DrInSane

0.1

Fri Mar 15 2013 11:13:35

Contents

Chapter 1

Development Environment

1.1 IDE

Ecipse with the C/C++ plugin

Compiler: GCC 4.7

1.2 Version Control

git with Github as server

https://github.com/philipgraf/Dr_mad_daemon

1.3 Nightly Build

Hudson extensible continuous integration server 3.0.0

1.4 Graphics

Gimp 2.8.4 for all the graphics

1.5 External Libraries

SDL 1.6 for all the rendering and user inputs

Box2D 2.2.1 for the physics

yaml-cpp 0.5 for the configfiles

cwiid for the wiimote input

1.6 Operating system

ArchLinux

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

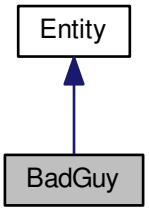
Class Documentation

4.1 BadGuy Class Reference

Badguy class.

`#include <BadGuy.h>`

Collaboration diagram for BadGuy:



Public Member Functions

- BadGuy (string type, int x, int y)

  Constructor of BadGuy.
- void move ()

  The basic BadGuy movement.
- void logic ()

  The main logic.

Additional Inherited Members

4.1.1 Detailed Description

Badguy class.

This class defines all badguy specific things

Author

Felix Eckner

Date

14.04.2013

Version

0.1.0 Alpha-State

4.1.2 Constructor & Destructor Documentation

4.1.2.1 BadGuy::BadGuy (string *type*, int *x*, int *y*)

Constructor of BadGuy.

Call the constructor of Entity. Get the information about the specific BadGuy from the bayguy.yml Build collision box with width and height from the file.

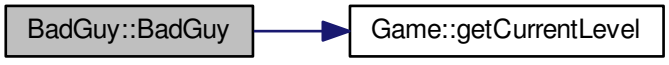
See Also

Entity()

Parameters

type	based on this the width, height, maxSpeed and items will be read form file
x	the x position of the BadGuy
y	the y position of the BadGuy

Here is the call graph for this function:



4.1.3 Member Function Documentation

4.1.3.1 void BadGuy::logic ( ) [virtual]

The main logic.

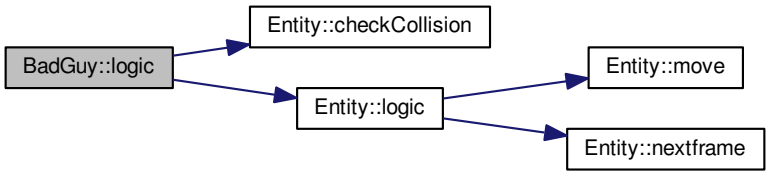
check the collisions of the BadGuy. Set alive variable to false if he got a top collision. Change the direction if collision is left of right. Call the logic from the Entity.

See Also

Entity::logic()

Reimplemented from Entity.

Here is the call graph for this function:



4.1.3.2 void BadGuy::move ( ) [virtual]

The basic BadGuy movement.

move the BadGuy if he is grounded. set the action variable corresponding to the direction

See Also

grounded
direction

Reimplemented from Entity.

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

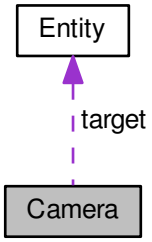
- src/BadGuy.h
- src/BadGuy.cpp

4.2 Camera Class Reference

Camera class.

`#include <Camera.h>`

Collaboration diagram for Camera:



Public Types

- enum { STICKY = 0, SMOOTH, CUSTOM }

  Cameramode enum.

Public Member Functions

- Camera (Entity *target, int w=WIDTH, int h=HEIGHT)

  Constuctor.
- Camera (float x=0, float y=0, int w=WIDTH, int h=HEIGHT)

  constuctor Set the cameraMode to CUSTOM and position and size to given values.
- void logic ()

  Main logic of the Camera If the mode is sticky the position will be recalculated from the Entity stored in the target variable Checks the x and y value of the camera and set it back to zero if it is lower then zero and set it back to the maximum width.
- void drawImage ()

  call the draw functions in the correct order.
- void move (int h, int v)

  Increase the x and y of the camera by the given values.
- void setPosition (int x, int y)

  Set the position of the camera to given x and y values.
- void setPosition (Entity *target)

  Set the camera position depends on given Target.
- mode getCameraMode ()

  Get the current Camera Mode.
- void setCameraMode (mode cameraMode)

  Set the current Camera Mode with given value.
- int getHeight () const

  Get the Height of the camera.
- void setHeight (int height)

  Set the Height to the given value.
- const Entity * getTarget () const

  Get the target the camera have focused on.
- void setTarget (Entity *target)

  Set the target of the camera.
- int getWidth () const

  get the Width of the camera
- void setWidth (int width)

  Set the width of the camera to given value.
- int getX () const

  get the x position of the camera
- int getY () const

  get the y position of the camera
- SDL_Rect getRect ()

  get the x,y position and the width and height as SDL_Rect

Public Attributes

- enum Camera:: { ... } **mode**

Private Member Functions

- void drawItems ()

  draw all items stored in the item list
- void drawNotification ()

  draw all notifications stored in the notification list
- void drawEntities ()

  draw all entities stored in the entity list and put the crosshairs on it if it is selected
- void drawTiles (int layer)

  compute the viewable range of the litelist and draw the range of given layer on the screen.
- void drawBackground ()

  draw the backgrounde image of the Level on the screen.

Private Attributes

- int x

  The absolute x value of the camera.
- int y

  The absolute y value of the camera.
- int width

  The camera width.
- int height

  The camera height.
- Entity * target

  This contain an Entity if cameraMode is STICKY and SMOOTH.
- mode cameraMode

  cameraMode of this camera
- SDL_Surface * crosshairs

  crosshairs which will be drawn over the selected Entity if the player have one

4.2.1 Detailed Description

Camera class.

This class handles all the output to the screen

Author

Philip Graf

Date

14.04.2013

Version

0.1.0 Alpha-State

4.2.2 Member Enumeration Documentation

4.2.2.1 anonymous enum

Cameramode enum.

manage the main properties of the camera

Enumerator

STICKY the camera will center the entity stored in the target variable

SMOOTH the camera will follow the entity smoothly (not implemented yet)

CUSTOM the camera will stay on the x, y position

4.2.3 Constructor & Destructor Documentation

4.2.3.1 Camera::Camera (Entity * *target*, int *w* = WIDTH, int *h* = HEIGHT)

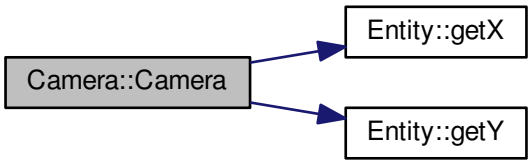
Constuctor.

Set the cameraMode to STICKY and target, width, height to the given values

Parameters

target	The entity which the camera will focus
w	the width of the camera
h	the height of the camera

Here is the call graph for this function:



4.2.3.2 Camera::Camera (float *x* = 0, float *y* = 0, int *w* = WIDTH, int *h* = HEIGHT)

constuctor Set the cameraMode to CUSTOM and position and size to given values.

Parameters

x	The x position of the camera
y	The y position of the camera
w	The width of the camera h The Height of the camera

4.2.4 Member Function Documentation

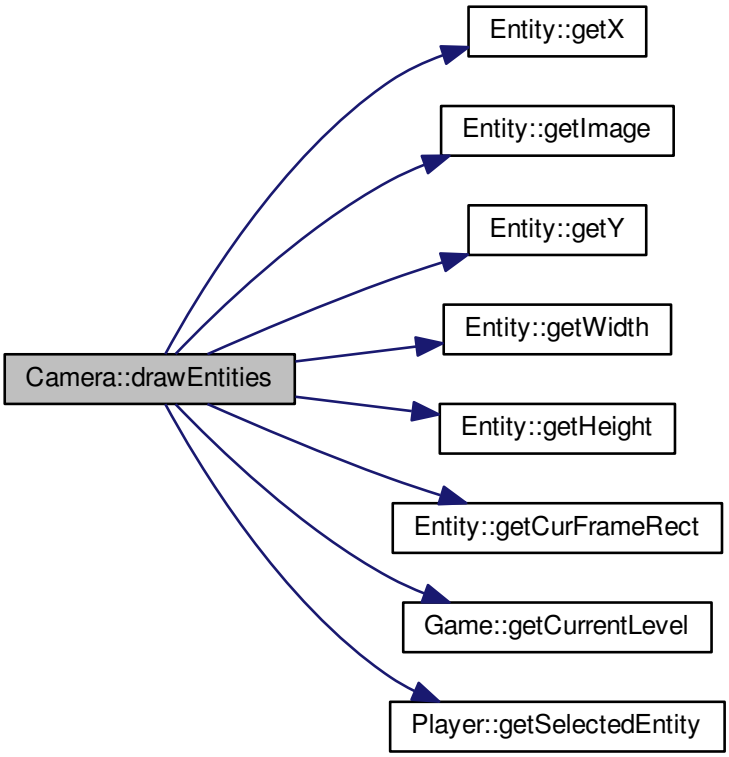
4.2.4.1 void Camera::drawEntities ( ) [private]

draw all entities stored in the entity list and put the crosshairs on it if it is selected

See Also

Entity::entityList
crosshairs

Here is the call graph for this function:



4.2.4.2 void Camera::drawImage ( )

call the draw functions in the correct order.

From background to foreground.

See Also

drawBackground()
drawTiles()
drawEntities()
Items()
drawNotification()

4.2.4.3 void Camera::drawItems ( ) [private]

draw all items stored in the item list

See Also

Item::itemList

4.2.4.4 void Camera::drawNotification ( ) [private]

draw all notifications stored in the notification list

See Also

Notification::notificationList

4.2.4.5 void Camera::drawTiles (int *layer*) [private]

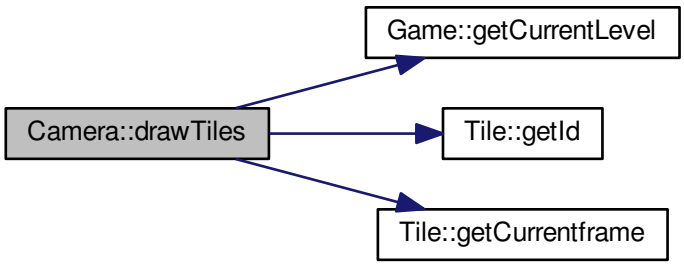
compute the viewable range of the litelist and draw the range of given layer on the screen.

(Background, Main and Foreground)

Parameters

layer	contails the layer which will be rendered

Here is the call graph for this function:



4.2.4.6 Camera::mode Camera::getCameraMode ( )

Get the current Camera Mode.

Returns

cameraMode

4.2.4.7 int Camera::getHeight ( ) const

Get the Height of the camera.

Returns

the height of the camera

4.2.4.8 SDL_Rect Camera::getRect ( )

get the x,y position and the width and height as SDL_Rect

Returns

the whole camera proportion as SDL_Rect

4.2.4.9 int Camera::getWidth ( ) const

get the Width of the camera

Returns

the current camera height

4.2.4.10 int Camera::getX ( ) const

get the x position of the camera

Returns

the current x position

4.2.4.11 int Camera::getY ( ) const

get the y position of the camera

Returns

the current y position

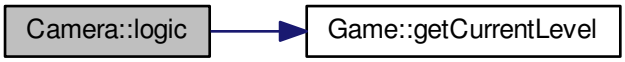
4.2.4.12 void Camera::logic ( )

Main logic of the Camera If the mode is sticky the position will be recalculated from the Entity stored in the target variable Checks the x and y value of the camera and set it back to zero if it is lower then zero and set it back to the maximum width.

See Also

mode
target

Here is the call graph for this function:



4.2.4.13 void Camera::move (int *h*, int *v*)

Increase the x and y of the camera by the given values.

Parameters

h	Horizontal movement
v	Vertical movement

4.2.4.14 void Camera::setCameraMode (mode *cameraMode*)

Set the current Camera Mode with given value.

Parameters

cameraMode	Mode that will be set

4.2.4.15 void Camera::setHeight (int *height*)

Set the Height to the given value.

Parameters

height	value which the height will be set to.

4.2.4.16 void Camera::setPosition (int *x*, int *y*)

Set the position of the camera to given x and y values.

