Arcade Toulouse 2022

Generated by Doxygen 1.9.0

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

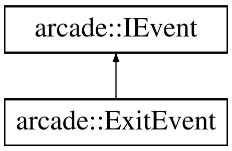
Class Documentation

3.1 arcade::ExitEvent Class Reference

Event to exit the program.

```
#include <ExitEvent.hpp>
```

Inheritance diagram for arcade::ExitEvent:



Additional Inherited Members

3.1.1 Detailed Description

Event to exit the program.

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

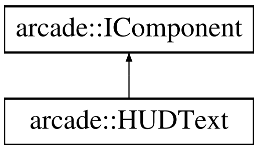
- include/events/ExitEvent.hpp

3.2 arcade::HUDText Class Reference

The HUDText component is used by all graphical libraries to display the string given by the '_text' properties. This component is used to display text such as hp point, score...

```
#include <HUDText.hpp>
```

Inheritance diagram for arcade::HUDText:



Classes

- struct rgb_s

  rgb structure for text's color handling

Public Types

- typedef struct arcade::HUDText::rgb_s rgb_t

  rgb structure for text's color handling

Public Member Functions

- HUDText (const std::string &text, std::string font, uint16_t r=255, uint16_t g=255, uint16_t b=255)

  Construct a new HUDText object.

Public Attributes

- std::string _text

  text to display
- std::string _font

  filepath for text's font
- rgb_t _color

  text's color (as rgb color)

3.2.1 Detailed Description

The HUDText component is used by all graphical libraries to display the string given by the '_text' properties. This component is used to display text such as hp point, score...

AN ENTITY CAN EITHER BE OF THE ORDER OF THE HUD OR NOT BE. AN ENTITY WITH A COMPONENT HUD CAN'T HAVE SPRITETEXT AND/OR SPRITE2D COMPONENT(S)

3.2.2 Member Typedef Documentation

3.2.2.1 rgb_t

```
typedef struct arcade::HUDText::rgb_s arcade::HUDText::rgb_t
```

rgb structure for text's color handling

Parameters

r	for red color proportion
g	for green color proportion
b	for blue color proportion

3.2.3 Constructor & Destructor Documentation

3.2.3.1 HUDText()

```
arcade::HUDText::HUDText (
            const std::string & text,
            std::string font,
            uint16_t r = 255,
            uint16_t g = 255,
            uint16_t b = 255 )  [inline]
```

Construct a new HUDText object.

the color is set to white by default

Parameters

text	text to display
font	filepath for text's font
r	text color, red proportion, 255 by default
g	text color, green proportion, 255 by default
b	text color, blue proportion, 255 by default

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

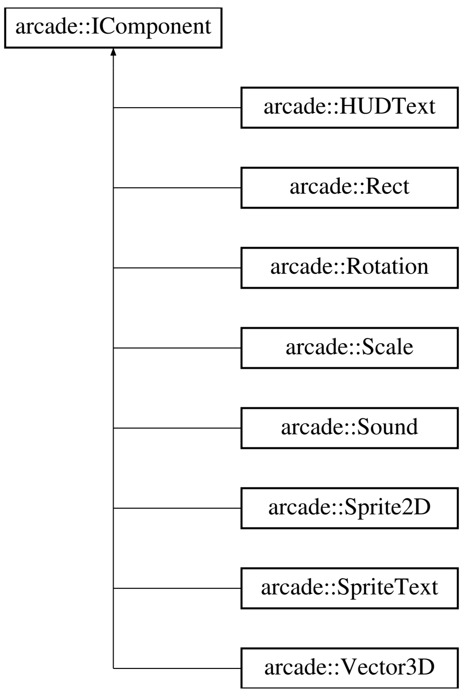
- include/components/HUDText.hpp

3.3 arcade::IComponent Class Reference

There are several components inheriting from IComponent and used by game and graphical libraries. To create and use libraries you must handle all components.

```
#include <IComponent.hpp>
```

Inheritance diagram for arcade::IComponent:



3.3.1 Detailed Description

There are several components inheriting from IComponent and used by game and graphical libraries. To create and use libraries you must handle all components.

