SDL2Me

Generated by Doxygen 1.8.5

Sun Jan 19 2014 03:09:23

Contents

Chapter 1

SDL2Me Tutorial

1.1 Introduction

SDL2Me, or S2M for short, is a library that uses SDL2 in order to bring a more familiar game development environment, implementing concepts such as Sprites, Cameras, Rooms, etc. It is inspired in the way Game Maker software (by YoYoGames) handles game programming.

1.2 Actual use example

This is an example of how to initialise the libraries and draw a simple Sprite on the screen:

```
#include "S2M.h"

// Create an Options instance by checking the config file specified.
// This will load all configurations into memory.
Options gOptions = Options("config.cfg");

// Create the Graphics instance based on the game screen size specified and the Options instance.
Graphics gGraphics = Graphics(320,240,"S2M Test",&Options);

// Create a Room and make it current.
Room gRoom = Room(); // Default black Room
gGraphics.setCurrentRoom(gRoom);

// Create a first and only Sprite.
Sprite *test = gGraphics.createSprite("test.bmp",16,16,0);
test->x = 0;
test->y = 0;

// Game Loop
while (true) {

    // Update the graphics
    gGraphics.update()
}
```

1.3 IDEAL use example

This is an example of how to initialise the libraries and draw a simple Sprite on the screen, following the desired patterns and main goals to accomplish of these libraries.

```
#include "S2M.h"

// Create an Options instance by checking the config file specified.
// This will load all configurations into memory.
Options gOptions = Options("config.cfg");

// Create the Graphics instance based on the game screen size specified and the Options instance.
Graphics gGraphics = Graphics(320,240,"S2M Test",&Options);
```

```
// Create a Room and make it current.
Room gRoom = Room(); // Default black Room
S2M_MakeCurrentRoom(gRoom);

// Create a first and only Sprite.
Sprite *testSprite = gGraphics.createSprite("test.bmp",16,16,0); // After this it still won't show in
      the screen.

// Create an object bound to that Sprite.
Object *testObject = gRoom.createObject(testSprite);

// Game Loop
while (true) {

    // Update the room
    S2M_UpdateRoom();   // Will call the current Room's update method
                        // The current Room's update method will call each object's update method.
                        // Each object's update method will update its associated sprite's position.
                        // The Graphics instance handles the Sprites update method, so no need for those
       calls here...

    // Update the graphics
    gGraphics.update()
}
```

1.3.1 Step 1: Opening the box

etc...

Chapter 2

page1

Chapter 3

Functions

Chapter 4

Namespace Index

4.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 5

Hierarchical Index

5.1 Class Hierarchy

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

Chapter 6

Class Index

6.1 Class List

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

Chapter 7

File Index

7.1 File List

Here is a list of all files with brief descriptions:

Chapter 8

Namespace Documentation

8.1 S2M_Room Namespace Reference

Every Room operation is included here.

Functions

- void AddBackground (Background *background)
- void LoadScript (string filename)
- void LoadScript ()

8.1.1 Detailed Description

Every Room operation is included here.

8.1.2 Function Documentation

8.1.2.1 void S2M_Room::AddBackground (Background * *background*)

8.1.2.2 void S2M_Room::LoadScript (string *filename*)

8.1.2.3 void S2M_Room::LoadScript ( )

8.2 S2M_Script Namespace Reference

Scripting operations are included here.

Functions

- vector< string > & SplitString (const string &s, char delim, vector< string > &elems)
- vector< string > SplitString (const string &s, char delim)
- string ReadFile (const char *filename)
- vector< string > FindAllStrings (string line)
- command ParseCommand (string line)
- vector< event > ParseFile (string filename)

8.2.1 Detailed Description

Scripting operations are included here.

8.2.2 Function Documentation

8.2.2.1 vector< string > S2M_Script::FindAllStrings (string *line*)

8.2.2.2 command S2M_Script::ParseCommand (string *line*)

8.2.2.3 vector< event > S2M_Script::ParseFile (string *filename*)

8.2.2.4 string S2M_Script::ReadFile (const char * *filename*)

8.2.2.5 vector< string > & S2M_Script::SplitString (const string & *s,* char *delim,* vector< string > & *elems*)

8.2.2.6 vector< string > S2M_Script::SplitString (const string & *s,* char *delim*)

Chapter 9

Class Documentation

9.1 Background Class Reference

A Room's background.