Parameters

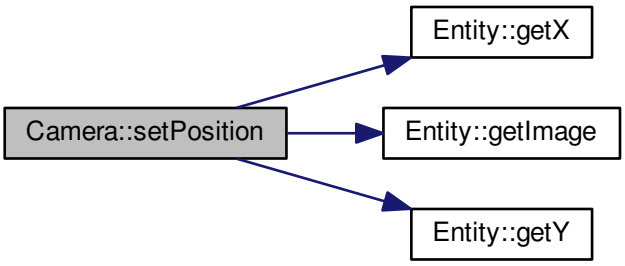
x	new x value
y	new y value

4.2.4.17 void Camera::setPosition (Entity * *target*)

Set the camera position depends on given Target.

target Entity which will be centered

Here is the call graph for this function:



4.2.4.18 void Camera::setWidth (int *width*)

Set the width of the camera to given value.

Parameters

width	the value camera will be set to.

4.2.5 Member Data Documentation

4.2.5.1 int Camera::x [private]

The absolute x value of the camera.

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

- src/Camera.h
- src/Camera.cpp

4.3 controll_t Struct Reference

Public Attributes

- int **left**
- int **right**
- int **up**
- int **down**
- int **jump**
- int **run**
- int **use**

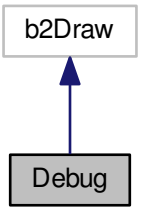
- int **grab**
- int **pda**

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

- src/Game.h

4.4 Debug Class Reference

Collaboration diagram for Debug:



Public Member Functions

- void **DrawPolygon** (const b2Vec2 *vertices, int32 vertexCount, const b2Color &color)
- void **DrawSolidPolygon** (const b2Vec2 *vertices, int32 vertexCount, const b2Color &color)
- void **DrawCircle** (const b2Vec2 &center, float32 radius, const b2Color &color)
- void **DrawSolidCircle** (const b2Vec2 &center, float32 radius, const b2Vec2 &axis, const b2Color &color)
- void **DrawSegment** (const b2Vec2 &p1, const b2Vec2 &p2, const b2Color &color)
- void **DrawTransform** (const b2Transform &xf)

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

- src/Debug.h
- src/Debug.cpp

4.5 Entity Class Reference

Entity Class.

`#include <Entity.h>`

Public Member Functions

- Entity ()

  Constuctor Set the given values an initial all other Entityvalues with default values.
- virtual ~Entity ()

  Iterate thru the items and create for each a new item with random relX and relY values.

- virtual void logic ()

  currently just call the move and nextframe function
- virtual void move ()

  virtual function movement defined from inheriting classes
- int checkCollision ()

  check the collision and set the corresponding bit of the return value and set grounded of true if the bottom sensors is touched.
- SDL_Rect getCurFrameRect ()

  Return the current animation picture of the entity.
- bool isAlive () const

  returns TRUE if the entity is alive
- void setAlive (bool alive)

  set the alive state to the given value
- float getX () const

  get the x position of the entity
- float getY () const

  Get get current y Position of the Entity.
- SDL_Surface * getImage ()

  Get the image of the Entity.
- int getCurrentframe () const

  get the current frame
- void setCurrentframe (int currentframe)

  set the current frame to given value
- float getHeight () const

  get the height of the Entity
- void setHeight (float height)

  set the height of the entity to given value
- float getWidth () const

  get the width of the Entity
- void setWidth (float width)

  set the width of the entity to given value
- b2Body * getBody ()

  get the body of the entity which contains the x and y values and all his shapes
- void setDirection (Uint8 direction)

  set bits of the direction variable
- void delDirection (Uint8 direction)

  delete bits of the direction variable
- Uint8 getDirection () const

  get the direction of the entity
- std::map< std::string, int > & getItems ()

  get the itemlist of the entity
- virtual void addItem (std::string item)

  add the given item to the itemlist if the item is already in the item list the amount of the item will be increased.

Static Public Attributes

- static std::vector< Entity * > entityList

  list of all entities

Protected Member Functions

- void nextframe ()

  vector of all bottom shapes for a smoother movement

Protected Attributes

- std::map< std::string, int > items

  a map with all items and there amount
- SDL_Surface * image

  Image of the entity with all animationframes.
- bool alive

  contains true if the entity is still alive and false if not
- Uint8 **direction**
- int currentframe

  Current Frame number this is used to calculate the correct part of the image.
- int action

  the action e.g.
- std::vector< int > actionframes

  contains the number of frames for each action
- std::vector< unsigned > animationDuration

  contains the duration of each actionframe
- float width

  width of the entity in meter
- float height

  height of the entity in meter
- float maxVelocity

  the maximal velocity of the entity.
- bool grounded

  is true when sensorBottom collide with something.
- b2Fixture * sensorRight

  right sensor used for collision detection
- b2Fixture * sensorLeft

  left sensor used for collision detection
- b2Fixture * sensorTop

  top sensor used for collision detection
- b2Fixture * sensorBottom

  bottom sensor used for collision detection
- b2Body * body

  the main body.
- std::vector< b2Fixture * > **wheels**

Private Attributes

- Uint32 timer

  time in milliseconds needed for nextframe

4.5.1 Detailed Description

Entity Class.

This class handle all things you can interact with.

Author

Felix Eckner

Date

14.04.2013

Version

0.1.0 Alpha-State

4.5.2 Constructor & Destructor Documentation

4.5.2.1 Entity::Entity ( )

Constuctor Set the given values an initial all other Entityvalues with default values.

Parameters

imagename	Name of the image which will be loaded and converted into a SDL_Surface
w	The width of the entity
h	The height of the entity
x	The x value of the entity
y	the y value of the entity

4.5.2.2 Entity::~Entity ( ) [virtual]

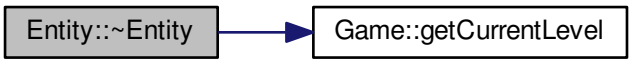
Iterate thru the items and create for each a new item with random relX and relY values.

delete the entity in the entity list and free all allocated memory.

See Also

Item
entityList

Here is the call graph for this function:



4.5.3 Member Function Documentation

4.5.3.1 void Entity::addItem (std::string *item*) [virtual]

add the given item to the itemlist if the item is already in the item list the amount of the item will be increased.

Parameters

item	the item which will be added

Reimplemented in Player.

Here is the caller graph for this function:

4.5.3.2 int Entity::checkCollision ( )

check the collision and set the corresponding bit of the return value and set grounded of true if the bottom sensors is touched.

1st Bit for top 2nd Bit for left 3rd Bit for bottom 4th Bit for right

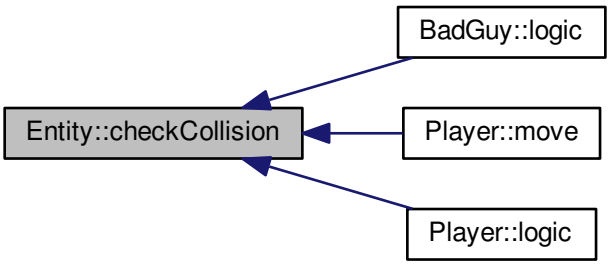
See Also

grounded

Returns

the collided sites as bits in a integer

Here is the caller graph for this function:



4.5.3.3 void Entity::delDirection (Uint8 *direction*)

delete bits of the direction variable

Parameters

direction	this contains the bits which will be deleted

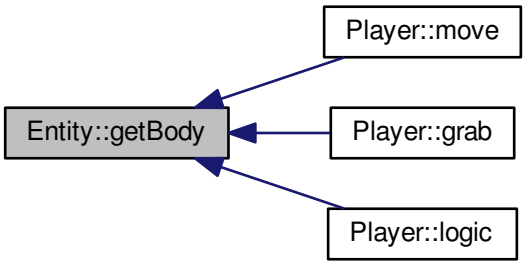
4.5.3.4 b2Body * Entity::getBody ( )

get the body of the entity which contains the x and y values and all his shapes

Returns

the body of the entity

Here is the caller graph for this function:



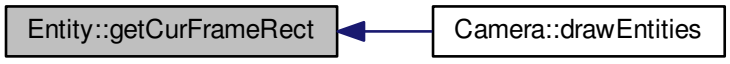
4.5.3.5 SDL_Rect Entity::getCurFrameRect ( )

Return the current animation picture of the entity.

Returns

the current animation picture as SDL_Rect

Here is the caller graph for this function:



4.5.3.6 int Entity::getCurrentframe ( ) const

get the current frame

Returns

the current frame

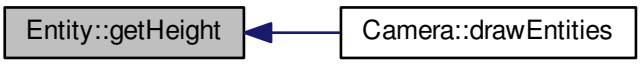
4.5.3.7 float Entity::getHeight ( ) const

get the height of the Entity

Returns

the height of the Entity

Here is the caller graph for this function:



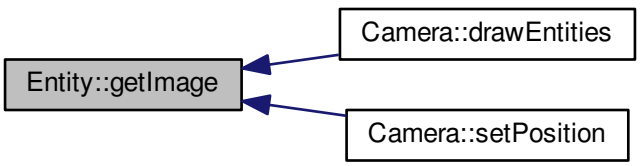
4.5.3.8 SDL_Surface * Entity::getImage ( )

Get the image of the Entity.

Returns

the image of the Entity as SDL_Surface

Here is the caller graph for this function:



4.5.3.9 std::map< std::string, int > & Entity::getItems ( )

get the itemlist of the entity

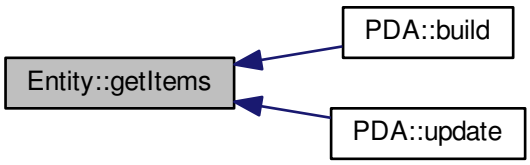
Returns

the items of the entity

See Also

items

Here is the caller graph for this function:



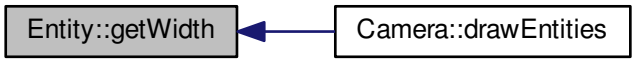
4.5.3.10 float Entity::getWidth ( ) const

get the width of the Entity

Returns

the with of the Entity

Here is the caller graph for this function:



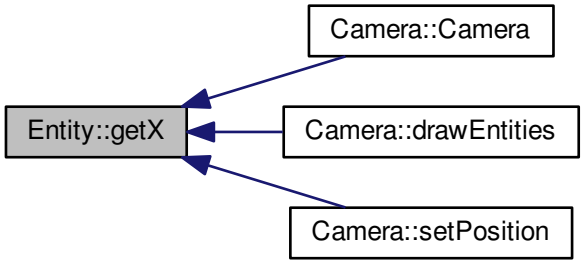
4.5.3.11 float Entity::getX ( ) const

get the x position of the entity

Returns

the x position of the entity

Here is the caller graph for this function:



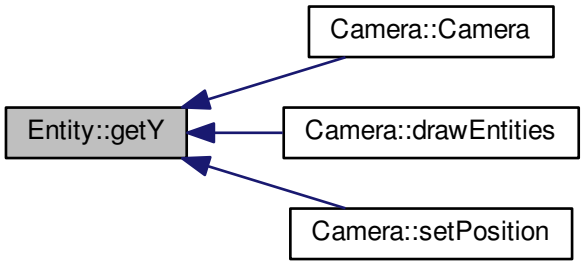
4.5.3.12 float Entity::getY ( ) const

Get get current y Position of the Entity.

Returns

the y position of the Entity

Here is the caller graph for this function:



4.5.3.13 bool Entity::isAlive ( ) const

returns TRUE if the entity is alive

Returns

the alive state of the entity

Here is the caller graph for this function:

4.5.3.14 void Entity::logic ( ) [virtual]

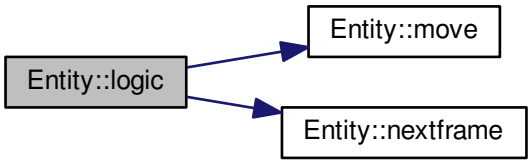
currently just call the move and nextframe function

See Also

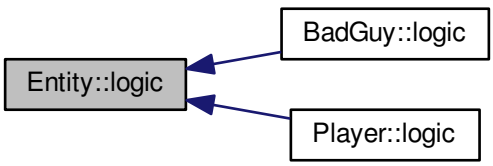
move()
nextframe()

Reimplemented in Player, and BadGuy.

Here is the call graph for this function:



Here is the caller graph for this function:



4.5.3.15 void Entity::nextframe ( ) [protected]

vector of all bottom shapes for a smoother movement

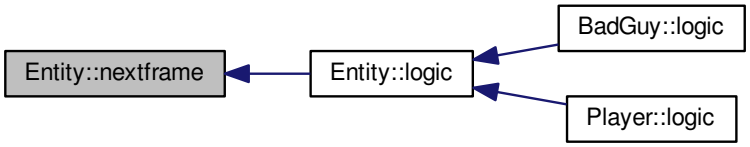
Changes the frame of a entity.

