Stickman Project

1.0

Generated by Doxygen 1.8.4

Wed Nov 27 2013 19:35:18

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 documented files with brief descriptions:

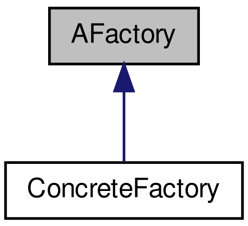
Chapter 4

Class Documentation

4.1 AFactory Class Reference

Provide a factory to create blocks.

Inheritance diagram for AFactory:



Public Member Functions

- virtual ∼AFactory ()

  Default destructor.
- IBlock ∗ createBlock (char c)

  Create a block using the build() method.
- virtual IBlock ∗ build (char s)=0

  Create a block according to the given character.

Protected Attributes

- IBlock ∗ m_block

  The block constructed.

4.1.1 Detailed Description

Provide a factory to create blocks.

Author

Adrien Bodineau and Alexandre Gomes

Version

1.0

4.1.2 Member Function Documentation

4.1.2.1 IBlock ∗ AFactory::build (char *s*) `[pure virtual]`

Create a block according to the given character.

Parameters

s	A char that indicate the block to construct

Returns

The block constructed

Implemented in ConcreteFactory.

4.1.2.2 IBlock ∗ AFactory::createBlock (char *c*)

Create a block using the build() method.

Parameters

c	A char that indicate the block to construct

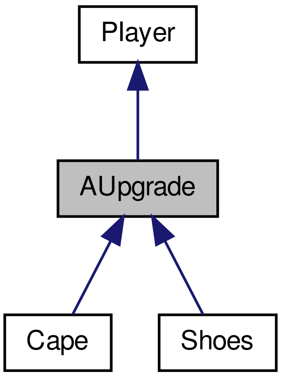
Returns

The block constructed

4.2 AUpgrade Class Reference

Abstract class for the upgrades.

Inheritance diagram for AUpgrade:



Public Member Functions

- AUpgrade ()

  Default constructor.
- AUpgrade (Player ∗character, int x, int y)

  Another constructor.
- virtual ∼AUpgrade ()

  Default destructor.

Protected Attributes

- Player ∗ m_character

  Decorated character.

4.2.1 Detailed Description

Abstract class for the upgrades.

Author

Adrien Bodineau and Alexandre Gomes

Version

1.0

4.2.2 Constructor & Destructor Documentation

4.2.2.1 AUpgrade::AUpgrade (Player ∗ *character,* int *x,* int *y*)

Another constructor.

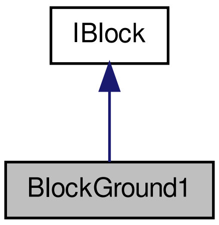
Parameters

character	Decorated character
x	Position x
y	Position y

4.3 BlockGround1 Class Reference

A concrete block that implements IBlock.

Inheritance diagram for BlockGround1:



Public Member Functions

- BlockGround1 ()

  Default constructor.
- virtual ∼BlockGround1 ()

  Default destructor.
- sf::Texture ∗ getTexture ()

  Return the texture of the block.
- sf::Sprite ∗ getSprite ()

  Return the sprite of the block.

Private Attributes

- sf::Texture ∗ m_texture

  The texture of the block.
- sf::Sprite ∗ m_sprite

  The sprite of the block.

4.3.1 Detailed Description

A concrete block that implements IBlock.

Author

Adrien Bodineau and Alexandre Gomes

Version

1.0

4.3.2 Member Function Documentation

4.3.2.1 sf::Sprite ∗ BlockGround1::getSprite (void) [virtual]

Return the sprite of the block.

Returns

The sprite of the block

Implements IBlock.

4.3.2.2 sf::Texture ∗ BlockGround1::getTexture ( ) [virtual]

Return the texture of the block.

Returns

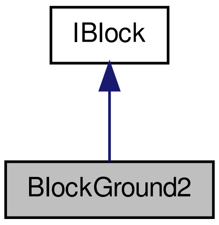
The texture of the block

Implements IBlock.

4.4 BlockGround2 Class Reference

Another concrete block that implements IBlock.

Inheritance diagram for BlockGround2:



Public Member Functions

- BlockGround2 ()

  Default constructor.
- virtual ∼BlockGround2 ()

  Default destructor.
- sf::Texture ∗ getTexture ()

  Return the texture of the block.

- sf::Sprite ∗ getSprite ()

  Return the sprite of the block.

Private Attributes

- sf::Texture ∗ m_texture

  The texture of the block.
- sf::Sprite ∗ m_sprite

  The sprite of the block.

4.4.1 Detailed Description

Another concrete block that implements IBlock.

Author

Adrien Bodineau and Alexandre Gomes

Version

1.0

4.4.2 Member Function Documentation

4.4.2.1 sf::Sprite ∗ BlockGround2::getSprite (void) [virtual]

Return the sprite of the block.

Returns

The sprite of the block

Implements IBlock.

4.4.2.2 sf::Texture ∗ BlockGround2::getTexture ( ) [virtual]

Return the texture of the block.