`#include <room.h>`

Public Member Functions

- Background (string filename, char s)

  The constructor.
- Background (string filename, char s, float xspe, float yspe)

  Another constructor.
- ~Background ()

  The destructor.
- int getWidth ()
- int getHeight ()
- void update ()

Public Attributes

- float x
- float y
- float xspeed
- float yspeed

Friends

- class Graphics

9.1.1 Detailed Description

A Room's background.

9.1.2 Constructor & Destructor Documentation

9.1.2.1 Background::Background (string *filename,* char *s*)

The constructor.

Parameters

filename	the image to load
style	

9.1.2.2 Background::Background (string *filename,* char *s,* float *xspe,* float *yspe*)

Another constructor.

9.1.2.3 Background::~Background ( )

The destructor.

Frees the texture and destroys the Background.

9.1.3 Member Function Documentation

9.1.3.1 int Background::getHeight ( )

9.1.3.2 int Background::getWidth ( )

9.1.3.3 void Background::update ( )

9.1.4 Friends And Related Function Documentation

9.1.4.1 friend class Graphics [friend]

9.1.5 Member Data Documentation

9.1.5.1 float Background::x

9.1.5.2 float Background::xspeed

9.1.5.3 float Background::y

9.1.5.4 float Background::yspeed

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

- src/room.h
- src/room.cpp

9.2 Camera Class Reference

Manages the Screen's viewport on a Room.

`#include <room.h>`

Public Member Functions

- Camera (Object *object)

  The constructor.
- Camera (int x, int y)

Another constructor.

- ~Camera ()

  The destructor.

- void setSpeed (int xspe, int yspe)

  Sets the Camera speed.

- void goTo (Object *object, bool smooth)

  Takes the camera on a smooth ride or instant leap towards the given object.

- void goTo (int x, int y, bool smooth)

  Takes the camera on a smooth ride or an instant leap towards the destination point.

- void move (int dx, int dy)

  DEBUG FUNCTION.

- void update ()

  The default update method.

Public Attributes

- float x

  Camera position variables.

- float y
- float xspeed
- float yspeed

Friends

- class Graphics
- class Background

9.2.1 Detailed Description

Manages the Screen's viewport on a Room.

9.2.2 Constructor & Destructor Documentation

9.2.2.1 Camera::Camera (Object * *object*)

The constructor.

The constructor takes the Object to follow as a parameter and follows it around.

Parameters

object	the Object to follow around
graphics	a pointer to a Graphics instance from where to get the game width and height

9.2.2.2 Camera::Camera (int *x*, int *y*)

Another constructor.

This constructor makes the Camera stay static on the given coordinates (top-left side of the screen). Useful when a sudden change in the viewport is needed.

Parameters

x	the horizontal position on Room of the Camera
y	the vertical position on Room of the Camera
graphics	a pointer to a Graphics instance from where to get the game width and height

9.2.2.3 Camera::~Camera ( )

The destructor.

9.2.3 Member Function Documentation

9.2.3.1 void Camera::goTo (Object * *object*, bool *smooth*)

Takes the camera on a smooth ride or instant leap towards the given object.

Won't stop until the object is centered on the screen.

Parameters

object	the object to go to.
smooth	whether you want a smooth travel to the given object or not (instant leap)

9.2.3.2 void Camera::goTo (int *x*, int *y*, bool *smooth*)

Takes the camera on a smooth ride or an instant leap towards the destination point.

Won't stop until the destination point is on the top-left side of the screen.

Parameters

x	the horizontal coordinate of the destination point
y	the vertical coordinate of the destination point
smooth	whether you want a smooth travel to the destination point or not (instant leap)

9.2.3.3 void Camera::move (int *dx*, int *dy*)

DEBUG FUNCTION.

9.2.3.4 void Camera::setSpeed (int *xspe*, int *yspe*)

Sets the Camera speed.

Parameters

xspe	horizontal speed
yspe	vertical speed

9.2.3.5 void Camera::update ( )

The default update method.

Overridable method. You can implement any path you would like your camera to follow as long as you modify the x and y coordinates whithin this method.

9.2.4 Friends And Related Function Documentation

9.2.4.1 friend class Background [friend]

9.2.4.2 friend class Graphics [friend]

9.2.5 Member Data Documentation

9.2.5.1 float Camera::x

Camera position variables.

9.2.5.2 float Camera::xspeed

9.2.5.3 float Camera::y

9.2.5.4 float Camera::yspeed

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

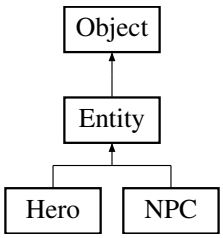
- src/room.h
- src/room.cpp

9.3 Entity Class Reference

Any Object that is affected by gravity or capable of random motion.

