Graphical>	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

std::shared_ptr<Graphical>	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 detroyed
- 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

height	height of the cube
width	width of the cube
manager	manager of the Scene
driver	vidéo handling
destructable	Cube destructable or no
TypeBox	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 detroyed

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

height	height of the floor
width	width of the floor
manager	manager of the Scene
driver	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

event	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 ()

initalisation 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

const	irr::core::vector3df &position: (x,y,z) position of the eye
const	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

IrrlcihtDevice ∗_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 ( )

initalisation 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

float	x: position on x
float	y: position on y
float	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

smgr	managing the scene
Driver	handling vidéo
Graphical	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

Map	a map that contain map
Collision	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

↩ _↩ *x*	position z
↩ _↩ *z*	position x

Returns

float

4.10.3.3 IA::IaDir 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

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

Returns

IaDir

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

check if not last

Parameters

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

Returns

IaDir

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

delete the bomb after explosion

Parameters

Map	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

Map	a map that contain map
Collision	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

Map	map that contain the map
Collision	handle the collision of character
pos	positions of IA 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

Map	map that contain the map
Collision	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

Map	map that contain the map
Collision	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

Map	map type that contain the map
f	first pos
s	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 IItem Class Reference

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

Inherited by BombStandard.

Public Member Functions

- virtual ∼IItem ()=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 IItem::∼IItem ( ) `[virtual], [default]`

4.11.2 Member Function Documentation

4.11.2.1 virtual bool IItem::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 IItem::getId ( ) `[pure virtual]`

Implemented in BombStandard.

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

Implemented in BombStandard.

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

Implemented in BombStandard.

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

Implemented in BombStandard.

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

- include/Item.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

std::shared_ptr<Graphical>	Graphical: graphical object

Returns

actualState::INTRO

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

do the introduction move

Parameters

std::shared_ptr<Graphical>	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

std::shared_ptr<Graphical>	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

manager	the manager of the scene
driver	for handling video
vParam	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

rdm	random
caseWSpeed	case box of speed
caseWSRange	case box of range
caseWBoost	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

x	the position x of cube
y	position y of the Cube
isToDestroy	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

pos	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

x	position x of the Cube
y	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 ()

  initalision 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

clickX	position X of click
clickY	position Y of click
isClick	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 ( )

initalision 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

bool	multi: multi state

Returns

nothing

Parameters

bool	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 ()

  initalize 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

std::shared_ptr<Graphical>	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

Graphical	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

clickX	position X of click
clickY	position X of click
isClick	is it clicked or no

Returns

actualState

4.15.2.7 std::string Param::getFormatedStringButtonName (Button *b1*)

get a formatted string of button name

Parameters

b1	Button 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 ( )

initalize 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

b1	Button type

Returns

void

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

set the buttons limit

Parameters

const::std::string	&nameButton: button name
int	lBottom: bottom limit
int	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

song	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

song	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 assiociated 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 assiociated 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 IItem

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