YOU CAN ADD NEW COMPONENTS ONLY IF YOU USE YOUR OWN LIBRARIES DUE TO THE NECESSITY FOR ALL LIBRARIES TO KNOW THE SAME COMPONENTS

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

- include/IComponent.hpp

3.4 arcade::ICore Class Reference

Interface of the core class needed by graphical librairies to send events to the core.

```
#include <ICore.hpp>
```

Public Member Functions

- virtual void manageEvent (IEvent &event)=0

  Used by the graphical librairies to send events to the core. This method needs to be passed as a pointer to a method along with a reference to ICore.

3.4.1 Detailed Description

Interface of the core class needed by graphical librairies to send events to the core.

You are expected to provide an implementation of this interface, in the form of a Core class, used to contain your libraries, both graphical and non-graphical.

The core is also responsible for the main game loop, and measuring time between frames, aswell as loading, switching and unloading libraries.

This interface exists to allow an evenmential event handling. When an event is received from a graphical library, it is sent to the core, who first checks if the event is an exit event or a library switch, and if not sends the event to the graphical library.

3.4.2 Member Function Documentation

3.4.2.1 manageEvent()

```
virtual void arcade::ICore::manageEvent (
            IEvent & event )  [pure virtual]
```

Used by the graphical librairies to send events to the core. This method needs to be passed as a pointer to a method along with a reference to ICore.

Parameters

event	The event sent to the core

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

- include/ICore.hpp

3.5 arcade::IEntity Class Reference

This interface is used co create a game entity described by a vector of components, and a vector of tags.

```
#include <IEntity.hpp>
```

Public Member Functions

- virtual std::vector< std::unique_ptr< IComponent > > & getComponents ()=0

  Getter for the components of an entity.
- virtual bool hasTag (const std::string &tag)=0

  Check if the entity has the given tag.

3.5.1 Detailed Description

This interface is used co create a game entity described by a vector of components, and a vector of tags.

You are expected to implement a class inheriting from IEntity, and implementing all of its virtual methods.

3.5.2 Member Function Documentation

3.5.2.1 getComponents()

```
virtual std::vector<std::unique_ptr<IComponent> >& arcade::IEntity::getComponents ( )  [pure
virtual]
```

Getter for the components of an entity.

we are using smart pointers to wrap our IComponents to avoid manual memory management

Returns

std::vector<std::unique_ptr<IComponent>> & : a reference to the vector of IComponents

3.5.2.2 hasTag()

```
virtual bool arcade::IEntity::hasTag (
            const std::string & tag )  [pure virtual]
```

Check if the entity has the given tag.

tags are used to identify different types of entities in the game, without having to check their components, saving time.

Since we have not defined any common tags, you are to use tags only internally in your game, and not between a game and a graphical library.

Parameters

const	std::string & : the tag to check

Returns

true if the tag is found, false otherwise

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

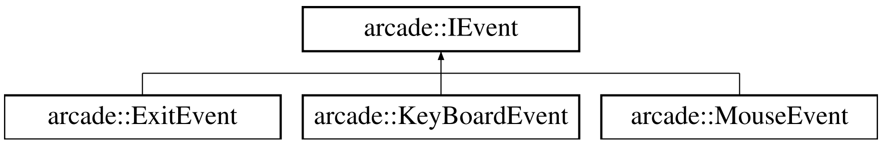
- include/IEntity.hpp

3.6 arcade::IEvent Class Reference

This Interface is used to encapsulate all events, sent from any graphical library.

```
#include <IEvent.hpp>
```

Inheritance diagram for arcade::IEvent:



Public Types

- enum ButtonState { **None** = -1, **Pressed**, **Released** }

  Describes the stated of a button. Works both for mouse and keyboard.