Returns

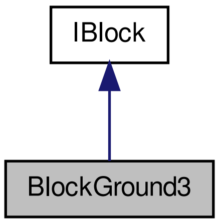
The texture of the block

Implements IBlock.

4.5 BlockGround3 Class Reference

Another concrete block that implements IBlock.

Inheritance diagram for BlockGround3:



Public Member Functions

- BlockGround3 ()

  Default constructor.
- virtual ∼BlockGround3 ()

  Default destructor.
- sf::Texture ∗ getTexture ()

  Return the texture of the block.
- sf::Sprite ∗ getSprite ()

  Return the sprite of the block.

Private Attributes

- sf::Texture ∗ m_texture

  The texture of the block.
- sf::Sprite ∗ m_sprite

  The sprite of the block.

4.5.1 Detailed Description

Another concrete block that implements IBlock.

Author

Adrien Bodineau and Alexandre Gomes

Version

1.0

4.5.2 Member Function Documentation

4.5.2.1 sf::Sprite ∗ BlockGround3::getSprite (void) [virtual]

Return the sprite of the block.

Returns

The sprite of the block

Implements IBlock.

4.5.2.2 sf::Texture ∗ BlockGround3::getTexture ( ) [virtual]

Return the texture of the block.

Returns

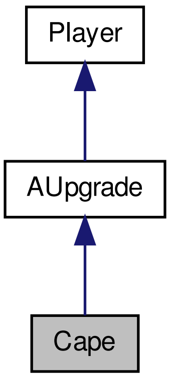
The texture of the block

Implements IBlock.

4.6 Cape Class Reference

Cape upgrade.

Inheritance diagram for Cape:



Public Member Functions

- Cape (Player ∗character, int posX, int posY)

  Default constructor.
- virtual ∼Cape ()

  Default destructor.
- virtual int getJumpSpeed ()

  Get the jumping speed value.
- virtual int getJumpHeight ()

  Get the jumping height value.
- virtual int getSpeed ()

  Get the speed value.

Additional Inherited Members

4.6.1 Detailed Description

Cape upgrade.

Author

Adrien Bodineau and Alexandre Gomes

Version

1.0

4.6.2 Constructor & Destructor Documentation

4.6.2.1 Cape::Cape (Player ∗ *character*, int *posX*, int *posY*)

Default constructor.

Parameters

character	Decorated character
posX	Position x
posY	Position y

4.6.3 Member Function Documentation

4.6.3.1 int Cape::getJumpHeight ( ) `[virtual]`

Get the jumping height value.

Returns

Jumping height value

Reimplemented from Player.

4.6.3.2 int Cape::getJumpSpeed ( ) `[virtual]`

Get the jumping speed value.

Returns

Jump speed value

Reimplemented from Player.

4.6.3.3 int Cape::getSpeed ( ) `[virtual]`

Get the speed value.

Returns

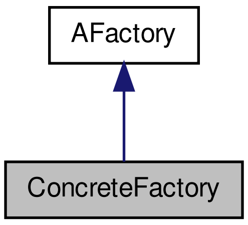
Speed value

Reimplemented from Player.

4.7 ConcreteFactory Class Reference

A concrete factory to create blocks.

Inheritance diagram for ConcreteFactory:



Public Member Functions

- ConcreteFactory ()

  Default constructor.
- ∼ConcreteFactory ()

  Default destructor.
- virtual IBlock ∗ build (char s)

  Create a block according to the given character.

Additional Inherited Members

4.7.1 Detailed Description

A concrete factory to create blocks.

Author

Adrien Bodineau and Alexandre Gomes

Version

1.0

4.7.2 Member Function Documentation

4.7.2.1 IBlock ∗ ConcreteFactory::build (char *s*) `[virtual]`

Create a block according to the given character.

Parameters

s	A char that indicate the block to construct

Returns

The block constructed

Implements AFactory.

4.8 GameManager Class Reference

Provide an instance to manage the game.

Public Member Functions

- virtual ∼GameManager (void)

  The default destructor.
- void action (void)

  Set up the parameters and run the game loop.
- void update (void)

  Update the data of the game.
- void draw (void)

  Draw the elements of the game on the screen.
- void collisionR (void)

  Check if the player is colliding with an object on his right.
- void collisionL (void)

  Check if the player is colliding with an object on his left.
- void collisionT (void)

  Check if the player is colliding with an object on his top.
- void collisionG (void)

  Check if the player is on the ground.

Static Public Member Functions

- static GameManager ∗ getInstance (void)

  Give the instance of the class, and create it if it's required.

Private Member Functions

- GameManager (int width, int height, std::string const &title)

  Private constructor of the class.