`#include <entity.h>`

Inheritance diagram for Entity:



Public Member Functions

- Entity (Sprite *sprite, float x, float y)

  Deafult constructor. Works just like it would work with an Object.
- ~Entity ()

  Deafult destructor. Works just like it would work with an Object.
- virtual void update ()

  Overridable update method.

Additional Inherited Members

9.3.1 Detailed Description

Any Object that is affected by gravity or capable of random motion.

Contrary to popular definition, an Entity is defined in S2M as any type of object (living, mechanic, inanimated) that is capable of being affected by gravity and having random motion at any given time of the game.

9.3.2 Constructor & Destructor Documentation

9.3.2.1 Entity::Entity (Sprite * *sprite,* float *x,* float *y*)

Deafult constructor. Works just like it would work with an Object.

9.3.2.2 Entity::~Entity ( )

Deafult destructor. Works just like it would work with an Object.

9.3.3 Member Function Documentation

9.3.3.1 void Entity::update ( ) [virtual]

Overridable update method.

If you derive the Entity class be sure to place

```
Entity::update();
```

on the new class' update method so it can implement the Entity default update method, which implements Object's as well.

Reimplemented from Object.

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

- src/platformer/entity.h
- src/platformer/entity.cpp

9.4 Graphics Class Reference

Controlls everything graphics-related.

`#include <graphics.h>`

Public Member Functions

- Graphics (int logWidth, int logHeight, string winCaption, Options *options)
  The constructor.
- ~Graphics ()
  The destructor.
- SDL_Texture * loadTexture (string filename, bool drawable)
  Loads a bitmap file and makes it an SDL_Texture.
- void drawTexture (SDL_Texture *source, int x, int y, int w, int h)
  Draws a texture directly on the screen.
- void blitTexture (SDL_Texture *tsrc, SDL_Texture *tdest, int x, int y, bool clear)
  Blits a source texture into a destination texture.
- void blitTexture (SDL_Texture *tsrc, SDL_Texture *tdest, int x, int y, int w, int h, bool clear)
  Blits a source texture into a destination texture.

- void blitTexture (SDL_Texture *tsrc, SDL_Texture *tdest, SDL_Rect *src, SDL_Rect *dest, bool clear)

  Blits a source texture into a destination texture.
- void update ()

  Flips the game screen and updates all instantiated Sprites.
- void addSprite (Sprite *sprite)

  Adds a Sprite to the sprites list.

Public Attributes

- SDL_Window * window

  The game window SDL_Window instance pointer.
- SDL_Renderer * renderer

  The game renderer SDL_Renderer instance pointer.
- int gameWidth
- int gameHeight

9.4.1 Detailed Description

Controlls everything graphics-related.

The Graphics class is an abstraction of everything graphic-based in the game. Every operation that involves graphics should need a pointer to this class.

9.4.2 Constructor & Destructor Documentation

9.4.2.1 Graphics::Graphics (int *logWidth*, int *logHeight*, string *winCaption*, Options * *options*)

The constructor.

Creates a new window and gives it a caption. Uses logical size instead of window size. What this means is that it will create a window based on the size you input times the scaling integer that the Options instance has, based on what you have specified in the config file.

Parameters

winWidth	the game width
winHeight	the game height
winCaption	the window caption
options	a pointer to an Options instance

See Also

~Graphics

9.4.2.2 Graphics::~Graphics ( )

The destructor.

Destroys the current window, frees all existing textures and by all means finishes the game.

See Also

Graphics

9.4.3 Member Function Documentation

9.4.3.1 void Graphics::addSprite (Sprite * *sprite*)

Adds a Sprite to the sprites list.

Parameters

sprite	the sprite pointer to add

9.4.3.2 void Graphics::blitTexture (SDL_Texture * *tsrc*, SDL_Texture * *tdest*, int *x*, int *y*, bool *clear*)

Blits a source texture into a destination texture.

Parameters

tsrc	the source texture
tdest	the destination texture
x	destination horizontal position
y	destination vertical position
clear	whether to clear or not the destination texture before blitting

9.4.3.3 void Graphics::blitTexture (SDL_Texture * *tsrc*, SDL_Texture * *tdest*, int *x*, int *y*, int *w*, int *h*, bool *clear*)

Blits a source texture into a destination texture.