3.6.1 Detailed Description

This Interface is used to encapsulate all events, sent from any graphical library.

You can consult every possible event type in the event folder from this repository.

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

- include/IEvent.hpp

3.7 arcade::IGame Class Reference

System responsible for the game logic of a game.

```
#include <IGame.hpp>
```

Public Member Functions

- virtual IScene & init ()=0

  Initializes the game, populating the scenes and returns the first scene to be rendered.
- virtual IScene & update (const std::uint64_t &deltaTime)=0

  Updates the current scene.
- virtual void manageEvents (IEvent &event)=0

  Manages the events sent by the graphical library, through the core.
- virtual void destroy ()=0

  This is a method for you to do cleanup when the game is destroyed.

3.7.1 Detailed Description

System responsible for the game logic of a game.

3.7.2 Member Function Documentation

3.7.2.1 init()

```
virtual IScene& arcade::IGame::init ( )  [pure virtual]
```

Initializes the game, populating the scenes and returns the first scene to be rendered.

Returns

The current scene of the game, to be

3.7.2.2 manageEvents()

```
virtual void arcade::IGame::manageEvents (
             IEvent & event )  [pure virtual]
```

Manages the events sent by the graphical library, through the core.

Parameters

event	the event to be managed

3.7.2.3 update()

```
virtual IScene& arcade::IGame::update (
             const std::uint64_t & deltaTime )  [pure virtual]
```

Updates the current scene.

Parameters

deltaTime	time elapsed since the last update, in miliseconds

Returns

the current scene of the game to be updated

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

- include/IGame.hpp

3.8 arcade::IGraphical Class Reference

System responsible for handling inputs, sound and rendering for a scene from a IGame.

```
#include <IGraphical.hpp>
```

Public Member Functions

- virtual void init (IScene &scene)=0
- virtual void update (IScene &scene)=0
- virtual void destroy (IScene &scene)=0

3.8.1 Detailed Description

System responsible for handling inputs, sound and rendering for a scene from a IGame.

You are expected to create your own implementation of IGraphical for instance SFML or NCurses class, inheriting from IGraphical and implementing all of its methods.

3.8.2 Member Function Documentation

3.8.2.1 destroy()

```
virtual void arcade::IGraphical::destroy (
             IScene & scene )  [pure virtual]
```

This method is called to do cleanup before quitting your graphical library.

Typically you will destroy your window, or other library specific resources.

Parameters

scene	this is the scene that contains all the entities from a game Scene. It is up to the implementation to find which entities are graphical entities and which are not, using the entity's components.

Returns

void

3.8.2.2 init()

```
virtual void arcade::IGraphical::init (
            IScene & scene )  [pure virtual]
```

This init method is called in order to initialize everything the graphical library needs to render a scene.

That can be a window, a texture / sprite cache, fonts, etc.

This method is called once at the beginning of the game, and subsequently if the game has been destroyed and needs to be restarted.

Parameters

scene	this is the scene that contains all the entities from a game Scene. It is up to the implementation to find which entities are graphical entities and which are not, using the entity's components.

Returns

void

3.8.2.3 update()

```
virtual void arcade::IGraphical::update (
            IScene & scene )  [pure virtual]
```

The update method is called when the game needs to be rendered. That doesn't necessarily mean every frame, it could be later.

This is where you should update the position of your sprites, unload or load textures, update sound, etc.

Parameters

scene	this is the scene that contains all the entities from a game Scene. It is up to the implementation to find which entities are graphical entities and which are not, using the entity's components.

Returns

void

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

- include/IGraphical.hpp

3.9 arcade::IScene Class Reference

This interface represents a scene from a game, which contains a vector of entities that describe a particular moment from a game.

```
#include <IScene.hpp>
```

Public Member Functions

- virtual std::vector< std::shared_ptr< IEntity > > & getEntities ()=0

  Gets the entities from the scene.
- virtual uint32_t getSceneWidth () const =0

  Gets the Scene Width.
- virtual uint32_t getSceneHeight () const =0

  Gets the Scene Height in length units.