If the duration the current frame should be displayed is over it will be set to the next or the first (if the current frame is the last)

See Also

actionframes
animationDuration

Here is the caller graph for this function:



4.5.3.16 void Entity::setAlive (bool *alive*)

set the alive state to the given value

Parameters

alive	new alive state of the entity

4.5.3.17 void Entity::setCurrentframe (int *currentframe*)

set the current frame to given value

Parameters

currentframe	the value current frame will be set to

4.5.3.18 void Entity::setDirection (Uint8 *direction*)

set bits of the direction variable

Parameters

direction	this contains the bits which will be set

4.5.3.19 void Entity::setHeight (float *height*)

set the height of the entity to given value

Parameters

height	the new height of the entity

4.5.3.20 void Entity::setWidth (float *width*)

set the width of the entity to given value

Parameters

width	the new width of the entity

4.5.4 Member Data Documentation

4.5.4.1 int Entity::action [protected]

the action e.g.

move left, move right, jump, needed for rendering

4.5.4.2 b2Body* Entity::body [protected]

the main body.

contains the x and y values and to this body all forces (gravity, ...) are applied

4.5.4.3 bool Entity::grounded [protected]

is true when sensorBottom collide with something.

4.5.4.4 float Entity::maxVelocity [protected]

the maximal velocity of the entity.

May be manipulated for running, etc.

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

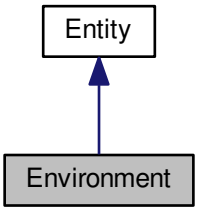
- src/Entity.h
- src/Entity.cpp

4.6 Environment Class Reference

Environment Class.

`#include <Environment.h>`

Collaboration diagram for Environment:



Public Member Functions

- Environment (std::string type, int x, int y)

  Constructor of Environment Call the constructor of Entity.

Additional Inherited Members

4.6.1 Detailed Description

Environment Class.

This class defines all Environment specific things

Author

Felix Eckner

Date

14.04.2013

Version

0.1.0 Alpha-State

4.6.2 Constructor & Destructor Documentation

4.6.2.1 Environment::Environment (std::string *type*, int *x*, int *y*)

Constructor of Environment Call the constructor of Entity.

set alive to false and all other values from the environment.yml file. Build the collision box and place it on given x and y values.

Parameters

type	is the type of the environment object
x	the x position of the environment
y	the y position of the environment

Here is the call graph for this function:

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

- src/Environment.h
- src/Environment.cpp

4.7 Event Class Reference

Event Class.

`#include <Event.h>`

Public Member Functions

- void initWiimote ()
- virtual void onEvent (SDL_Event *event)

  Function to execute the different Eventhandler.
- virtual void **onInputFocus** ()
- virtual void **onInputBlur** ()
- virtual void **onKeyDown** (SDLKey sym, SDLMod mod, Uint16 unicode)
- virtual void **onKeyUP** (SDLKey sym, SDLMod mod, Uint16 unicode)
- virtual void **onMouseFocus** ()
- virtual void **onMouseBlur** ()
- virtual void **onMouseMove** (int mX, int mY, int xRel, int yRel, bool left, bool right, bool middle)
- virtual void **onMouseWheel** (bool up, bool down)
- virtual void **onLButtonDown** (int mX, int mY)
- virtual void **onLButtonUp** (int mX, int mY)
- virtual void **onRButtonDown** (int mX, int mY)
- virtual void **onRButtonUp** (int mX, int mY)
- virtual void **onMButtonDown** (int mX, int mY)
- virtual void **onMButtonUp** (int mX, int mY)
- virtual void **onJoyAxis** (Uint8 which, Uint8 axis, Sint16 value)
- virtual void **onJoyButtonDown** (Uint8 which, Uint8 button)
- virtual void **onJoyButtonUp** (Uint8 which, Uint8 button)
- virtual void **onJoyHat** (Uint8 which, Uint8 value)
- virtual void **onJoyBall** (Uint8 which, Uint8 ball, Sint16 xRel, Sint16 yRel)
- virtual void **onMinimize** ()
- virtual void **onRestore** ()
- virtual void **onResize** (int w, int h)
- virtual void **onExpose** ()
- virtual void **onExit** ()
- virtual void **onWiiButtonEvent** (int buttons)
- virtual void **onUser** (Uint8 type, int code, void *data1, void *data2)

Private Attributes

- bdaddr_t blueaddr

  contains the bluetooth address of the wiimote
- cwiid_wiimote_t * wiimote

  contains the state of the wiimote

4.7.1 Detailed Description

Event Class.

This Class handles all Input Events

Author

Philip Graf

Date

14.04.2013

Version

0.1.0 Alpha-State

4.7.2 Member Function Documentation

4.7.2.1 void Event::initWiimote ( )

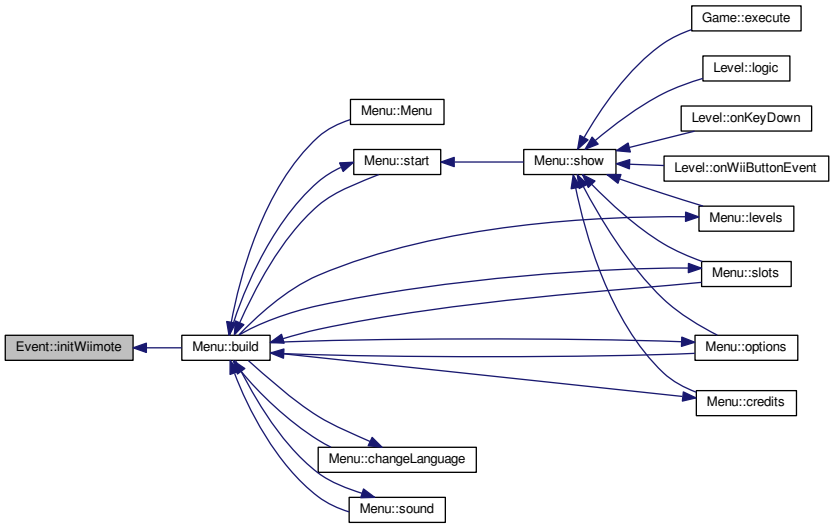
Set the callback function for the wiimote

Parameters

wiimote	wiimote struct to identifiere whicht wiimote
wiimote_callback	pointer to the callbackfunction

Set the Reportmodes Available Modes: CWIID_RPT_STATUS <- I think battery and extensions and so on CWIID_RPT_BTN <- enable buttons CWIID_RPT_ACC <- enable acceleration CWIID_RPT_IR <- enable infrared CWIID_RPT_NUNCHUK <- enable nunchuk CWIID_RPT_CLASSIC <- enable nintendo classic controller

Here is the caller graph for this function:



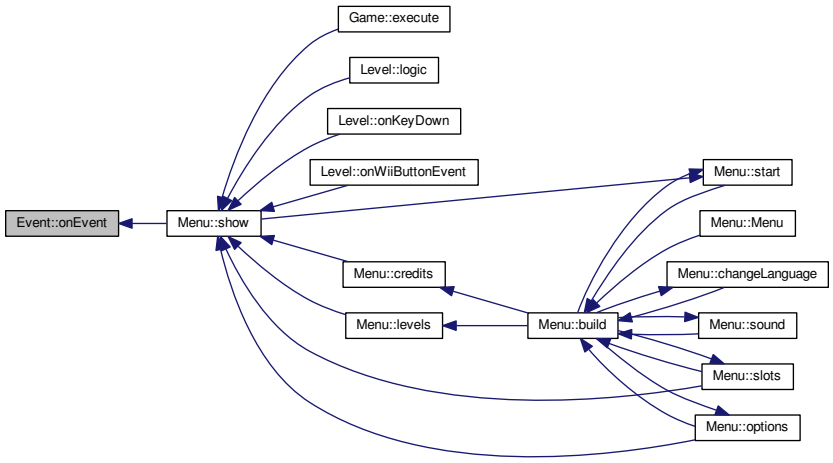
4.7.2.2 void Event::onEvent (SDL_Event * *event*) [virtual]

Function to execute the different Eventhandler.

Parameters

event	SDL_Event which contains happened UI interactions

Here is the caller graph for this function:



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

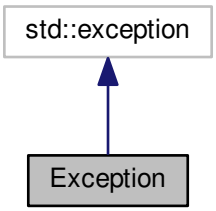
- src/Event.h
- src/Event.cpp

4.8 Exception Class Reference

Exception Class.

`#include <Exception.h>`

Collaboration diagram for Exception:



Public Types

- enum **ExceptionType** { **UNKNOWN_EXCEPTION** = 0, **UNKNOWN_ERRORSTR** }

4.8.1 Detailed Description

Exception Class.

This Class will handle all exceptions soon

Author

Philip Graf

Date

14.04.2013

Version

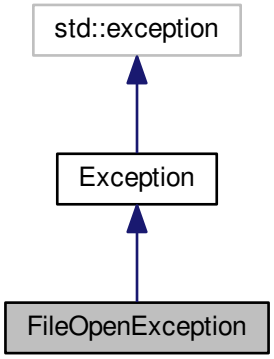
0.1.0 Alpha-State

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

- src/Exception.h
- src/Exception.cpp

4.9 FileOpenException Class Reference

Collaboration diagram for FileOpenException:



Additional Inherited Members

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

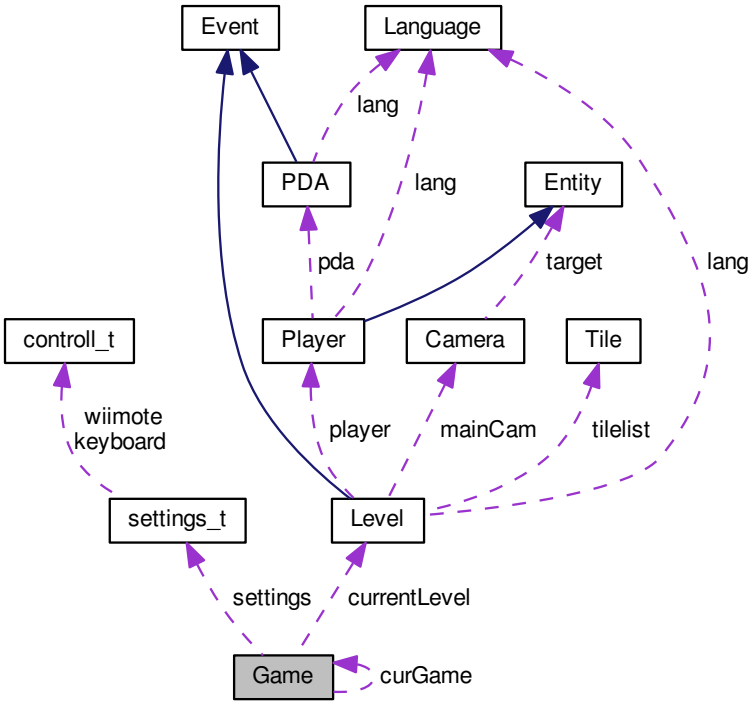
- src/Exception.h

4.10 Game Class Reference

Game class.

`#include <Game.h>`

Collaboration diagram for Game:



Public Member Functions

- Game ()

  Constructor of the Game.
- virtual ~Game ()

  Destructor of the Game.
- int execute ()

  Starts the Game.
- void init ()

  This will initialize many things Here is where all the loading methods are called, fonts get defined.
- void saveSettings ()

  Save all settings back to YAML-file.
- Level * getCurrentLevel ()

  Returns a pointer to the current Level.
- void setCurrentLevel (Level *curLev)

  Sets a Pointer to the current Level.
- TTF_Font * getFont (int which=1)

  Returns a pointer to a Font stored in the font array.

Public Attributes

- settings_t settings

  A struct with gamesettings.

Static Public Attributes

- static Game * curGame

  Pointer to running Game.
- static map< std::string, Mix_Chunk * > sounds

  map with all needed Sound effects

Private Member Functions

- void loadSettings ()

  Load the gamesettings from YAML-file.
- void loadSounds ()

  Load all soundfiles.

Private Attributes

- SDL_Surface * display

  the whole screen of the game
- TTF_Font * font [6]

  An Array with different fonts.
- Level * currentLevel

  Pointer to the current level.

4.10.1 Detailed Description

Game class.