This method lets you scale your texture.

Parameters

tsrc	the source texture
tdest	the destination texture
x	destination horizontal position
y	destination vertical position
w	width to scale the given texture to
h	height to scale the given texture to
clear	whether to clear or not the destination texture before blitting

9.4.3.4 void Graphics::blitTexture (SDL_Texture * *tsrc*, SDL_Texture * *tdest*, SDL_Rect * *src*, SDL_Rect * *dest*, bool *clear*)

Blits a source texture into a destination texture.

This method is a bit more complex but gives you access to cropping and scaling your textures.

Parameters

tsrc	the source texture
tdest	the destination texture
src	SDL_Rect of the source texture
dest	SDL_Rect of the destination texture
clear	whether to clear or not the destination texture before blitting

9.4.3.5 void Graphics::drawTexture (SDL_Texture * *source*, int *x*, int *y*, int *w*, int *h*)

Draws a texture directly on the screen.

This function draws a texture onscreen, bypassing viewports.

Parameters

source	the texture to blit
x	horizontal position on screen
y	vertical position on screen
w	width to scale the given texture to
h	height to scale the given texture to

9.4.3.6 SDL_Texture * Graphics::loadTexture (string *filename,* bool *drawable*)

Loads a bitmap file and makes it an SDL_Texture.

Parameters

filename	the bitmap to load (extension included)
drawable	indicates if the texture can or cannot be altered over time

Returns

Pointer to SDL_Texture representing the loaded bitmap

9.4.3.7 void Graphics::update ( )

Flips the game screen and updates all instantiated Sprites.

9.4.4 Member Data Documentation

9.4.4.1 int Graphics::gameHeight

9.4.4.2 int Graphics::gameWidth

9.4.4.3 SDL_Renderer* Graphics::renderer

The game renderer SDL_Renderer instance pointer.

9.4.4.4 SDL_Window* Graphics::window

The game window SDL_Window instance pointer.

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

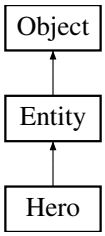
- src/graphics.h
- src/graphics.cpp

9.5 Hero Class Reference

An Entity that can be controlled directly with user input.

`#include <entity.h>`

Inheritance diagram for Hero:



Additional Inherited Members

9.5.1 Detailed Description

An Entity that can be controlled directly with user input.

A Hero is a type of Entity that can be controlled with an assigned Joystick. The default assigned Joystick is gJoystick, but you can assign any other VirtualJoystick if you want your Hero's movement to be script/code-determined.

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

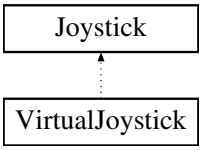
- src/platformer/entity.h

9.6 Joystick Class Reference

Representation of game buttons/keys.

`#include <joystick.h>`

Inheritance diagram for Joystick:



Public Member Functions

- bool getDir (int i)
- bool getBtn (int i)
- void update ()

9.6.1 Detailed Description

Representation of game buttons/keys.

A Joystick is a representation of the game buttons. It is useful if we bond it to a Hero instance: then it will follow the user's input. When we bond a VirtualJoystick to a Hero it will stop following our input and start following game compiled (or scripted) directives.

9.6.2 Member Function Documentation

9.6.2.1 bool Joystick::getBtn (int *i*)

9.6.2.2 bool Joystick::getDir (int *i*)

9.6.2.3 void Joystick::update ( )

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

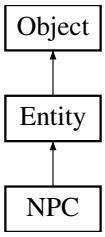
- src/joystick.h
- src/joystick.cpp

9.7 NPC Class Reference

An Entity that can't be directly controlled with user input.

`#include <entity.h>`

Inheritance diagram for NPC:



Additional Inherited Members

9.7.1 Detailed Description

An Entity that can't be directly controlled with user input.

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

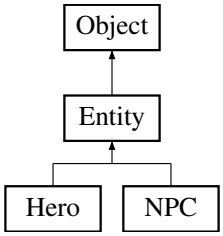
- src/platformer/entity.h

9.8 Object Class Reference

Anything that can be positioned within a Room.

`#include <object.h>`

Inheritance diagram for Object:



Public Member Functions

- Object (Sprite ∗sprite, float x, float y)

- ~Object ()
- void setAnimation (int i)
- int getAnimation ()
- void setDepth (char d)
- char getDepth ()
- virtual void update ()

Public Attributes

- float x
- float y
- Sprite * sprite

Friends

- class Graphics
- void Room::update ()

9.8.1 Detailed Description

Anything that can be positioned within a Room.