3.9.1 Detailed Description

This interface represents a scene from a game, which contains a vector of entities that describe a particular moment from a game.

A scene can describe a menu, a game scene, a pause menu or any other state of a game that you might want to isolate. You could also handle all your game logic from a single scene, but that is not advised.

3.9.2 Member Function Documentation

3.9.2.1 getEntities()

```
virtual std::vector<std::shared_ptr<IEntity> >& arcade::IScene::getEntities ( )  [pure virtual]
```

Gets the entities from the scene.

Returns

std::vector<std::shared_ptr<IEntity>> & : A reference to the vector of entities contained in the scene

3.9.2.2 getSceneHeight()

```
virtual uint32_t arcade::IScene::getSceneHeight ( ) const  [pure virtual]
```

Gets the Scene Height in length units.

Returns

uint32_t : the scene heigth in length units

3.9.2.3 getSceneWidth()

```
virtual uint32_t arcade::IScene::getSceneWidth ( ) const  [pure virtual]
```

Gets the Scene Width.

Returns

uint32_t : the scene width in length units

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

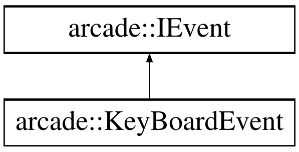
- include/IScene.hpp

3.10 arcade::KeyBoardEvent Class Reference

Event describing a keypress.

```
#include <KeyBoardEvent.hpp>
```

Inheritance diagram for arcade::KeyBoardEvent:



Public Types

- enum Key {
  Unknown = -1, **A** = 0, **B**, **C**,
  D, **E**, **F**, **G**,
  H, **I**, **J**, **K**,
  L, **M**, **N**, **O**,
  P, **Q**, **R**, **S**,
  T, **U**, **V**, **W**,
  X, **Y**, **Z**, **Num0**,
  Num1, **Num2**, **Num3**, **Num4**,
  Num5, **Num6**, **Num7**, **Num8**,
  Num9, **Escape**, **LControl**, **LShift**,
  LAlt, **LSystem**, **RControl**, **RShift**,
  RAlt, **RSystem**, **Menu**, **LBracket**,
  RBracket, **Semicolon**, **Comma**, **Period**,
  Quote, **Slash**, **Backslash**, **Tilde**,
  Equal, **Hyphen**, **Space**, **Enter**,
  Backspace, **Tab**, **PageUp**, **PageDown**,
  End, **Home**, **Insert**, **Delete**,
  Add, **Subtract**, **Multiply**, **Divide**,
  Left, **Right**, **Up**, **Down**,
  Numpad0, **Numpad1**, **Numpad2**, **Numpad3**,
  Numpad4, **Numpad5**, **Numpad6**, **Numpad7**,
  Numpad8, **Numpad9**, **F1**, **F2**,
  F3, **F4**, **F5**, **F6**,
  F7, **F8**, **F9**, **F10**,
  F11, **F12**, **F13**, **F14**,
  F15, **Pause**, **KeyCount**, **Dash** = Hyphen,
  BackSpace = Backspace, **BackSlash** = Backslash, **SemiColon** = Semicolon, **Return** = Enter }

  Describes which key is pressed.

Public Member Functions

- **KeyBoardEvent** (const Key &key, ButtonState state)

Public Attributes

- Key _key

  Key value.
- ButtonState _state

  Key state using the enum described in IEvent.

3.10.1 Detailed Description

Event describing a keypress.