This class handles the Game initialization and settings

Author

Philip Graf

Date

14.03.2013

Version

0.1.0 Alpha-State

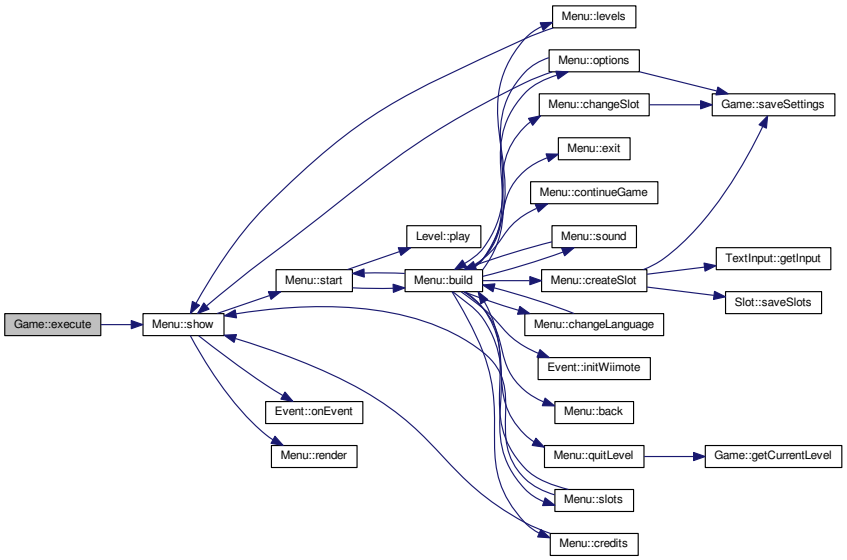
4.10.2 Member Function Documentation

4.10.2.1 int Game::execute ( )

Starts the Game.

This method calls the Game::init() to initialize the game and generates the mainmenu If mainmenu is left it will also show a goodbye message

Here is the call graph for this function:



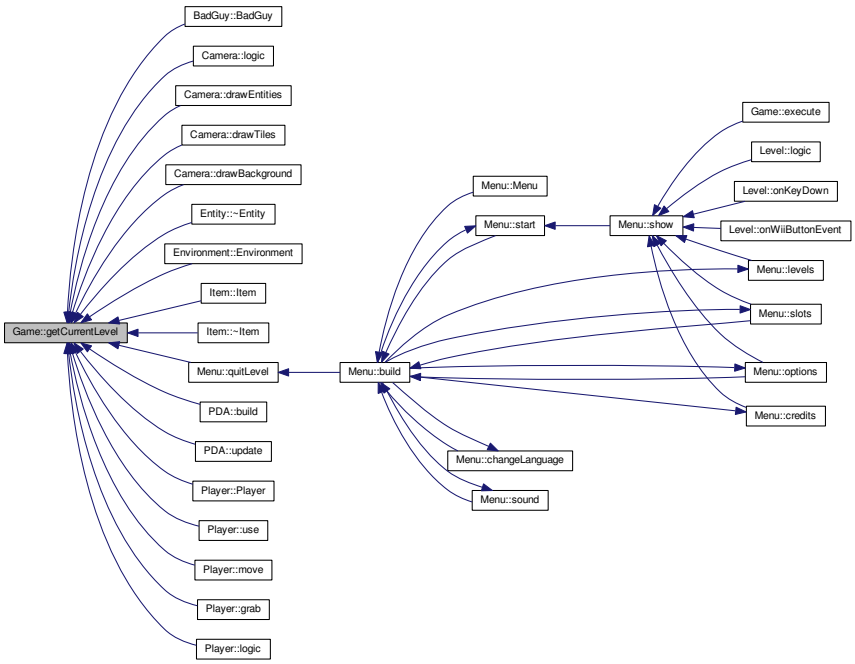
4.10.2.2 Level * Game::getCurrentLevel ( )

Returns a pointer to the current Level.

Returns

current Level

Here is the caller graph for this function:



4.10.2.3 TTF_Font * Game::getFont (int *which* = 1)

Returns a pointer to a Font stored in the font array.

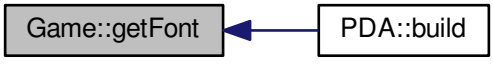
Parameters

which	int value for selecting a font

Returns

font

Here is the caller graph for this function:



4.10.2.4 void Game::init ( )

This will initialize many things Here is where all the loading methods are called, fonts get defined.

Also sound and video (screen) get configured Initialize SDL

Create Game Window with defined width and height and Bits per pixel use by the system

Set Title

Initialize SDL_ttf for Fonts

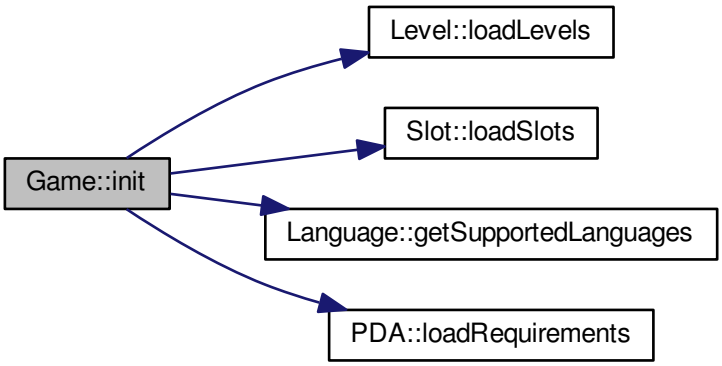
Load the menu Header font

Load settings from game.yml file

Initialize SDL_mixer for audio

load all the sound files

Here is the call graph for this function:



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

- src/Game.h
- src/Game.cpp

4.11 Item Class Reference

Item class.

`#include <Item.h>`

Public Member Functions

- Item (std::string name, int x, int y, int relX=0, int relY=0)

  Load the image.
- virtual ~Item ()

  delete the item form the itemList and free all allocated memory
- void logic ()

  enable collision with the player after 1000 milliseconds and check the collision.
- SDL_Surface * getImage ()

  get the image of the Item

- SDL_Rect getClipRect ()

  get the x,y and the width and height of the item and build it as SDL_Rect.

Static Public Attributes

- static std::vector< Item * > itemlist

  contains all items

Private Attributes

- SDL_Surface * image

  the image of the item
- std::string type

  the type of the item
- b2Body * body

  the body for collision detection
- Uint32 timer

  time in milliseconds the item does not collide with the player until this is grater then 1000

4.11.1 Detailed Description

Item class.

This class handles the Items.

Author

Felix Eckner

Date

14.04.2013

Version

0.1.0 Alpha-State

4.11.2 Constructor & Destructor Documentation

4.11.2.1 Item::Item (std::string *name*, int *x*, int *y*, int *relX* = 0, int *relY* = 0)

Load the image.

also a colorkey is set to make a defined color(0xFF00FF) transparent as .bmp do not provide an alpha channel but give the best performance build the collision box of the item. push the item in the itemList.

Parameters

name	the type of the item
x	the x position of the item
y	the y position of the item
relX	the x value of the throw vector
relY	the y value of the throw vector

Here is the call graph for this function:

4.11.3 Member Function Documentation

4.11.3.1 SDL_Rect Item::getClipRect ( )

get the x,y and the width and height of the item and build it as SDL_Rect.

Returns

the proportion of the item

4.11.3.2 SDL_Surface * Item::getImage ( )

get the image of the Item

Returns

the image of the Item

4.11.3.3 void Item::logic ( )

enable collision with the player after 1000 milliseconds and check the collision.

If a collision is detected the item will be added to the colliding entity.

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

- src/Item.h
- src/Item.cpp

4.12 items_t Struct Reference

Public Attributes

- SDL_Surface * **itemname**
- SDL_Surface * **amound**

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

- src/PDA.h

4.13 Language Class Reference

Language class.

`#include <Language.h>`

Public Member Functions

- Language ()
  Load the language configuration file.
- std::string operator[] (std::string key)
  get the translation of the given string.

Static Public Member Functions

- static void getSupportedLanguages ()
  get all languages from the greeting entry and store it in supLanguages

Static Public Attributes

- static std::vector< std::string > supLanguages
  a list with all supported Languages

Private Attributes

- YAML::Node root
  the root node of the lang.yml file

4.13.1 Detailed Description

Language class.

This class handels all translations.

Author

Philip Graf

Date

14.03.2013

Version

0.1.0 Alpha-State

4.13.2 Member Function Documentation

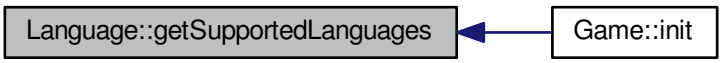
4.13.2.1 void Language::getSupportedLanguages ( ) [static]

get all languages from the greeting entry and store it in supLanguages

See Also

supLanguages

Here is the caller graph for this function:



4.13.2.2 std::string Language::operator[] (std::string *key*)

get the translation of the given string.

try to return the language which is stores in the game settings if this translation is not available the engish translation will be returned.

Parameters

key	the key in the root note

See Also

root

Returns

the translated string

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

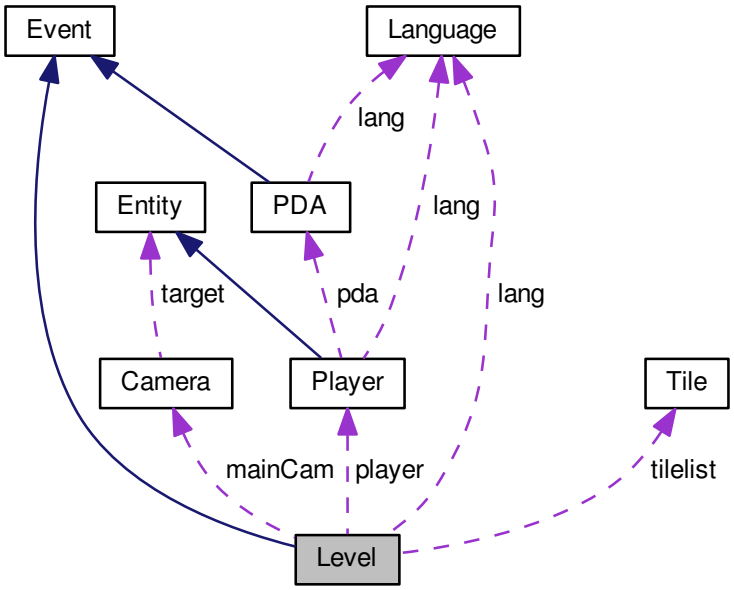
- src/Language.h
- src/Language.cpp

4.14 Level Class Reference

Level class.

`#include <Level.h>`

Collaboration diagram for Level:



Public Member Functions

- Level (unsigned levelnum)

  Initialize all necessary information from the levelconfig create all items, badguys and environments, also create the player and the main camera.
- ~Level ()

  free all allocated memory
- void render ()

  Render function from level.
- void logic ()

  Logic function of the level.
- void play ()

  Play function of the level.
- void switchActions ()

  Call the right level-Switch logic call the level logic depend on the levelnumber.
- int getGravity () const

  Get the gravity of the level.
- void setGravity (int gravity)

  set the gravity to given value.
- const string & **getName** () const
- int **getTime** () const
- void **setTime** (int time)
- int **getTileID** (int x, int y, int layer=1)
- Player * **getPlayer** ()
- int **getHeight** () const
- int **getWidth** () const

- Tile **** **getTilelist** () const
- SDL_Surface * **getBackground** () const
- b2World * **getWorld** () const
- void **setRunning** (bool running)
- bool **isFinished** () const
- void **setFinished** (bool finished)
- void **toggleSwitch** (Uint8 flags)
- Uint8 **getSwitches** () const

Static Public Member Functions

- static void loadLevels ()

  load the levelnames from the levelconfigs

Static Public Attributes

- static vector< string > levels

  list of all level in the right order
- static map< string, string > levelnames

  a map with the level and the levelname, needed for the levelmenu

Private Member Functions

- void loadMapFile (string filename)

  load the map file and build the tilelist array.
- void onKeyUP (SDLKey sym, SDLMod mod, Uint16 unicode)

  handle the key up events
- void onKeyDown (SDLKey sym, SDLMod mod, Uint16 unicode)

  handle the key down events
- void onWiiButtonEvent (int button)

  handle the wii button events
- void updateTime ()

  if 1000 milliseconds are gone the time variable will be decreased by one.
- void level0Logic ()

  Switch logic of level0.
- void level1Logic ()

  Switch logic of level1.
- void level2Logic ()

  Switch logic of level2.
- void **level3Logic** ()

Private Attributes

- Tile **** tilelist

  3D Tilearray [0=bg,1=main,2=fg][x][y]
- int levelnum

  the level number
- string name

  the name of the Level.
- SDL_Surface * bgImage

the background image of the Level

- Mix_Music * bgMusic

  the background music of the Level

- b2World * world

  the collision world where all the basic collision detection happens

- Player * player

  an object of the player

- int width

  the width of the level in meter

- int height

  the height of the level in meter

- int time

  the time in seconds the player have to complete the level

- b2Vec2 * gravity2d

  the gravity vector

- Camera * mainCam

  the main camera of the level

- Uint8 switches

  8 switches of the level.