Private Attributes

- Player ∗ m_player

  Pointer on the player.
- sf::RenderWindow ∗ m_screen

  Pointer on the screen.
- sf::View ∗ m_view

Pointer on the view (2D camera)
- ILevel ∗ m_level

  Pointer on the level.
- char ∗ m_colG

  Pointer to check the ground collision.
- char ∗ m_colL

  Pointer to check the left collision.
- char ∗ m_colT

  Pointer to check the top collision.
- char ∗ m_colR

  Pointer to check the right collision.
- bool m_win

  Boolean to check if the player has won, true if he has, false otherwise.
- bool m_lost

  Boolean to check if the player has lost, true if he has, false otherwise.
- sf::Music ∗ m_music

  Pointer to the music of the current level.
- sf::Sound ∗ m_upgradeSound

  Pointer to the upgrade sound.
- sf::SoundBuffer ∗ m_upgradeSoundBuffer

  Pointer to the buffer of the upgrade sound.
- sf::Sound ∗ m_lostSound

  Pointer to the lost sound.
- sf::SoundBuffer ∗ m_lostSoundBuffer

  Pointer to the buffer of the lost sound.
- sf::Sound ∗ m_winSound

  Pointer to the win sound.
- sf::SoundBuffer ∗ m_winSoundBuffer

  Pointer to the buffer of the win sound.

Static Private Attributes

- static GameManager ∗ m_gameManager

  Static pointer on the unique instance of the class.

4.8.1 Detailed Description

Provide an instance to manage the game.

Author

Adrien Bodineau and Alexandre Gomes

Version

1.0

4.8.2 Constructor & Destructor Documentation

4.8.2.1 GameManager::GameManager (int *width,* int *height,* std::string const & *title*) [private]

Private constructor of the class.

Parameters

width	An integer representing the width of the screen
height	An integer representing the height of the screen
title	A string representing the title of the screen

4.8.3 Member Function Documentation

4.8.3.1 static GameManager ∗ GameManager::getInstance (void) `[inline]`, `[static]`

Give the instance of the class, and create it if it's required.

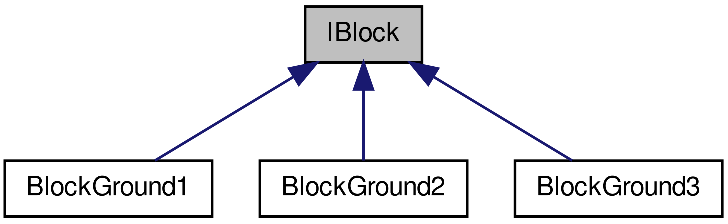
Returns

A pointer on the instance of the class

4.9 IBlock Class Reference

Interface that will be implemented by all the different kinds of blocks.

Inheritance diagram for IBlock:



Public Member Functions

- virtual sf::Texture ∗ getTexture ()=0

  Return the texture of the block.
- virtual sf::Sprite ∗ getSprite ()=0

  Return the sprite of the block.

4.9.1 Detailed Description

Interface that will be implemented by all the different kinds of blocks.

Author

Adrien Bodineau and Alexandre Gomes

Version

1.0

4.9.2 Member Function Documentation

4.9.2.1 sf::Sprite ∗ IBlock::getSprite (void) [pure virtual]

Return the sprite of the block.

Returns

The sprite of the block

Implemented in BlockGround1, BlockGround2, and BlockGround3.

4.9.2.2 sf::Texture ∗ IBlock::getTexture ( ) [pure virtual]

Return the texture of the block.

Returns

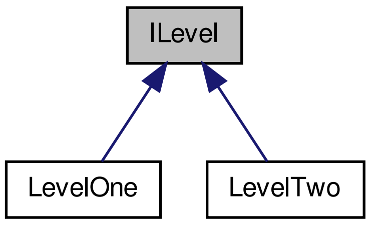
The texture of the block

Implemented in BlockGround1, BlockGround2, and BlockGround3.

4.10 ILevel Class Reference

Interface that will be implemented by the levels.

Inheritance diagram for ILevel:



Public Member Functions

- virtual ∼ILevel ()

  Default destructor.
- virtual int getHeight (void)=0

  Get the height of the level.
- virtual int getWidth (void)=0

  Get the width of the level.
- virtual void loadLevel (Player ∗player)=0

  Load the level and the player according the tile map.
- virtual void drawLevel (sf::RenderWindow ∗screen)=0

Draw all the elements of the level on the given screen.

- virtual std::vector
  < sf::Sprite ∗ > ∗ getBlocks ()=0

  Get the blocks of the level.
- virtual sf::Sprite ∗ getEndSprite (void)=0

  Get the sprite of the end.
- virtual ILevel ∗ getNext (void)=0

  Get the next level.
- virtual sf::Sprite ∗ getCapeSprite ()=0

  Get the sprite of the cape.
- virtual sf::Sprite ∗ getShoesSprite ()=0

  Get the sprite of the shoes.
- virtual void removeElement (char target)=0

  Remove an element from the map.

4.10.1 Detailed Description

Interface that will be implemented by the levels.

Author

Adrien Bodineau and Alexandre Gomes

Version

1.0

4.10.2 Member Function Documentation

4.10.2.1 void ILevel::drawLevel (sf::RenderWindow ∗ *screen*) `[pure virtual]`

Draw all the elements of the level on the given screen.

Parameters

screen	The screen where the elements will be displayed

Implemented in LevelOne, and LevelTwo.