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

- include/events/KeyBoardEvent.hpp

3.11 metadata Struct Reference

Structure containing the metadata of the library (type, name, description)

```
#include <api.h>
```

Public Types

- enum { **UNKNOWN** = -1, **GAME**, **GRAPHIC** }

Public Attributes

- enum metadata:: { ... } **type**
- const char ∗ **name**
- const char ∗ **desc**

3.11.1 Detailed Description

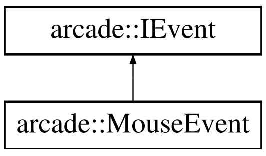
Structure containing the metadata of the library (type, name, description)

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

- include/api.h

3.12 arcade::MouseEvent Class Reference

Inheritance diagram for arcade::MouseEvent:



Public Types

- enum **Button** { **None** = -1, **Left**, **Middle**, **Right** }

Public Member Functions

- **MouseEvent** (double x, double y, ButtonState state=ButtonState::None, Button button=None)

Public Attributes

- Button **_btn**
- double **_x**
- double **_y**
- ButtonState **_state**

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

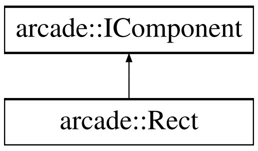
- include/events/MouseEvent.hpp

3.13 arcade::Rect Class Reference

The Rect component give the position, the width and height of an image part in the image of the Sprite2D component for graphical library with graphical interface. This component is usefull for spritesheet handling/animation.

```
#include <Rect.hpp>
```

Inheritance diagram for arcade::Rect:



Public Member Functions

- Rect (double left=0, double top=0, double w=0, double h=0)

  Construct a new Rect object.

Public Attributes

- double _left

    rect's left position in the associated sprite
- double _top

    rect's top position in the associated sprite
- double _width

    rect's width in the associated sprite
- double _height

    rect's height in the associated sprite

3.13.1 Detailed Description

The Rect component give the position, the width and height of an image part in the image of the Sprite2D component for graphical library with graphical interface. This component is usefull for spritesheet handling/animation.

3.13.2 Constructor & Destructor Documentation

3.13.2.1 Rect()

```
arcade::Rect::Rect (
            double left = 0,
            double top = 0,
            double w = 0,
            double h = 0 )  [inline]
```

Construct a new Rect object.

Parameters

left	rect's left position in the associated sprite
top	rect's top position in the associated sprite
w	rect's width in the associated sprite
h	rect's height in the associated sprite

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

- include/components/Rect.hpp

3.14 arcade::HUDText::rgb_s Struct Reference

rgb structure for text's color handling

```
#include <HUDText.hpp>
```

Public Attributes

- uint16_t **r**
- uint16_t **g**
- uint16_t **b**

3.14.1 Detailed Description

rgb structure for text's color handling

Parameters

r	for red color proportion
g	for green color proportion
b	for blue color proportion

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

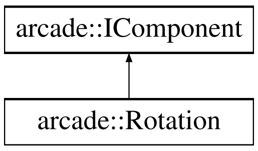
- include/components/HUDText.hpp

3.15 arcade::Rotation Class Reference

The Rotation component give the image rotation value in degree for graphical library with graphical interface.

```
#include <Rotation.hpp>
```

Inheritance diagram for arcade::Rotation:



Public Member Functions

- Rotation (float angle)

  Construct a new Rotation object.

Public Attributes

- float _angle

  angle rotation value in degree

3.15.1 Detailed Description

The Rotation component give the image rotation value in degree for graphical library with graphical interface.

3.15.2 Constructor & Destructor Documentation

3.15.2.1 Rotation()

```
arcade::Rotation::Rotation (
            float angle )  [inline]
```

Construct a new Rotation object.

Parameters

angle	angle rotation value in degree

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

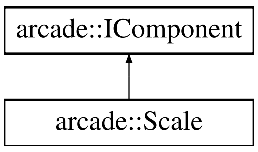
- include/components/Rotation.hpp

3.16 arcade::Scale Class Reference

The Scale component set width and height value of scale for updating the image size in graphical library with graphical interface.

```
#include <Scale.hpp>
```

Inheritance diagram for arcade::Scale:



Public Member Functions

- Scale (double width=0, double height=0)

  Construct a new Scale object.