- bool running

  while this is true the level will be rendered

- bool levelFinished

  contains true if the level is finished and false if the player is dead

- Language lang

  this objekt is used for translation the notifications

- Uint32 timer

  contains the time in milliseconds which is needed for the time

4.14.1 Detailed Description

Level class.

This class contain all Tiles, the name, the Background Image, the Player, absolute width and height in Tiles, the gravity and the time.

Author

Felix Eckner

Date

14.04.2013

Version

0.1.0 Alpha-State

4.14.2 Constructor & Destructor Documentation

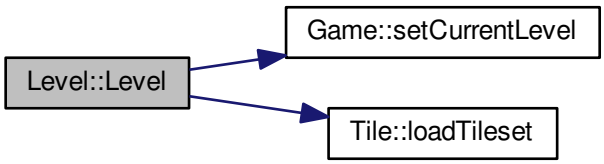
4.14.2.1 Level::Level (unsigned *levelnum*)

Initialize all necessary information from the levelconfig create all items, badguys and environments, also create the player and the main camera.

Parameters

levelnum	the number of the level this is needed to load the correct mapfile.

Here is the call graph for this function:



4.14.3 Member Function Documentation

4.14.3.1 int Level::getGravity ( ) const

Get the gravity of the level.

Returns

the y value of the gravity vector

4.14.3.2 void Level::level0Logic ( ) [private]

Switch logic of level0.

switch1 (paystation) if slot is in the inventory, turn lights to green and and switch the flags of the door to finish

4.14.3.3 void Level::level1Logic ( ) [private]

Switch logic of level1.

switch1 (hand) open the door and change the hand to green.

4.14.3.4 void Level::level2Logic ( ) [private]

Switch logic of level2.

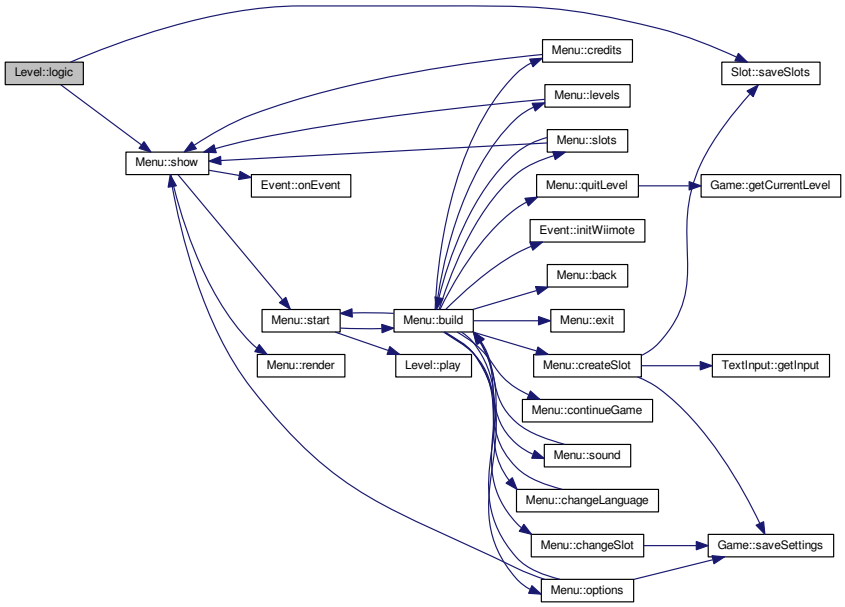
switch1 (main switch) ibuild run away and door is open

4.14.3.5 void Level::logic ( )

Logic function of the level.

update the time, call all tile, entity, items, notification and camera logic. also check if the player is alive and if the level is finished.

Here is the call graph for this function:



4.14.3.6 void Level::onKeyDown (SDLKey *sym*, SDLMod *mod*, Uint16 *unicode*) [private], [virtual]

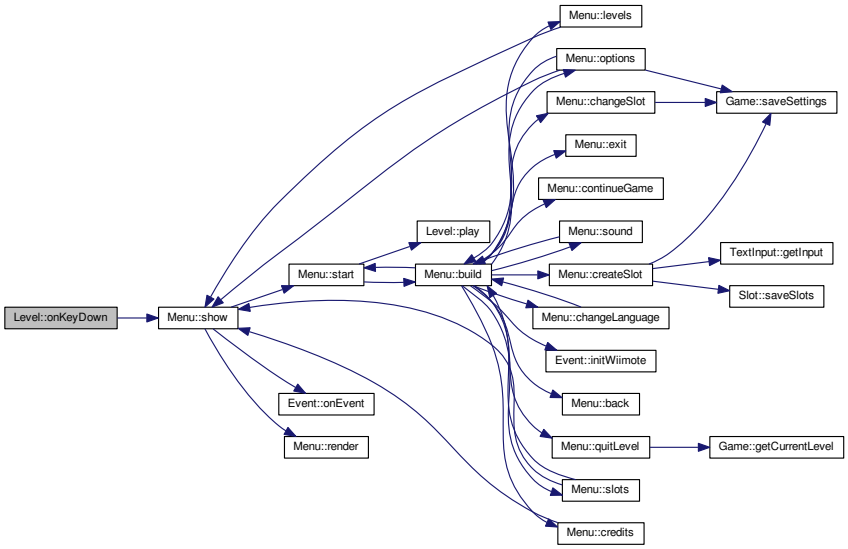
handle the key down events

Parameters

sym	the symbol of the key
mod	modifications like left-Shift or alt button
unicode	the unicode value of the pressed key

Reimplemented from Event.

Here is the call graph for this function:



4.14.3.7 void Level::onKeyUP (SDLKey *sym*, SDLMod *mod*, Uint16 *unicode*) [private], [virtual]

handle the key up events

Parameters

sym	the symbol of the key
mod	modifications like left-Shift or alt button
unicode	the unicode value of the pressed key

Reimplemented from Event.

4.14.3.8 void Level::play ( )

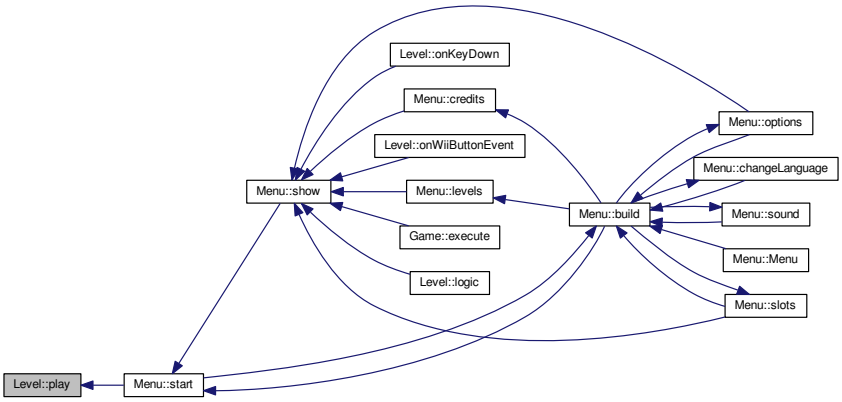
Play function of the level.

the main loop which runs until running is false.

See Also

running

Here is the caller graph for this function:



4.14.3.9 void Level::render ( )

Render function from level.

tell the main camera to render the level and flip the surface

4.14.3.10 void Level::setGravity (int *gravity*)

set the gravity to given value.

Parameters

gravity	the new gravity value

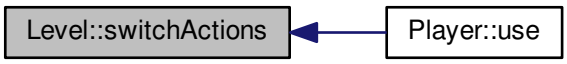
4.14.3.11 void Level::switchActions ( )

Call the right level-Switch logic call the level logic depend on the levelnumber.

See Also

level0logic()
level1logic()
level2logic()
level3logic()
levelnum()

Here is the caller graph for this function:



4.14.4 Member Data Documentation

4.14.4.1 string Level::name [private]

the name of the Level.

4.14.4.2 Uint8 Level::switches [private]

8 switches of the level.

All switches must be set to 1 to complete the level

4.14.4.3 Tile**** Level::tilelist [private]

3D Tilearray [0=bg,1=main,2=fg][x][y]

3 dimensional array with all tiles

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

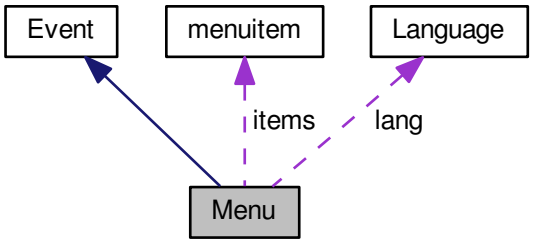
- src/Level.h
- src/Level.cpp

4.15 Menu Class Reference

Menu class.

`#include <Menu.h>`

Collaboration diagram for Menu:



Public Member Functions

- Menu (int menuType=0)
  set the type of the menu and call the build function
- virtual ~Menu ()
  free all the allocated memory
- int show ()
  the menu loop.

Private Types

- typedef void(Menu::* **fptr**)()

Private Member Functions

- void onExit ()
  the event which will be called if the close button is clicked.
- void onKeyDown (SDLKey sym, SDLMod mod, Uint16 unicode)
  handle the key events
- void onMouseMove (int mX, int mY, int xRel, int yRel, bool left, bool right, bool middle)
  mouse movement handler set the currentitem to the item under the mouse.
- void onLButtonDown (int mX, int mY)
  mouse button handler execute the labelaction of the current selected menuitem
- void onWiiButtonEvent (int buttons)
  handle the wii button events
- void render ()
  render the background and the menu label on the screen
- void select (int direction)
  play the sound and change the color and the current selected index
- void build ()
  build all the labels based on the menu type
- void start ()
  start the selected level
- void levels ()

create a new menu with the menu type LEVELMENU

- void slots ()

  create a new menu with the menu type SLOTMENU
- void exit ()

  set running to false and quit the whole game
- void options ()

  create a new menu with the menu type OPTIONSMENU
- void quitLevel ()

  quit the current level
- void back ()

  get back to the menu before
- void credits ()

  create a new menu with the menu type CREDITS
- void sound ()

  change the volume in 25 per cent steps.
- void controllerSettings ()

  not implemented yet.
- void continueGame ()

  set running of the PAUSEMENU to false.
- void changeLanguage ()

  change the language and rebuild the menu with the new language
- void changeSlot ()

  change the active slot to the currently selected one.
- void createSlot ()

  create a textinput field and create an new slot with the return name.

Private Attributes

- std::vector< fptr > labelactions

  list of all labelactions
- std::vector< std::string > labeltexts

  list of all labeltexts
- std::vector< int > labelfonts

  list of the font sizes
- SDL_Surface * background

  the background images
- SDL_Surface * backgroudFilter

  a alpha screen for the background
- menuitem * items

  list of all items
- unsigned int currentItem

  the index of the current selected menuitem
- SDL_Color colors [2]

  2 colors first for unselected and the second for selected items
- int menuType

  the menutype this is used to built the lists
- int returnValue

  this value will be returned if the menu will be destroyed
- Language lang

  this objekt is used for translation
- bool running

  the menu is visible while this is True

4.15.1 Detailed Description

Menu class.

Creates the menus and execute the different actions

Author

Philip Graf

Date

14.03.2013

Version

0.1.0 Alpha-State

4.15.2 Constructor & Destructor Documentation

4.15.2.1 Menu::Menu (int *menuType* = 0)

set the type of the menu and call the build function

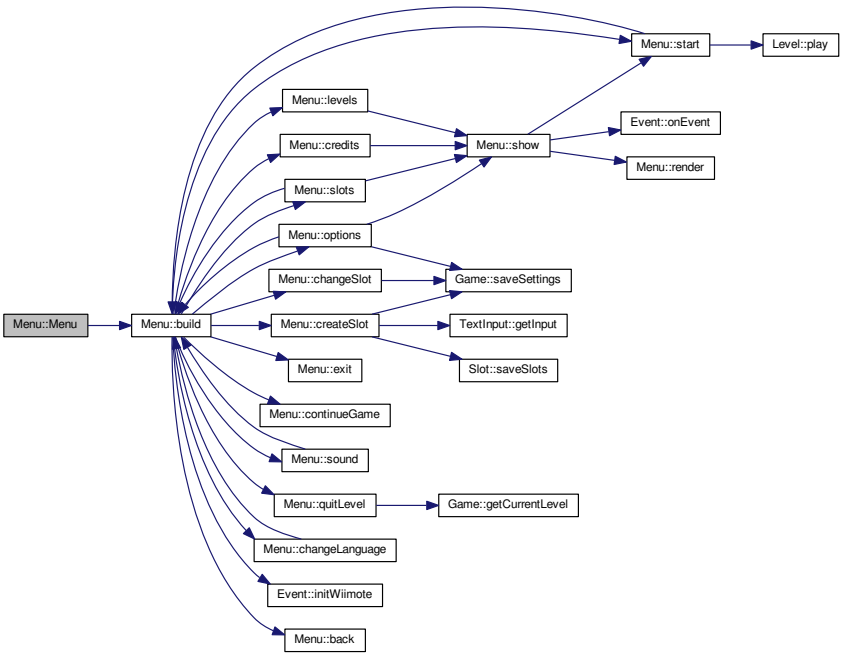
Parameters

menuType	the menu type

See Also

build()

Here is the call graph for this function:



4.15.3 Member Function Documentation

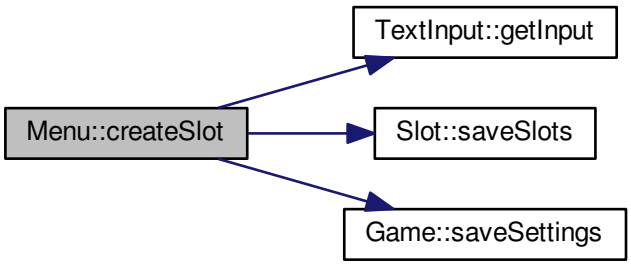
4.15.3.1 void Menu::createSlot ( ) [private]

create a textinput field and create an new slot with the return name.