4.10.2.2 std::vector< sf::Sprite ∗ > ∗ ILevel::getBlocks (void) `[pure virtual]`

Get the blocks of the level.

Returns

A vector with all the elements

Implemented in LevelOne, and LevelTwo.

4.10.2.3 sf::Sprite ∗ ILevel::getCapeSprite (void) `[pure virtual]`

Get the sprite of the cape.

Returns

Sprite of the cape

Implemented in LevelOne, and LevelTwo.

4.10.2.4 sf::Sprite ∗ ILevel::getEndSprite (void) [pure virtual]

Get the sprite of the end.

Returns

The sprite of the end

Implemented in LevelOne, and LevelTwo.

4.10.2.5 int ILevel::getHeight (void) [pure virtual]

Get the height of the level.

Returns

The height of the level

Implemented in LevelOne, and LevelTwo.

4.10.2.6 ILevel ∗ ILevel::getNext (void) [pure virtual]

Get the next level.

Returns

Next level, NULL if this is the last level

Implemented in LevelOne, and LevelTwo.

4.10.2.7 sf::Sprite ∗ ILevel::getShoesSprite (void) [pure virtual]

Get the sprite of the shoes.

Returns

Sprite of the shoes

Implemented in LevelOne, and LevelTwo.

4.10.2.8 int ILevel::getWidth (void) [pure virtual]

Get the width of the level.

Returns

The width of the level

Implemented in LevelOne, and LevelTwo.

4.10.2.9 void ILevel::loadLevel (Player ∗ *player*) [pure virtual]

Load the level and the player according the tile map.

Parameters

player	The pointer to the player, so the method can set his position

Implemented in LevelOne, and LevelTwo.

4.10.2.10 void ILevel::removeElement (char *target*) `[pure virtual]`

Remove an element from the map.

Parameters

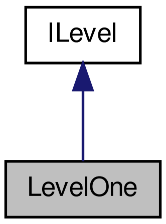
target	Character representing the element to remove

Implemented in LevelOne, and LevelTwo.

4.11 LevelOne Class Reference

Implements the first level of the game.

Inheritance diagram for LevelOne:



Public Member Functions

- LevelOne (void)

  Default constructor.
- ∼LevelOne (void)

  Default destructor.
- int getHeight (void)

  Get the height of the level.
- int getWidth (void)

  Get the width of the level.
- virtual void loadLevel (Player ∗player)

  Load the level and the player according the tile map.
- virtual void drawLevel (sf::RenderWindow ∗screen)

  Draw all the elements of the level on the given screen.
- virtual std::vector
  < sf::Sprite ∗ > ∗ getBlocks (void)

  Get the blocks of the level.
- virtual sf::Sprite ∗ getEndSprite (void)

Get a pointer on the sprite of the end.
- virtual sf::Sprite ∗ getCapeSprite ()

    Get a pointer on the sprite of the cape.
- virtual sf::Sprite ∗ getShoesSprite ()

    Get a pointer on the sprite of the shoes.
- virtual void removeElement (char target)

    Remove an element from the map.
- virtual ILevel ∗ getNext (void)

    Get the next level.

Private Attributes

- std::vector< std::string > ∗ m_tileMap

    The tile map (i.e. the map of the level)
- std::vector< sf::Sprite ∗ > ∗ m_blocks

    Contains all the elements of the level.
- IBlock ∗ m_block

    Constructed block.
- sf::Texture ∗ m_endTexture

    Texture of the end sprite.
- sf::Sprite ∗ m_endSprite

    Sprite of the end sprite.
- AFactory ∗ m_factory

    The factory that will create the blocks.
- sf::Texture ∗ m_capeTexture

    Cape texture.
- sf::Sprite ∗ m_capeSprite

    Cape sprite.
- sf::Texture ∗ m_shoesTexture

    Shoes texture.
- sf::Sprite ∗ m_shoesSprite

    Shoes sprite.
- ILevel ∗ m_next

    Next level.
- bool m_drawCape

    Boolean to know when the cape has to be draw or not.
- bool m_drawShoes

    Boolean to know when the shoes has to be draw or not.

4.11.1 Detailed Description

Implements the first level of the game.

Author

Adrien Bodineau and Alexandre Gomes

Version

1.0

4.11.2 Member Function Documentation

4.11.2.1 void LevelOne::drawLevel (sf::RenderWindow ∗ *screen*) `[virtual]`

Draw all the elements of the level on the given screen.

Parameters

screen	The screen where the elements will be displayed

Implements ILevel.

4.11.2.2 std::vector< sf::Sprite * > * LevelOne::getBlocks (void) [virtual]

Get the blocks of the level.

Returns

A vector with all the elements

Implements ILevel.

4.11.2.3 sf::Sprite * LevelOne::getCapeSprite (void) [virtual]

Get a pointer on the sprite of the cape.

Returns

The pointer on the sprite of the cape

Implements ILevel.

4.11.2.4 sf::Sprite * LevelOne::getEndSprite (void) [virtual]

Get a pointer on the sprite of the end.

Returns

The pointer on the sprite of the end

Implements ILevel.