Public Attributes

- double _width

  scale's width value for the associated sprite
- double _height

  scale's height value for the associated sprite

3.16.1 Detailed Description

The Scale component set width and height value of scale for updating the image size in graphical library with graphical interface.

Like Vector3D component, scale must use units (0, 1, 2...) for an easier handling with the position

3.16.2 Constructor & Destructor Documentation

3.16.2.1 Scale()

```
arcade::Scale::Scale (
            double width = 0,
            double height = 0 )  [inline]
```

Construct a new Scale object.

Parameters

width	scale's width value for the associated sprite
height	scale's height value for the associated sprite

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

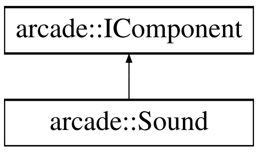
- include/components/Scale.hpp

3.17 arcade::Sound Class Reference

The Sound component contain a filepath for the sound in .wav and a SoundStatus enum to set the status of the Sound being either PLAY, PAUSE OR STOP.

`#include <Sound.hpp>`

Inheritance diagram for arcade::Sound:



Public Types

- enum SoundStatus_e { **PLAY**, **PAUSE**, **STOP** }

  enum used to describe the status of the sound
- typedef enum arcade::Sound::SoundStatus_e SoundStatus_t

  enum used to describe the status of the sound

Public Member Functions

- Sound (const std::string &path, SoundStatus_t status=SoundStatus_t::PLAY)

  Construct a new Sound object.

Public Attributes

- SoundStatus_t _status

  the sound's file path (in .wav)
- std::string _filepath

  the sound's status

3.17.1 Detailed Description

The Sound component contain a filepath for the sound in .wav and a SoundStatus enum to set the status of the Sound being either PLAY, PAUSE OR STOP.

3.17.2 Member Typedef Documentation

3.17.2.1 SoundStatus_t

```
typedef enum arcade::Sound::SoundStatus_e arcade::Sound::SoundStatus_t
```

enum used to describe the status of the sound

Parameters

PLAY	if the sound should be played
PAUSE	if the sound should be paused
STOP	if the sound should be stopped

3.17.3 Member Enumeration Documentation

3.17.3.1 SoundStatus_e

```
enum arcade::Sound::SoundStatus_e
```

enum used to describe the status of the sound

Parameters

PLAY	if the sound should be played
PAUSE	if the sound should be paused
STOP	if the sound should be stopped

3.17.4 Constructor & Destructor Documentation

3.17.4.1 Sound()

```
arcade::Sound::Sound (
            const std::string & path,
            SoundStatus_t status = SoundStatus_t::PLAY )  [inline]
```

Construct a new Sound object.

Parameters

path	the sound's file path (in .wav)
status	the sound's status

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

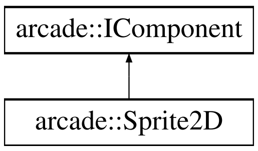
- include/components/Sound.hpp

3.18 arcade::Sprite2D Class Reference

The Sprite2D component is used by graphical libraries with a graphical interface (such as sfml, sdl2...) to draw the sprite according to the file given by the '_file' properties.

```
#include <Sprite2D.hpp>
```

Inheritance diagram for arcade::Sprite2D:



Public Member Functions

- Sprite2D (const std::string &file="")
  Construct a new Sprite 2D object.

Public Attributes

- std::string _file

    the image file path (in .bmp)

3.18.1 Detailed Description

The Sprite2D component is used by graphical libraries with a graphical interface (such as sfml, sdl2...) to draw the sprite according to the file given by the '_file' properties.