As long as it has x and y coordinates it is considered an object.

9.8.2 Constructor & Destructor Documentation

9.8.2.1 Object::Object (Sprite * *sprite*, float *x*, float *y*)

9.8.2.2 Object::~Object ( )

9.8.3 Member Function Documentation

9.8.3.1 int Object::getAnimation ( )

9.8.3.2 char Object::getDepth ( )

9.8.3.3 void Object::setAnimation (int *i*)

9.8.3.4 void Object::setDepth (char *d*)

9.8.3.5 void Object::update ( ) [virtual]

Reimplemented in Entity.

9.8.4 Friends And Related Function Documentation

9.8.4.1 friend class Graphics [friend]

9.8.4.2 void Room::update ( ) [friend]

9.8.5 Member Data Documentation

9.8.5.1 Sprite* Object::sprite

9.8.5.2 float Object::x

9.8.5.3 float Object::y

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

- src/object.h
- src/object.cpp

9.9 Options Class Reference

Serves as a deposit for all the game parameters.

`#include <options.h>`

Public Member Functions

- Options (string filename)

  The constructor.
- void setScale (int s)

  Adjusts the scaling parameter.
- int getScale ()

  Gets the scaling parameter.

9.9.1 Detailed Description

Serves as a deposit for all the game parameters.

Stores all the game options and adjustments and provides methods for setting and retrieving them.

9.9.2 Constructor & Destructor Documentation

9.9.2.1 Options::Options (string *filename*)

The constructor.

Parameters

filename	the config filename from where to load all the configurations

9.9.3 Member Function Documentation

9.9.3.1 int Options::getScale ( )

Gets the scaling parameter.

Returns

The scaling integer.

9.9.3.2 void Options::setScale (int *s*)

Adjusts the scaling parameter.

Parameters

s	the scale integer

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

- src/options.h
- src/options.cpp

9.10 Room Class Reference

An abstraction of a certain space within a game.

`#include <room.h>`

Public Member Functions

- Room (string filename, string scriptname)
  The constructor.
- Room ()
  Another constructor.
- Room (int w, int h)
  Yet another constructor.
- ~Room ()
  The destructor.
- int getWidth ()
- int getHeight ()
- void setCamera (Camera *c)
  Sets the pointer to the default Camera instance.
- Object * createObject (Sprite *sprite, float x, float y)
- void addObject (Object *object)
- virtual void update ()
  The update method.

Public Attributes

- Camera * camera
  The default camera.

Friends

- class Graphics
- bool compareObjectsByDepth (Object *obj1, Object *obj2)
  A function that sorts Sprites by its depth.
- void S2M_Room::AddBackground (Background *background)
- void S2M_Room::LoadScript (string filename)
- void S2M_Room::LoadScript ()

9.10.1 Detailed Description

An abstraction of a certain space within a game.

Rooms are a very abstract concept, much like objects. They represent a certain part of the game, like a level, the game menu or the rolling credits screen.

9.10.2 Constructor & Destructor Documentation

9.10.2.1 Room::Room (string *filename*, string *scriptname*)

The constructor.

Creates a Room from a file.

Parameters

filename	the design file
scriptname	the script file

9.10.2.2 Room::Room ( )

Another constructor.

Creates an empty black Room.

9.10.2.3 Room::Room (int *w*, int *h*)

Yet another constructor.

Creates an empty black Room with the specified dimensions

Parameters

w	the Room's width
h	the Rooms height

9.10.2.4 Room::~Room ( )

The destructor.

Destroys the Room and all its associated tile textures

9.10.3 Member Function Documentation

9.10.3.1 void Room::addObject (Object * *object*)

9.10.3.2 Object * Room::createObject (Sprite * *sprite*, float *x*, float *y*)

9.10.3.3 int Room::getHeight ( )

9.10.3.4 int Room::getWidth ( )

9.10.3.5 void Room::setCamera (Camera * *c*)

Sets the pointer to the default Camera instance.

The Graphics instance needs a Camera instance pointer in order to know at any moment which part of the Room to draw on the screen.

Parameters

camera	a pointer to a Camera instance

9.10.3.6 void Room::update ( ) [virtual]

The update method.

9.10.4 Friends And Related Function Documentation

9.10.4.1 bool compareObjectsByDepth (Object * *obj1*, Object * *obj2*) [friend]

A function that sorts Sprites by its depth.