4.11.2.5 int LevelOne::getHeight (void) [virtual]

Get the height of the level.

Returns

The height of the level

Implements ILevel.

4.11.2.6 ILevel * LevelOne::getNext (void) [virtual]

Get the next level.

Returns

Next level, NULL if this is the last level

Implements ILevel.

4.11.2.7 sf::Sprite ∗ LevelOne::getShoesSprite (void) [virtual]

Get a pointer on the sprite of the shoes.

Returns

The pointer on the sprite of the shoes

Implements ILevel.

4.11.2.8 int LevelOne::getWidth (void) [virtual]

Get the width of the level.

Returns

The width of the level

Implements ILevel.

4.11.2.9 void LevelOne::loadLevel (Player ∗ *player*) [virtual]

Load the level and the player according the tile map.

Parameters

player	The pointer to the player, so the method can set his position

Implements ILevel.

4.11.2.10 void LevelOne::removeElement (char *target*) [virtual]

Remove an element from the map.

Parameters

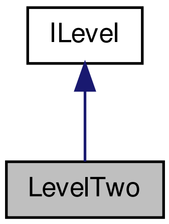
target	Character representing the element to remove

Implements ILevel.

4.12 LevelTwo Class Reference

Implements the second level of the game.

Inheritance diagram for LevelTwo:



Public Member Functions

- LevelTwo (void)

  Default constructor.
- ∼LevelTwo (void)

  Default destructor.
- int getHeight (void)

  Get the height of the level.
- int getWidth (void)

  Get the width of the level.
- virtual void loadLevel (Player ∗player)

  Load the level and the player according the tile map.
- virtual void drawLevel (sf::RenderWindow ∗screen)

  Draw all the elements of the level on the given screen.
- virtual std::vector
  < sf::Sprite ∗ > ∗ getBlocks (void)

  Get the blocks of the level.
- virtual sf::Sprite ∗ getEndSprite (void)

  Get a pointer on the sprite of the end.
- virtual sf::Sprite ∗ getCapeSprite ()

  Get a pointer on the sprite of the cape.
- virtual sf::Sprite ∗ getShoesSprite ()

  Get a pointer on the sprite of the shoes.
- virtual void removeElement (char target)

  Remove an element from the map.
- virtual ILevel ∗ getNext (void)

  Get the next level.

Private Attributes

- std::vector< std::string > ∗ m_tileMap

  The tile map (i.e. the map of the level)
- std::vector< sf::Sprite ∗ > ∗ m_blocks

  Contains all the elements of the level.
- IBlock ∗ m_block

Constructed block.
- sf::Texture ∗ m_endTexture

  Texture of the end sprite.
- sf::Sprite ∗ m_endSprite

  Sprite of the end sprite.
- AFactory ∗ m_factory

  The factory that will create the blocks.
- sf::Texture ∗ m_capeTexture

  Cape texture.
- sf::Sprite ∗ m_capeSprite

  Cape sprite.
- sf::Texture ∗ m_shoesTexture

  Shoes texture.
- sf::Sprite ∗ m_shoesSprite

  Shoes sprite.
- ILevel ∗ m_next

  Next level.
- bool m_drawCape

  Boolean to know when the cape has to be draw or not.
- bool m_drawShoes

  Boolean to know when the shoes has to be draw or not.

4.12.1 Detailed Description

Implements the second level of the game.

Author

Adrien Bodineau and Alexandre Gomes

Version

1.0

4.12.2 Member Function Documentation

4.12.2.1 void LevelTwo::drawLevel (sf::RenderWindow ∗ *screen*) `[virtual]`

Draw all the elements of the level on the given screen.

Parameters

screen	The screen where the elements will be displayed

Implements ILevel.

4.12.2.2 std::vector< sf::Sprite ∗ > ∗ LevelTwo::getBlocks (void) `[virtual]`

Get the blocks of the level.

Returns

A vector with all the elements

Implements ILevel.

4.12.2.3 sf::Sprite ∗ LevelTwo::getCapeSprite (void) [virtual]

Get a pointer on the sprite of the cape.

Returns

The pointer on the sprite of the cape

Implements ILevel.

4.12.2.4 sf::Sprite ∗ LevelTwo::getEndSprite (void) [virtual]

Get a pointer on the sprite of the end.

Returns

The pointer on the sprite of the end

Implements ILevel.

4.12.2.5 int LevelTwo::getHeight (void) [virtual]

Get the height of the level.

Returns

The height of the level

Implements ILevel.

4.12.2.6 ILevel ∗ LevelTwo::getNext (void) [virtual]

Get the next level.

Returns

Next level, NULL if this is the last level

Implements ILevel.

4.12.2.7 sf::Sprite ∗ LevelTwo::getShoesSprite (void) [virtual]

Get a pointer on the sprite of the shoes.

Returns

The pointer on the sprite of the shoes

Implements ILevel.

4.12.2.8 int LevelTwo::getWidth (void) [virtual]

Get the width of the level.

Returns

The width of the level

Implements ILevel.

4.12.2.9 void LevelTwo::loadLevel (Player * *player*) [virtual]

Load the level and the player according the tile map.