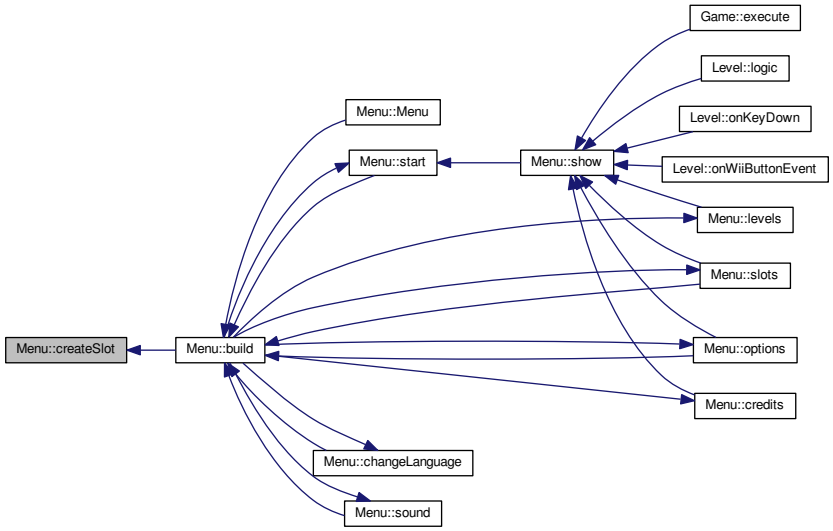
See Also

TextInput

Here is the call graph for this function:



Here is the caller graph for this function:



4.15.3.2 void Menu::onExit ( ) [private], [virtual]

the event which will be called if the close button is clicked.

quit the game.

Reimplemented from Event.

Here is the call graph for this function:

4.15.3.3 void Menu::onKeyDown (SDLKey *sym*, SDLMod *mod*, Uint16 *unicode*) [private], [virtual]

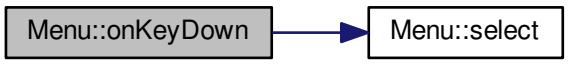
handle the key events

Parameters

sym	the symbol of the key
mod	modifications like left-Shift or alt button
unicode	the unicode value of the pressed key

Reimplemented from Event.

Here is the call graph for this function:



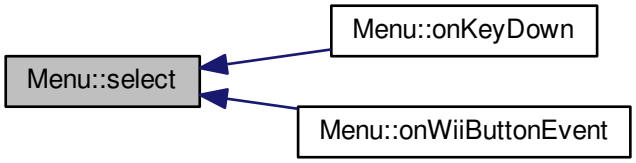
4.15.3.4 void Menu::select (int *direction*) [private]

play the sound and change the color and the current selected index

Parameters

direction	the direction that the currentItem index will change (UP or DOWN)

Here is the caller graph for this function:

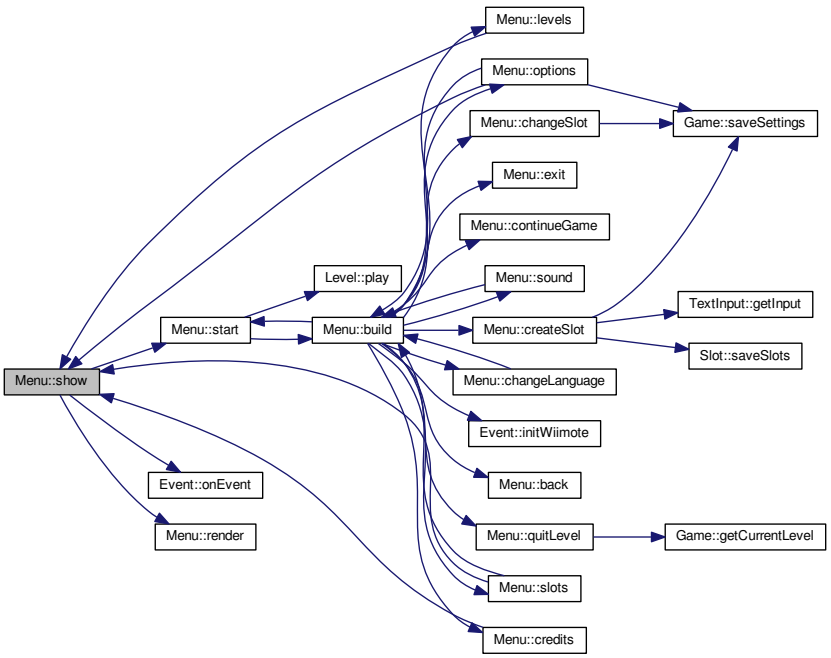


4.15.3.5 int Menu::show ( )

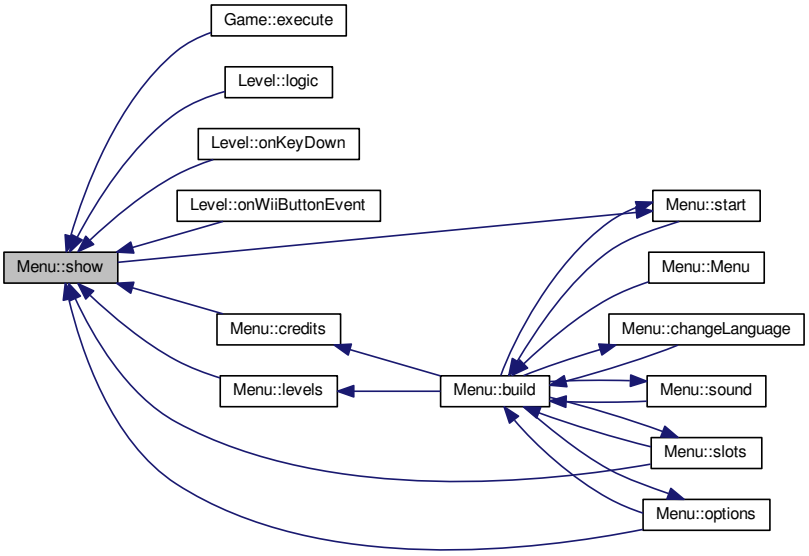
the menu loop.

Handles the Framecontrol and the user inputs. This loop runs until running is false.

Here is the call graph for this function:



Here is the caller graph for this function:

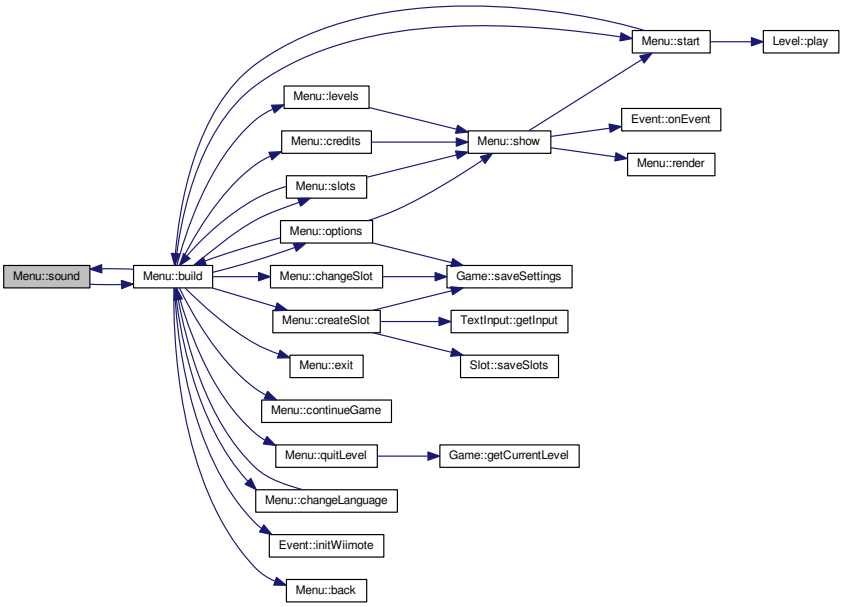


4.15.3.6 void Menu::sound ( ) [private]

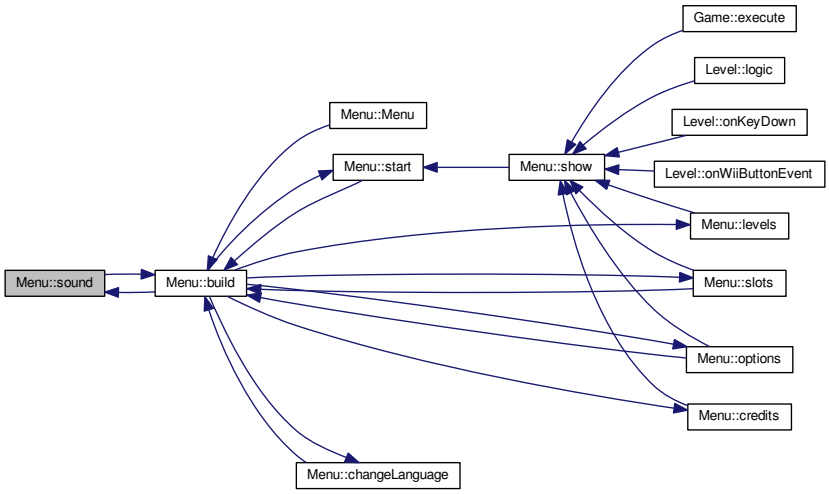
change the volume in 25 per cent steps.

the volume of the music always have half of the sounds

Here is the call graph for this function:



Here is the caller graph for this function:



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

- src/Menu.h
- src/Menu.cpp

4.16 menuitem Struct Reference

Public Attributes

- SDL_Surface * **labelSurface**
- SDL_Rect **position**

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

- src/Menu.h

4.17 Notification Class Reference

Notification class.

`#include <Notification.h>`

Public Member Functions

- Notification (std::string message, int displaySecs, int type=NOTIFICATION_INFO, std::string iconName="")

  load the images and texts convert it to SDL_Surfaces and blit it on the notificationSurface.
- virtual ~Notification ()

  free all the allocated memory and remove it self from the notification list.
- void timeout ()

  decrease the counter and commit suicide if it zero
- SDL_Surface * getNotificationSurface ()

  get the surface of the notification

Static Public Attributes

- static std::vector
  < Notification * > notificationList

  list of all notifications

Private Attributes

- SDL_Surface * notificationSurface

  the surface where the image and text will be drawn on
- int counter

  visible counter if this is zero the notification will be destroyed

4.17.1 Detailed Description

Notification class.

This class handles all notifications on the screen.

Author

Philip Graf

Date

14.03.2013

Version

0.1.0 Alpha-State

4.17.2 Constructor & Destructor Documentation

4.17.2.1 Notification::Notification (std::string *message*, int *displaySecs*, int *type* = NOTIFICATION_INFO, std::string *iconName* = " ")

load the images and texts convert it to SDL_Surfaces and blit it on the notificationSurface.

Parameters

message	the message which will be shown
displaySecs	the duration in seconds until the notification will be destroyed.
type	the type of the Notification(INFO; WARRNING)
iconName	the iconname without .bmp

4.17.3 Member Function Documentation

4.17.3.1 SDL_Surface * Notification::getNotificationSurface ( )

get the surface of the notification

Returns

the surface of the notification

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

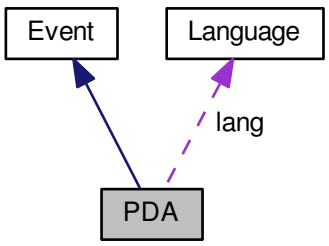
- src/Notification.h
- src/Notification.cpp

4.18 PDA Class Reference

PDA class.