9.10.4.2 friend class Graphics [friend]

9.10.4.3 void S2M_Room::AddBackground (Background * *background*) [friend]

9.10.4.4 void S2M_Room::LoadScript (string *filename*) [friend]

9.10.4.5 void S2M_Room::LoadScript ( ) [friend]

9.10.5 Member Data Documentation

9.10.5.1 Camera* Room::camera

The default camera.

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

- src/room.h
- src/room.cpp

9.11 Sprite Class Reference

A simple abstraction of a set of images and animations.

`#include <graphics.h>`

Public Member Functions

- Sprite (string filename, int w, int h)

  The constructor.
- ~Sprite ()

  The destructor.
- int getWidth ()

  Returns the current image of the Sprite.
- int getHeight ()

  Returns the Sprite's height.
- SDL_Rect * getRect (int i)

  Gets the given frame's SDL_Rect on the Sprite texture.
- void setFrame (int i)

Sets the current frame.

- int getFrame ()

  Gets the current frame number of the Sprite.
- void addAnimation (vector< int > animation)

  Adds an animation to the animations vector.
- vector< int > retAnimation (int i)

  Retrieves an animation from the animations vector.
- int getAnimationsSize ()

  Sets the current animation.
- void addObject (Object *object)
- const SDL_Texture & operator[] (const int idx)

  Operator overrider to get a frame from a Sprite.
- void update ()

  Updates the Sprite image according to animation and animation speed.

Friends

- void Graphics::update ()

  Only the Graphics and Object classes update method can change the Sprite's textures.
- void Object::update ()

9.11.1 Detailed Description

A simple abstraction of a set of images and animations.

A Sprite instance must be instantiated by each object who wants to have a screen representation.

9.11.2 Constructor & Destructor Documentation

9.11.2.1 Sprite::Sprite (string *filename*, int *w*, int *h*)

The constructor.

Parameters

filename	the bitmap filename where all the sprite images are to be loaded
w	the width of each image
h	the height of each image

9.11.2.2 Sprite::~Sprite ( )

The destructor.

Destroys the Sprite and frees its textures.

9.11.3 Member Function Documentation

9.11.3.1 void Sprite::addAnimation (vector< int > *animation*)

Adds an animation to the animations vector.

Each Sprite has a vector of animations, which are nothing more than image numbers arranged by order. They play in sequence and in the established order.

Parameters

animation	a vector of numbers that represent frames

9.11.3.2 void Sprite::addObject (Object * *object*)

9.11.3.3 int Sprite::getAnimationsSize ( )

Sets the current animation.

Each Sprite has a vector of animations, which are nothing more than image numbers arranged by order. They play in sequence and in the established order.

Parameters

i	the animation indexGets the current animation.

Returns a vector that represents the current animation.

Returns

The animation index

9.11.3.4 int Sprite::getFrame ( )

Gets the current frame number of the Sprite.

This method returns the current image number.

Returns

The current image number

9.11.3.5 int Sprite::getHeight ( )

Returns the Sprite's height.

/return The Sprite's height.

9.11.3.6 SDL_Rect * Sprite::getRect (int *i*)

Gets the given frame's SDL_Rect on the Sprite texture.

Parameters

i	the frame number

Returns

A pointer to an SDL_Rect structure.

9.11.3.7 int Sprite::getWidth ( )

Returns the current image of the Sprite.

Returns

The current image of this SpriteReturns the Sprite's width.

/return The Sprite's width.

9.11.3.8 const SDL_Texture & Sprite::operator[] (const int *idx*)

Operator overrider to get a frame from a Sprite.

Overrides the [] operator so that it returns the frame that represents the given number.

Parameters

idx	the given number

Returns

The frame that represents the given number.

9.11.3.9 vector< int > Sprite::retAnimation (int *i*)

Retrieves an animation from the animations vector.

Parameters

i	the animation index

Returns

The animation on the given index

9.11.3.10 void Sprite::setFrame (int *i*)

Sets the current frame.

Each image of a Sprite has an id number from 0 to n. This method sets the current image number. The change in textures takes place immediately.

Parameters

i	the frame number

9.11.3.11 void Sprite::update ( )

Updates the Sprite image according to animation and animation speed.

9.11.4 Friends And Related Function Documentation

9.11.4.1 void Graphics::update ( ) [friend]

Only the Graphics and Object classes update method can change the Sprite's textures.