Parameters

player	The pointer to the player, so the method can set his position

Implements ILevel.

4.12.2.10 void LevelTwo::removeElement (char *target*) [virtual]

Remove an element from the map.

Parameters

target	Character representing the element to remove

Implements ILevel.

4.13 Menu Class Reference

Game menu.

Public Member Functions

- virtual ∼Menu (void)

  Default destructor.
- void action (void)

  Method that implements the behavior of the menu.

Static Public Member Functions

- static Menu ∗ getInstance (void)

  Get menu instance.

Private Member Functions

- Menu (void)

  Private constructor.

Private Attributes

- sf::RenderWindow ∗ m_screen

  Screen.
- sf::Texture ∗ m_bgTexture

  Background texture.
- sf::Sprite ∗ m_bgSprite

  Background sprite.
- sf::Texture ∗ m_saxTexture

  Epic sax guy texture.
- sf::Texture ∗ m_saxDanceTexture

  Epic sax guy dance texture.
- sf::Texture ∗ m_playTexture

  Play button texture.
- sf::Sprite ∗ m_playSprite

Play button sprite.

- sf::Texture ∗ m_activePlayTexture

  Active play button texture.
- sf::Texture ∗ m_optionTexture

  Option button texture.
- sf::Sprite ∗ m_optionSprite

  Option button sprite.
- sf::Texture ∗ m_activeOptionTexture

  Active option button sprite.
- sf::Music ∗ m_music

  Music of the game.
- sf::Sound ∗ m_sound

  Sound of the buttons.
- sf::SoundBuffer ∗ m_soundBuffer

  Sound buffer of the buttons.
- bool m_played

  Boolean to know when the buttons sound has to be played or not.
- bool m_onButton

  Boolean to know when the mouse is currently on the button.
- bool m_saxguy

  Boolean to know when the EPIC SAX GUY mode is ON or OFF.

Static Private Attributes

- static Menu ∗ m_menu

  Instance.

4.13.1 Detailed Description

Game menu.

Author

Adrien Bodineau and Alexandre Gomes

Version

1.0

4.13.2 Member Function Documentation

4.13.2.1 static Menu ∗ Menu::getInstance (void) `[inline], [static]`

Get menu instance.

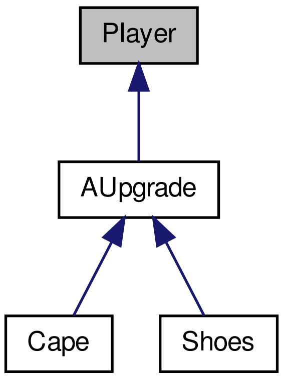
Returns

Menu instance

4.14 Player Class Reference

Player class.

Inheritance diagram for Player:



Public Member Functions

- Player (int x=0, int y=0)

  Default constructor.
- virtual ∼Player (void)

  Default destructor.
- virtual void setOnGround (bool value)

  Set onGround.
- virtual void goRight (void)

  Function to make the player go to the right.
- virtual void goLeft (void)

  Function to make the player go to the left.
- virtual void fall (void)

  Function to make the player fall.
- virtual void controls (char collisionR, char collisionL, char collisionT, char collisionG)

  Indicate what the player have to do according to the collisions and the press buttons.
- virtual void jump (void)

  Set up the jump.
- virtual void jumpAnimation (char collisionR, char collisionL, char collisionT, char collisionG)

  Make the player jump.
- virtual void setJumping (bool value)

  Set the jumping value.
- virtual void setSpeed (int speed)

  Set the speed value.
- virtual void setJumpSpeed (int jumpSpeed)

  Set the jumping speed value.
- virtual void setJumpHeight (int jumpHeight)

Set the jumping height value.
- virtual sf::IntRect ∗ getRect (void)

  Get the rect of the player.
- virtual bool getJumping (void)

  Get the jumping value.
- virtual int getSpeed ()

  Get the speed value.
- virtual int getJumpSpeed ()

  Get the jumping speed value.
- virtual int getJumpHeight ()

  Get the jumping height value.
- virtual sf::Sprite ∗ getSprite (void)

  Get the sprite of the player.
- virtual bool getOnGround (void)

  Get the boolean onGround.
- virtual int getPositionX (int inc=0)

  Get the position x.
- virtual int getPositionY (int inc=0)

  Get the position y.
- virtual int getWidth (int inc=0)

  Get the player's width.
- virtual int getHeight (int inc=0)

  Get the player's height.
- virtual sf::Texture ∗ getTexture ()

  Get the player's texture.
- virtual sf::Texture ∗ getRunTexture1 ()

  Get the player's run texture 1.
- virtual sf::Texture ∗ getRunTexture2 ()

  Get the player's run texture 2.
- virtual sf::Texture ∗ getRunTexture3 ()

  Get the player's run texture 3.
- virtual sf::Texture ∗ getRunTexture4 ()

  Get the player's run texture 4.
- virtual sf::Texture ∗ getRunTexture5 ()

  Get the player's run texture 5.
- virtual sf::Texture ∗ getRunTexture6 ()

  Get the player's run texture 6.
- virtual sf::Sound ∗ getJumpSound ()

  Get the jump sound.
- virtual sf::SoundBuffer ∗ getJumpSoundBuffer ()

  Get the jump sound buffer.
- virtual int getJump ()

  Get the value of jump.
- virtual double getCurrentFrame ()

  Get the player's current frame.