`#include <PDA.h>`

Collaboration diagram for PDA:



Public Member Functions

- PDA (int level=0)

  initialize level with given value and all the rest with default values and call the init function
- int show ()

  the main loop this will run until running is FALSE first it calls build and every frame render will be called.
- int getLevel () const

  get the current level of the PDA

Static Public Member Functions

- static void loadRequirements ()

  build the vector with all the update requirements.

Static Public Attributes

- static std::vector< std::map < std::string, int > > updateReqList

  list of all level requirements

Private Member Functions

- void init ()

  creates the level specific images and compute his positions.
- void render ()

  render all the surfaces on the screen.
- void build ()

  build the itemlist and the current remaining level time
- void update ()

  check for all requirements and if all in players itemlist increase the level of the level and finally call the init and build function
- void onKeyDown (SDLKey sym, SDLMod mod, Uint16 unicode)

  the Key-Down Event handler
- void onWiiButtonEvent (int button)

  the wiimote button event

Private Attributes

- SDL_Surface * image

  the whole image of the PDA
- SDL_Surface * display

  the main display of the PDA.
- SDL_Surface * lcd

  the lcd of the PDA only available with level two or more
- SDL_Surface * cursor

  the cursor of the PDA this is used to show the currentItem
- SDL_Surface * timer

  the times left before the player dies
- SDL_Surface * updateText

  the updatetext at the bottom of the PDA
- SDL_Surface * updateRequirements

  shows the requirements for the next level
- SDL_Color green

  the color of the fonts
- SDL_Rect displayRect

  the positions and the metrics of the display
- SDL_Rect lcdRect

  the positions and the metrics of the lcd
- SDL_Rect imageRect

  the positions and the metrics of the whole image
- SDL_Rect cursorRect

  the positions and the metrics of the cursor
- SDL_Rect timerRect

  the positions and the metrics of the timer
- SDL_Rect updateTextRect

  the positions and the metrics of the display
- SDL_Rect updateRequirementsRect

  the positions and the metrics of the display
- Language lang

  this objekt is used for translation
- int level

the level of the PDA

- unsigned currentItem

  the index of the current selected item

- std::vector< items_t > itemlist

  contains all items of the player

- bool running

  while this variable is true the PDA will be visible

4.18.1 Detailed Description

PDA class.

This class handles the Player PDA

Author

Philip Graf

Date

14.03.2013

Version

0.1.0 Alpha-State

4.18.2 Constructor & Destructor Documentation

4.18.2.1 PDA::PDA (int *level* = 0)

initialize level with given value and all the rest with default values and call the init function

Parameters

level	the level of the pda

See Also

init()

4.18.3 Member Function Documentation

4.18.3.1 int PDA::getLevel ( ) const

get the current level of the PDA

Returns

the current level

Here is the caller graph for this function:

4.18.3.2 void PDA::onKeyDown (SDLKey *sym*, SDLMod *mod*, Uint16 *unicode*) [private], [virtual]

the Key-Down Event handler

See Also

Event::onKeyDown()

Reimplemented from Event.

4.18.3.3 void PDA::onWiiButtonEvent (int *button*) [private], [virtual]

the wiimote button event

See Also

Event::onWiiButtonEvent()

Reimplemented from Event.

4.18.3.4 int PDA::show ( )

the main loop this will run until running is FALSE first it calls build and every frame render will be called.

See Also

build()
render()

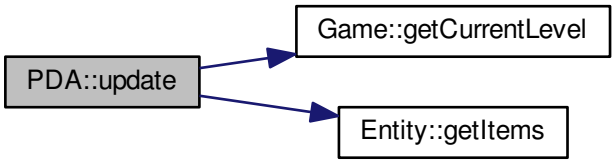
4.18.3.5 void PDA::update ( ) [private]

check for all requirements and if all in players itemlist increase the level of the level and finally call the init and build function

See Also

init()
build()

Here is the call graph for this function:



4.18.4 Member Data Documentation

4.18.4.1 SDL_Surface* PDA::display [private]

the main display of the PDA.

the curser, timer and the items will be rendered on it

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

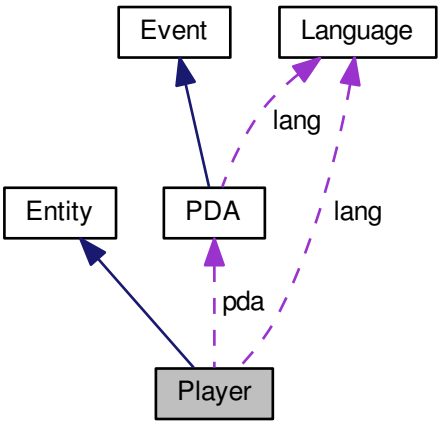
- src/PDA.h
- src/PDA.cpp

4.19 Player Class Reference

Player class.

`#include <Player.h>`

Collaboration diagram for Player:



Public Member Functions

- Player (int x, int y, int level)

  Constructor of Player.
- void use ()

  Handles the "use"-logic.
- void move ()

  Handles the player movement.
- void grab ()

  Handles joins between player and dead entities if the selected target isn't already attached to the player this method will generate a join (pic up) to the targeted entity.
- void logic ()

  Handles the logic related to the player.
- void addItem (std::string item)

  Handles the itemcollection This will call the Entity::addItem() and throw a Notification.
- PDA & getpda ()

  Returns the Players PDA.
- float getY () const

  Returns the current vertical position of the Player.
- unsigned getSelectedEntity () const

  Returns the entityList index of the currently selected(targeted) Entity.
- bool isJumping () const

  Returns TRUE if Player is currently jumping.
- void setJumping (bool jumping)

  Sets Player to jumping.
- bool isRunning () const

  Returns TRUE if Player is running.
- void setRunning (bool running)

  Sets Player to running.

Public Attributes

- PDA pda

  the players pda
- Language lang

  this objekt is used for translation

Private Attributes

- bool running

  is TRUE if the player is running
- bool jumping

  is true if the player is jumping
- unsigned selectedEntity

  the entityList index of the "active target" selected by the player
- int impactSoundPlayed

  counter to prevent the impact sound from playing multiple times when the player impacts in the ground after jumping or falling
- b2RevoluteJoint ∗ grebJoin

  connection between player and grabbed entity
- b2Vec2 ∗ distanceVec

  Vector from player to grabbed objekt.

Additional Inherited Members

4.19.1 Detailed Description

Player class.

This class handles the state of the player

Author

Felix Eckner

Date

14.03.2013

Version

0.1.0 Alpha-State

4.19.2 Constructor & Destructor Documentation

4.19.2.1 Player::Player (int *x*, int *y*, int *level*)

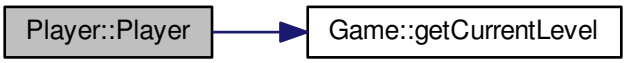
Constructor of Player.

Sets size, animation parameters, some physical attributes and load image Also the "physical body is generated and sensors for collision detection are added

Parameters

x	horizontal position
y	vertical position
level	level of the pda

Here is the call graph for this function:



4.19.3 Member Function Documentation

4.19.3.1 void Player::addItem (std::string *item*) [virtual]

Handles the itemcollection This will call the Entity::addItem() and throw a Notification.

See Also

Entity::addItem()
Notification()

Reimplemented from Entity.

Here is the call graph for this function:

4.19.3.2 PDA & Player::getpda ( )

Returns the Players PDA.

See Also

pda

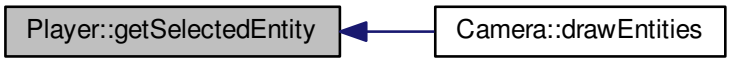
4.19.3.3 unsigned Player::getSelectedEntity ( ) const

Returns the entityList index of the currently selected(targeted) Entity.

See Also

selectedEntity

Here is the caller graph for this function:

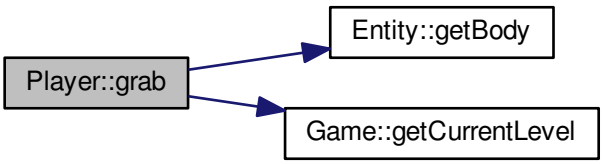


4.19.3.4 void Player::grab ( )

Handles joins between player and dead entities if the selected target isn't already attached to the player this method will generate a join (pic up) to the targeted entity.

on the other hand if there is already a join, it will be deleted and a force will be applied (throw away)

Here is the call graph for this function:



4.19.3.5 bool Player::isJumping ( ) const

Returns TRUE if Player is currently jumping.

See Also

jumping

4.19.3.6 bool Player::isRunning ( ) const

Returns TRUE if Player is running.

See Also

running

4.19.3.7 void Player::logic ( ) [virtual]

Handles the logic related to the player.

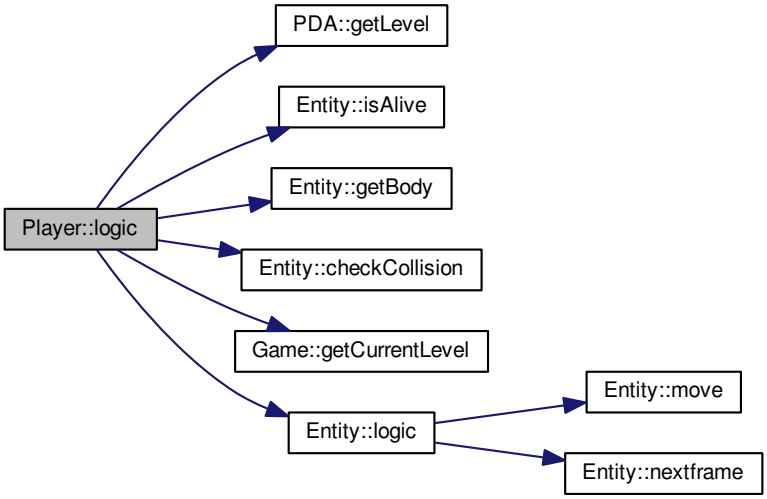
This includes targeting to check if there is a grabable dead entity(boxes or dead badguy) in range and select it, call checkCollision() method to check the collisions, kill the player if there is a collision with a badguy left or right or it is out of the level, check if player is grounded and call the Entity::logic

See Also

Entity::checkCollision()
Entity::logic()

Reimplemented from Entity.

Here is the call graph for this function:



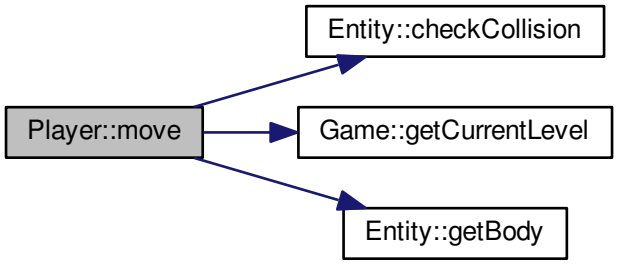
4.19.3.8 void Player::move ( ) [virtual]

Handles the player movement.

checkes the wanted direction of movement and applies the needed forces to the player also handles the the aiming if a dead body is grabbed

Reimplemented from Entity.

Here is the call graph for this function:



4.19.3.9 void Player::setJumping (bool *jumping*)

Sets Player to jumping.

Parameters

jumping	

See Also

jumping

4.19.3.10 void Player::setRunning (bool *running*)

Sets Player to running.

Parameters

jumping	

See Also

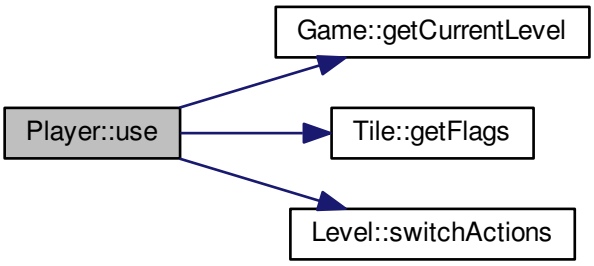
jumping

4.19.3.11 void Player::use ( )

Handles the "use"-logic.

if a dead object is below the player it will be looted (items will be thrown out and entity will be deleted). if the player is in front of a tile with a flag switchactions will be executed

Here is the call graph for this function:

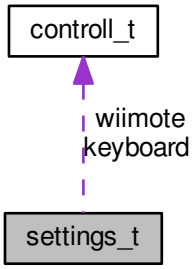


The documentation for this class was generated from the following files:

- src/Player.h
- src/Player.cpp

4.20 settings_t Struct Reference

Collaboration diagram for settings_t:



Public Attributes

- string **language**
- int **audioRate**
- Uint8 **volume**
- int **activeSlot**
- controll_t **keyboard**
- controll_t **wiimote**

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

- src/Game.h

4.21 Slot Class Reference

Slot class.