9.11.4.2 void Object::update ( ) [friend]

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

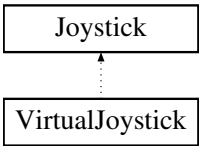
- src/graphics.h
- src/graphics.cpp

9.12 VirtualJoystick Class Reference

Virtual representation of game buttons/keys.

`#include <joystick.h>`

Inheritance diagram for VirtualJoystick:



9.12.1 Detailed Description

Virtual representation of game buttons/keys.

Careful, because this may be confusing: A VirtualJoystick is a virtual representation of the game buttons. It is useful whenever we want to make an Entity move, because assigning a Joystick to it (if it doesn't already have one, which is probable since it is created by the constructor) we can code the Joystick's buttons pressed and, therefore, make the Entity move.

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

- src/joystick.h

Chapter 10

File Documentation

10.1 src/graphics.cpp File Reference

```
#include "graphics.h"
```

Functions

- Graphics * S2M_CreateGraphics ()
- void S2M_UpdateGraphics ()
- Sprite * S2M_CreateSprite (string filename, int w, int h)

  Creates a Sprite and registers it on the Graphics sprite vector.

Variables

- Graphics * gGraphics

  Global instance of Graphics.
- Options * gOptions

10.1.1 Function Documentation

10.1.1.1 Graphics* S2M_CreateGraphics ( )

10.1.1.2 Sprite* S2M_CreateSprite (string *filename*, int *w*, int *h*)

Creates a Sprite and registers it on the Graphics sprite vector.

Parameters

filename	the bitmap filename where all the sprite images are to be loaded
w	the width of each image
h	the height of each image

Returns

A pointer to the recently created Sprite.

10.1.1.3 void S2M_UpdateGraphics ( )

10.1.2 Variable Documentation

10.1.2.1 Graphics* gGraphics

Global instance of Graphics.

Yeah, yeah, I know it's bad to have globals, but whatever, I'm just a C++ noob.

10.1.2.2 Options* gOptions

10.2 src/graphics.h File Reference

```
#include <string>
#include <vector>
#include <algorithm>
#include <iostream>
#include <cmath>
#include <SDL2/SDL.h>
#include <SDL2/SDL_image.h>
#include "Easings/Expo.h"
#include "options.h"
#include "object.h"
```

Classes

- class Graphics

  Controlls everything graphics-related.
- class Sprite

  A simple abstraction of a set of images and animations.

Functions

- Graphics * S2M_CreateGraphics ()
- void S2M_UpdateGraphics ()
- Sprite * S2M_CreateSprite (string filename, int w, int h)

  Creates a Sprite and registers it on the Graphics sprite vector.

Variables

- Graphics * gGraphics

  Global instance of Graphics.
- Options * gOptions

10.2.1 Function Documentation

10.2.1.1 Graphics* S2M_CreateGraphics ( )

10.2.1.2 Sprite* S2M_CreateSprite (string *filename*, int *w*, int *h*)

Creates a Sprite and registers it on the Graphics sprite vector.

Parameters

filename	the bitmap filename where all the sprite images are to be loaded
w	the width of each image
h	the height of each image

Returns

A pointer to the recently created Sprite.

10.2.1.3 void S2M_UpdateGraphics ( )

10.2.2 Variable Documentation

10.2.2.1 Graphics* gGraphics

Global instance of Graphics.

Yeah, yeah, I know it's bad to have globals, but whatever, I'm just a C++ noob.

10.2.2.2 Options* gOptions

10.3 src/joystick.cpp File Reference

```
#include "joystick.h"
```

Variables

- Joystick * gJoystick

  Global Joystick pointer.

10.3.1 Variable Documentation

10.3.1.1 Joystick* gJoystick

Global Joystick pointer.

10.4 src/joystick.h File Reference

```
#include <SDL2/SDL.h>
```

Classes

- class Joystick

  Representation of game buttons/keys.
- class VirtualJoystick

  Virtual representation of game buttons/keys.

Variables

- Joystick * gJoystick

  Global Joystick pointer.

10.4.1 Variable Documentation

10.4.1.1 Joystick* gJoystick

Global Joystick pointer.

10.5 src/object.cpp File Reference

```
#include "object.h"
#include "graphics.h"
```

10.6 src/object.h File Reference

```
#include "room.h"
```

Classes

- class Object

  Anything that can be positioned within a Room.