Protected Attributes

- sf::Texture ∗ m_texture

  Player's texture.
- sf::Texture ∗ m_runTexture1

  Player's run texture 1.
- sf::Texture ∗ m_runTexture2

  Player's run texture 2.
- sf::Texture ∗ m_runTexture3

  Player's run texture 3.
- sf::Texture ∗ m_runTexture4

  Player's run texture 4.
- sf::Texture ∗ m_runTexture5

  Player's run texture 5.
- sf::Texture ∗ m_runTexture6

  Player's run texture 6.
- sf::Texture ∗ m_animTab [6]

  Array of the run textures.
- sf::Texture ∗ m_jumpTexture

  Player's jump texture.
- sf::Texture ∗ m_fallTexture

  Player's fall texture.
- sf::Sprite ∗ m_sprite

  Player's sprite.
- sf::Sound ∗ m_jumpSound

  Jump sound.
- sf::SoundBuffer ∗ m_jumpSoundBuffer

  Jump sound buffer.
- sf::IntRect ∗ m_rect

  Player's rect.
- int m_dx

  Position x.
- int m_dy

  Position y.
- bool m_onGround

  Boolean to know if the player is on the ground.
- bool m_jumping

  Boolean to know if the player is jumping.
- int m_jump

  Value of the jump.
- double m_currentFrame

  Value of the current frame.
- int m_speed

  Player's speed.
- int m_jumpSpeed

  Jump speed.
- int m_jumpHeight

  Jump height.

4.14.1 Detailed Description

Player class.

Author

Adrien Bodineau and Alexandre Gomes

Version

1.0

4.14.2 Constructor & Destructor Documentation

4.14.2.1 Player::Player (int *x* = 0, int *y* = 0)

Default constructor.

Parameters

x	Position x
y	Position y

4.14.3 Member Function Documentation

4.14.3.1 void Player::controls (char *collisionR,* char *collisionL,* char *collisionT,* char *collisionG*) [virtual]

Indicate what the player have to do according to the collisions and the press buttons.

Parameters

collisionR	Character to indicate the right collision
collisionL	Character to indicate the left collision
collisionT	Character to indicate the top collision
collisionG	Character to indicate the ground collision

4.14.3.2 double Player::getCurrentFrame ( ) [virtual]

Get the player's current frame.

Returns

Player's current frame

4.14.3.3 int Player::getHeight (int *inc* = 0) [virtual]

Get the player's height.

Returns

Player's height

4.14.3.4 int Player::getJump ( ) [virtual]

Get the value of jump.

Returns

Jump value

4.14.3.5 int Player::getJumpHeight ( ) [virtual]

Get the jumping height value.

Returns

Jumping height value

Reimplemented in Cape, and Shoes.

4.14.3.6 bool Player::getJumping (void) [virtual]

Get the jumping value.

Returns

Jumping value

4.14.3.7 sf::Sound * Player::getJumpSound ( ) [virtual]

Get the jump sound.

Returns

Jump sound

4.14.3.8 sf::SoundBuffer * Player::getJumpSoundBuffer ( ) [virtual]

Get the jump sound buffer.

Returns

Jump sound buffer

4.14.3.9 int Player::getJumpSpeed ( ) [virtual]

Get the jumping speed value.

Returns

Jump speed value

Reimplemented in Cape, and Shoes.

4.14.3.10 bool Player::getOnGround (void) [virtual]

Get the boolean onGround.

Returns

Value of onGround

4.14.3.11 int Player::getPositionX (int *inc* = 0) [virtual]

Get the position x.

Parameters

inc	Get the position x+inc

Returns

Position x

4.14.3.12 int Player::getPositionY (int *inc* = 0) [virtual]

Get the position y.

Parameters

inc	Get the position y+inc

Returns

Position y

4.14.3.13 sf::IntRect * Player::getRect (void) [virtual]

Get the rect of the player.

Returns

Player's rect

4.14.3.14 sf::Texture * Player::getRunTexture1 ( ) [virtual]

Get the player's run texture 1.

Returns

Player's run texture 1

4.14.3.15 sf::Texture * Player::getRunTexture2 ( ) [virtual]

Get the player's run texture 2.

Returns

Player's run texture 2

4.14.3.16 sf::Texture ∗ Player::getRunTexture3 ( ) [virtual]

Get the player's run texture 3.

Returns

Player's run texture 3

4.14.3.17 sf::Texture ∗ Player::getRunTexture4 ( ) [virtual]

Get the player's run texture 4.

Returns

Player's run texture 4

4.14.3.18 sf::Texture ∗ Player::getRunTexture5 ( ) [virtual]

Get the player's run texture 5.