`#include <Slot.h>`

Public Member Functions

- Slot (std::string name="DrInSane")

  Constructor of Slot.
- void checkAndSetFinishedLevels (int levelnum)

  Unlocks the next level If a Level is finished successful and the level is the last playable in the current slot, the next level will be unlocked.
- int getFinishedLevels () const

  Returns the number of the highest finished level in this slot.
- void **setFinishedLevels** (int finishedLevels)
- const std::string & getName () const

  Returns the name of the slot.
- void setName (const std::string &name)

  Sets the name of a Slot.
- void setPlayerItems (const std::map< std::string, int > &playerItems)

  Sets a new map with playeritems.
- std::map< std::string, int > & getPlayerItems ()

  Returns the items of the player.
- int getPdaLevel () const

  Returns the PDA-Level of the slot.
- void setPdaLevel (int pdaLevel)

  Sets the level of the PDA.

Static Public Member Functions

- static void loadSlots ()

  Load existing slots.
- static void saveSlots ()

  Save all Slots to YAML-File.

Static Public Attributes

- static std::vector< Slot * > slots

  Stores all Slots.

Private Attributes

- std::string name

  name of the slot(player)
- int finishedLevels

  the number of the highest successfully finished level
- std::map< std::string, int > playerItems

  all the items the player already collected
- int pdaLevel

  the upgadelevel of the pda

4.21.1 Detailed Description

Slot class.

This class handles the "Savegame"-Slots and stores the game-progress

Author

Felix Eckner

Date

14.03.2013

Version

0.1.0 Alpha-State

4.21.2 Constructor & Destructor Documentation

4.21.2.1 Slot::Slot (std::string *name* = `"DrInSane"`)

Constructor of Slot.

Parameters

name	the name of the saveslot (DrInSane is default)

4.21.3 Member Function Documentation

4.21.3.1 void Slot::checkAndSetFinishedLevels (int *levelnum*)

Unlocks the next level If a Level is finished successful and the level is the last playable in the current slot, the next level will be unlocked.

Parameters

levelnum	the number of the finished level

See Also

finishedLevels

4.21.3.2 int Slot::getFinishedLevels ( ) const

Returns the number of the highest finished level in this slot.

See Also

finishedLevels

4.21.3.3 const string & Slot::getName ( ) const

Returns the name of the slot.

See Also

name

4.21.3.4 int Slot::getPdaLevel ( ) const

Returns the PDA-Level of the slot.

See Also

pdaLevel

4.21.3.5 map< string, int > & Slot::getPlayerItems ( )

Returns the items of the player.

A map of all Player-items stored in the slot will be returned

See Also

playerItems

4.21.3.6 void Slot::loadSlots ( ) [static]

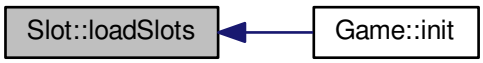
Load existing slots.

all the already existing slots in slot.yml will be loaded and stored in slots

See Also

slots

Here is the caller graph for this function:



4.21.3.7 void Slot::saveSlots ( ) [static]

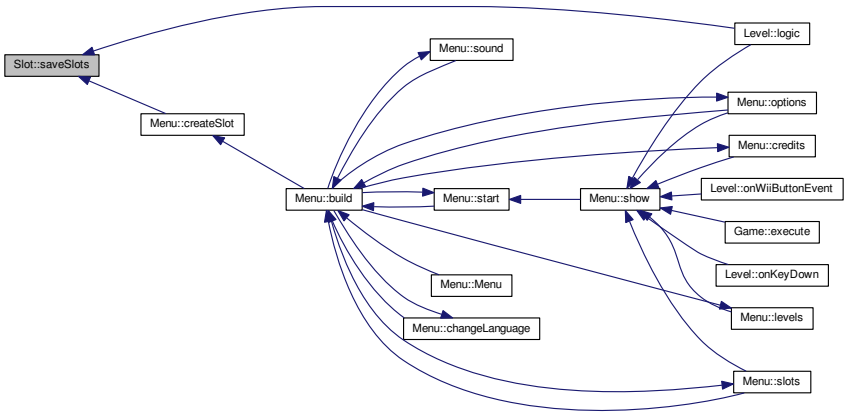
Save all Slots to YAML-File.

All the Slots in slots will be converted to YAML format and stored in slots.yml

See Also

slots

Here is the caller graph for this function:



4.21.3.8 void Slot::setName (const std::string & *name*)

Sets the name of a Slot.

Parameters

name	reference to a string

See Also

name

4.21.3.9 void Slot::setPdaLevel (int *pdaLevel*)

Sets the level of the PDA.

Parameters

pdaLevel	the new PDA-Level

See Also

pdaLevel

4.21.3.10 void Slot::setPlayerItems (const std::map< std::string, int > & *playerItems*)

Sets a new map with playeritems.

Parameters

playerItems	reference to a map with playerItems

See Also

playerItems

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

- src/Slot.h
- src/Slot.cpp

4.22 TextInput Class Reference

TextInput class.

`#include <TextInput.h>`

Public Member Functions

- TextInput (string title, int maxSize)

  Constructor of TextInput This will generate a kind of input-"window" where text can be typed.
- virtual ~TextInput ()

  Destructor of TextInput.
- string getInput ()

  Handles the typing.

Private Attributes

- string textInput

  Stores the typed text.

- int maxSize

  Defines how many digits can be typed.
- SDL_Surface * titleSurface

  Surface above the input where the title will be displayed.
- SDL_Surface * textInputSurface

  On this surface the typed text will be shown (the font)
- SDL_Surface * background

  Simply a background for the input-"window".
- SDL_Surface * textInputBackground

  The background for the textInputSurface.

4.22.1 Detailed Description

TextInput class.

This class handles "textinputfields";

Author

Philip Graf

Date

14.03.2013

Version

0.1.0 Alpha-State

4.22.2 Constructor & Destructor Documentation

4.22.2.1 TextInput::TextInput (string *title*, int *maxSize*)

Constructor of TextInput This will generate a kind of input-"window" where text can be typed.

Parameters

title	the text that will be shown in the titleSurface
maxSize	the max count of digits that can be typed

4.22.2.2 TextInput::~TextInput ( ) [virtual]

Destructor of TextInput.

This will free the used Surfaces

4.22.3 Member Function Documentation

4.22.3.1 string TextInput::getInput ( )

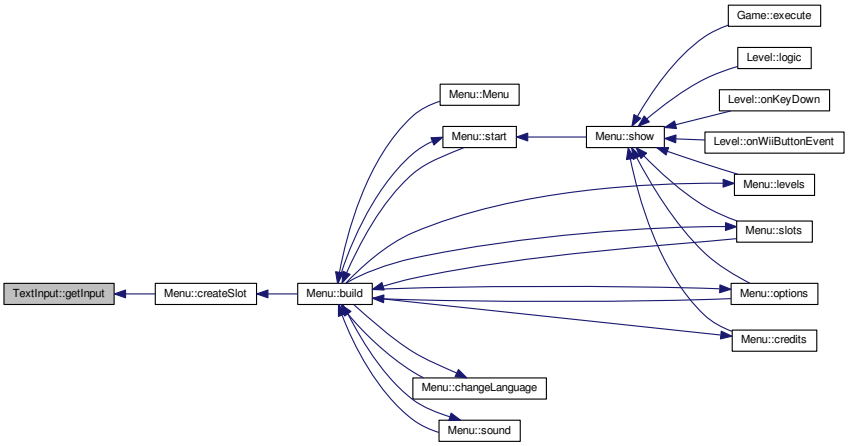
Handles the typing.

At first the position of the surfaces will be calculated. The Digits 0-9 and a-z will be added to textInput BACKSPACE will delete the last digit and ESC will clear textInput and return "". RETURN will end the typing and return textInput

See Also

textInput

Here is the caller graph for this function:



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

- src/TextInput.h
- src/TextInput.cpp

4.23 Tile Class Reference

Tile class.

`#include <Tile.h>`

Public Member Functions

- Tile (Sint64 id=0)
  Constructor of tile.
- void logic ()
  Executes the logic of the tile.
- Sint16 getId () const
  Returns the ID of the tile.
- int getCurrentframe () const
  Returns the current frame of the tile.
- void setCurrentframe (int currentframe)
  Sets the current frame of the tile.
- Sint64 getFlags () const
  Returns the flags of the tile.
- void setFlags (Sint64 flags)
  Sets new flags to the tile.
- void setId (Sint16 id)
  Sets a new id for this Tile.
- b2Body * **getBody** ()
- void **setBody** (b2Body *body)

Static Public Member Functions

- static void loadTileset ()

  Loads the image and convert it to a SDL_Surface.

Static Public Attributes

- static SDL_Surface * tileset

  This is the pointer to a BIG picture that includes all Tiles.

Private Member Functions

- void nextFrame ()

  Changes the frame of a tile.

Private Attributes

- Sint16 id

  This is the id of a tile which defines the look.
- Sint64 flags

  Stores the flags of a tile which are used for logical things like where is the finish or a switch.
- int currentframe

  Defines the frame of a tile that will be rendered.
- Uint32 timer

  Time in milliseconds when the current frame was set.
- b2Body * **body**

4.23.1 Detailed Description

Tile class.

This class will define the Tiles a Level is build of

Author

Felix Eckner

Date

14.04.2013

Version

0.1.0 Alpha-State

4.23.2 Constructor & Destructor Documentation

4.23.2.1 Tile::Tile (Sint64 *id* = 0)

Constructor of tile.

Split the given Value in ID (last 16bit) and FLAGS (first 48bit), set the first frame and the time

Parameters

id	the number got from the .map file which contains flags and ID

4.23.3 Member Function Documentation

4.23.3.1 int Tile::getCurrentframe ( ) const

Returns the current frame of the tile.

See Also

currentframe

Here is the caller graph for this function:

4.23.3.2 Sint64 Tile::getFlags ( ) const

Returns the flags of the tile.

See Also

flags

Here is the caller graph for this function:

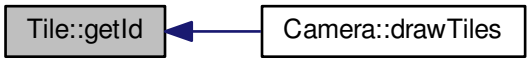
4.23.3.3 Sint16 Tile::getId ( ) const

Returns the ID of the tile.

See Also

id

Here is the caller graph for this function:

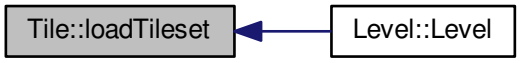


4.23.3.4 void Tile::loadTileset ( ) [static]

Loads the image and convert it to a SDL_Surface.

The image includes "textures" of all tiles (top to bottom) and frames (left to) right. also a colorkey is set to make a defined color(0xFF00FF) transparent as .bmp do not provide an alpha channel but give the best performance

Here is the caller graph for this function:



4.23.3.5 void Tile::logic ( )

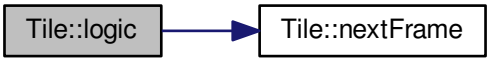
Executes the logic of the tile.

Actually only calls nextFrame

See Also

nextFrame()

Here is the call graph for this function:



4.23.3.6 void Tile::nextFrame ( ) [private]

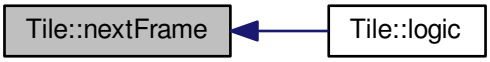
Changes the frame of a tile.

If the duration the current frame should be displayed is over it will be set to the next or the first (if the current frame is the last)

See Also

tileconf
tileduration

Here is the caller graph for this function:



4.23.3.7 void Tile::setCurrentframe (int *currentframe*)

Sets the current frame of the tile.

This is used to change the appearance of the tile. In example when a switch is used

Parameters

currentframe	the number of the frame that should be rendered

4.23.3.8 void Tile::setFlags (Sint64 *flags*)

Sets new flags to the tile.

This can be used to create a new finishpoint while running

Parameters

flags	

See Also

flags

4.23.3.9 void Tile::setId (Sint16 *id*)

Sets a new id for this Tile.

this can be used to alter a level while running

Parameters

id	the new id this tile should have from now on

See Also

id

4.23.4 Member Data Documentation

4.23.4.1 int Tile::currentframe [private]

Defines the frame of a tile that will be rendered.

Standard is 0 and will be altered if the tile is animated or changed by logic like a switch

4.23.4.2 Uint32 Tile::timer [private]

Time in milliseconds when the current frame was set.

This is needed to change the frame after the defined time for the tile to make the animation smooth

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

- src/Tile.h
- src/Tile.cpp

Index