TO ENSURE A FUNCTIONNAL USE OF ALL FILE, IMAGE MUST BE IN .bmp FORMAT AN ENTITY MUST HAVE 2D AND TEXT VERSION TO ENSURE THAT ALL LIBRARIES CAN DREW IT

3.18.2 Constructor & Destructor Documentation

3.18.2.1 Sprite2D()

```
arcade::Sprite2D::Sprite2D (
            const std::string & file = "" )  [inline]
```

Construct a new Sprite 2D object.

Parameters

file	the image file path (in .bmp)

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

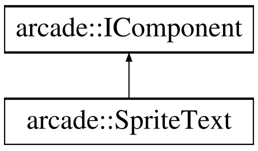
- include/components/Sprite2D.hpp

3.19 arcade::SpriteText Class Reference

The SpriteText component is used by graphical libraries with a non graphical interface (such as ncurses) to display the character according to the string given by the '_text' properties.

```
#include <SpriteText.hpp>
```

Inheritance diagram for arcade::SpriteText:



Public Member Functions

- SpriteText (const std::string &text="")

  Construct a new Sprite Text object.

Public Attributes

- std::string _text

  the character (or string) to display

3.19.1 Detailed Description

The SpriteText component is used by graphical libraries with a non graphical interface (such as ncurses) to display the character according to the string given by the '_text' properties.

AN ENTITY MUST HAVE 2D AND TEXT VERSION TO ENSURE THAT ALL LIBRARIES CAN DREW IT

3.19.2 Constructor & Destructor Documentation

3.19.2.1 SpriteText()

```
arcade::SpriteText::SpriteText (
            const std::string & text = "" )  [inline]
```

Construct a new Sprite Text object.

Parameters

text	the character (or string) to display

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

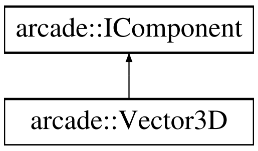
- include/components/SpriteText.hpp

3.20 arcade::Vector3D Class Reference

The Vector3D component give the position of the displayed component (Sprite2D/SpriteText/HUDText). The component contain '_x' and '_y' properties for the x and y position of the sprite in the window/terminal and a property '_z' for the layer position in the window.

```
#include <Vector3D.hpp>
```

Inheritance diagram for arcade::Vector3D:



Public Member Functions

- Vector3D (double x=0, double y=0, double z=0)

  Construct a new Vector 3 D object.

Public Attributes

- double _x

  the x position of the sprite/hudtext in the scene
- double _y

  the y position of the sprite/hudtext in the scene
- double _z

  the z (layer) position of the sprite/hudtext in the scene (In other word the bigger the z is, the higher the element will be displayed and on elements with lower z.)

3.20.1 Detailed Description

The Vector3D component give the position of the displayed component (Sprite2D/SpriteText/HUDText). The component contain '_x' and '_y' properties for the x and y position of the sprite in the window/terminal and a property '_z' for the layer position in the window.

x and y position in the architecture works as followed:

- Each entities will be considered as a one-by-one square.
- The entities position will consider that the scene is composed of cell of one-by-one so the positions will be like 0,0 or 0,1 or 1,0...
- If an entity need to be displayed between two cells, you can use float value. For example, an entity who need to be displayed between the 0,0 cell and 0,1 could have the position x=0 and y=0.5 This system is usefull for an easier handle of graphical library with non graphical interface such as ncurses

3.20.2 Constructor & Destructor Documentation

3.20.2.1 Vector3D()

```
arcade::Vector3D::Vector3D (
             double x = 0,
             double y = 0,
             double z = 0 )  [inline]
```

Construct a new Vector 3 D object.

Parameters

x	the x position of the sprite/hudtext in the scene
y	the y position of the sprite/hudtext in the scene
z	the z (layer) position of the sprite/hudtext in the scene (In other word the bigger the z is, the higher the element will be displayed and on elements with lower z.)

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

- include/components/Vector3D.hpp

Index