Returns

Player's run texture 5

4.14.3.19 sf::Texture ∗ Player::getRunTexture6 ( ) [virtual]

Get the player's run texture 6.

Returns

Player's run texture 6

4.14.3.20 int Player::getSpeed ( ) [virtual]

Get the speed value.

Returns

Speed value

Reimplemented in Cape, and Shoes.

4.14.3.21 sf::Sprite ∗ Player::getSprite (void) [virtual]

Get the sprite of the player.

Returns

Sprite of the player

4.14.3.22 sf::Texture ∗ Player::getTexture ( ) [virtual]

Get the player's texture.

Returns

Player's texture

4.14.3.23 int Player::getWidth (int *inc* = 0) [virtual]

Get the player's width.

Returns

Player's width

4.14.3.24 void Player::jumpAnimation (char *collisionR,* char *collisionL,* char *collisionT,* char *collisionG*) [virtual]

Make the player jump.

Parameters

collisionR	
collisionL	
collisionT	
collisionG	

4.14.3.25 void Player::setJumpHeight (int *jumpHeight*) [virtual]

Set the jumping height value.

Parameters

jumpHeight	New value of jumping height

4.14.3.26 void Player::setJumping (bool *value*) [virtual]

Set the jumping value.

Parameters

value	New value of jumping

4.14.3.27 void Player::setJumpSpeed (int *jumpSpeed*) [virtual]

Set the jumping speed value.

Parameters

jumpSpeed	New value of jump speed

4.14.3.28 void Player::setOnGround (bool *value*) [virtual]

Set onGround.

Parameters

value	New value of onGround

4.14.3.29 void Player::setSpeed (int *speed*) [virtual]

Set the speed value.

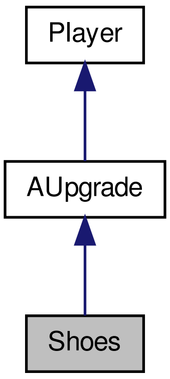
Parameters

speed	New value of speed

4.15 Shoes Class Reference

Shoes class.

Inheritance diagram for Shoes:



Public Member Functions

- Shoes (Player ∗character, int posX, int posY)

  Default constructor.
- virtual ∼Shoes ()

  Default destructor.
- virtual int getJumpSpeed ()

  Get the jumping speed value.
- virtual int getJumpHeight ()

  Get the jumping height value.
- virtual int getSpeed ()

  Get the speed value.

Additional Inherited Members

4.15.1 Detailed Description

Shoes class.

Author

Adrien Bodineau and Alexandre Gomes

Version

1.0

4.15.2 Constructor & Destructor Documentation

4.15.2.1 Shoes::Shoes (**Player** ∗ *character*, int *posX*, int *posY*)

Default constructor.

Parameters

character	Decorated character
posX	Position x
posY	Position y

4.15.3 Member Function Documentation

4.15.3.1 int Shoes::getJumpHeight ( ) `[virtual]`

Get the jumping height value.

Returns

Jumping height value

Reimplemented from Player.

4.15.3.2 int Shoes::getJumpSpeed ( ) `[virtual]`

Get the jumping speed value.

Returns

Jump speed value

Reimplemented from Player.

4.15.3.3 int Shoes::getSpeed ( ) `[virtual]`

Get the speed value.

Returns

Speed value

Reimplemented from Player.

Chapter 5

File Documentation

5.1 include/Decorator/AUpgrade.h File Reference

Classes

- class AUpgrade
  Abstract class for the upgrades.

5.2 include/Decorator/Cape.h File Reference

Classes

- class Cape
  Cape upgrade.

5.3 include/Decorator/Player.h File Reference

Classes

- class Player
  Player class.

5.4 include/Decorator/Shoes.h File Reference

Classes

- class Shoes
  Shoes class.

5.5 include/Factory/AFactory.h File Reference

Classes

- class AFactory
  Provide a factory to create blocks.

5.6 include/Factory/BlockGround1.h File Reference

Classes

- class BlockGround1

  A concrete block that implements IBlock.

5.7 include/Factory/BlockGround2.h File Reference

Classes

- class BlockGround2

  Another concrete block that implements IBlock.

5.8 include/Factory/BlockGround3.h File Reference

Classes

- class BlockGround3

  Another concrete block that implements IBlock.

5.9 include/Factory/ConcreteFactory.h File Reference

Classes

- class ConcreteFactory

  A concrete factory to create blocks.

5.10 include/Factory/IBlock.h File Reference

Classes

- class IBlock

  Interface that will be implemented by all the different kinds of blocks.

5.11 include/Singleton/GameManager.h File Reference

Classes

- class GameManager

  Provide an instance to manage the game.

5.12 include/Singleton/Menu.h File Reference

Classes

- class Menu

  Game menu.

5.13 include/Strategy/ILevel.h File Reference

Classes

- class ILevel

  Interface that will be implemented by the levels.

5.14 include/Strategy/LevelOne.h File Reference

Classes

- class LevelOne

  Implements the first level of the game.

5.15 include/Strategy/LevelTwo.h File Reference

Classes

- class LevelTwo

  Implements the second level of the game.

Index