Functions

- bool compareObjectsByDepth (Object *obj1, Object *obj2)

10.6.1 Function Documentation

10.6.1.1 bool compareObjectsByDepth (Object * *obj1*, Object * *obj2*)

10.7 src/options.cpp File Reference

```
#include "options.h"
```

10.8 src/options.h File Reference

```
#include <string>
```

Classes

- class Options

  Serves as a deposit for all the game parameters.

10.9 src/pause.cpp File Reference

```
#include "pause.h"
```

Functions

- void S2M_PauseGame ()
- void S2M_UnpauseGame ()

Variables

- bool gameInPause = false

10.9.1 Function Documentation

10.9.1.1 void S2M_PauseGame ( )

10.9.1.2 void S2M_UnpauseGame ( )

10.9.2 Variable Documentation

10.9.2.1 bool gameInPause = false

10.10 src/pause.h File Reference

Functions

- void S2M_PauseGame ()
- void S2M_UnpauseGame ()

Variables

- bool gameInPause

10.10.1 Function Documentation

10.10.1.1 void S2M_PauseGame ( )

10.10.1.2 void S2M_UnpauseGame ( )

10.10.2 Variable Documentation

10.10.2.1 bool gameInPause

10.11 src/platformer/entity.cpp File Reference

```
#include "entity.h"
#include "../room.h"
#include "../graphics.h"
#include <iostream>
#include <algorithm>
```

Functions

- Entity * S2M_CreateEntity (Sprite *sprite, float x, float y)

10.11.1 Function Documentation

10.11.1.1 Entity* S2M_CreateEntity (Sprite * *sprite,* float *x,* float *y*)

10.12 src/platformer/entity.h File Reference

```
#include "../object.h"
#include "../joystick.h"
```

Classes

- class Entity

  Any Object that is affected by gravity or capable of random motion.
- class Hero

  An Entity that can be controlled directly with user input.
- class NPC

  An Entity that can't be directly controlled with user input.

Functions

- Entity * S2M_CreateEntity (Sprite *sprite, float x, float y)

10.12.1 Function Documentation

10.12.1.1 Entity* S2M_CreateEntity (Sprite * *sprite,* float *x,* float *y*)

10.13 src/platformer/hero.h File Reference

10.14 src/room.cpp File Reference

```
#include "room.h"
#include "object.h"
#include "graphics.h"
#include "pause.h"
#include "script.h"
```

Functions

- Room * S2M_CreateRoom ()

  Create an empty Room.
- Room * S2M_CreateRoom (int w, int h)
- void S2M_SetRoom (Room *room)
- void S2M_UpdateRoom ()
- bool compareObjectsByDepth (Object *obj1, Object *obj2)

Variables

- Room * gRoom

10.14.1 Function Documentation

10.14.1.1 bool compareObjectsByDepth (Object * *obj1*, Object * *obj2*)

10.14.1.2 Room* S2M_CreateRoom ( )

Create an empty Room.

10.14.1.3 Room* S2M_CreateRoom (int *w*, int *h*)

10.14.1.4 void S2M_SetRoom (Room * *room*)

10.14.1.5 void S2M_UpdateRoom ( )

10.14.2 Variable Documentation

10.14.2.1 Room* gRoom

10.15 src/room.h File Reference

```
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>
#include <cmath>
#include <SDL2/SDL.h>
#include "RapidXML/rapidxml.hpp"
```

Classes

- class Room

  An abstraction of a certain space within a game.
- class Background

  A Room's background.
- class Camera

  Manages the Screen's viewport on a Room.

Namespaces

- S2M_Room

  Every Room operation is included here.

Macros

- #define STYLE_FILL char(0)
- #define STYLE_MOSAIC char(1)
- #define STYLE_CENTER char(2)
- #define STYLE_PARALLAX char(3)
- #define STYLE_MOSAIC_PARALLAX char(4)
- #define STYLE_STATIC char(5)

Functions

- bool compareObjectsByDepth (Object *obj1, Object *obj2)
- void S2M_Room::AddBackground (Background *background)
- void S2M_Room::LoadScript (string filename)
- void S2M_Room::LoadScript ()
- Room * S2M_CreateRoom ()

  Create an empty Room.
- Room * S2M_CreateRoom (int w, int h)
- void S2M_SetRoom (Room *room)
- void S2M_UpdateRoom ()

Variables

- Room * gRoom

10.15.1 Macro Definition Documentation

10.15.1.1 #define STYLE_CENTER char(2)

10.15.1.2 #define STYLE_FILL char(0)

10.15.1.3 #define STYLE_MOSAIC char(1)

10.15.1.4 #define STYLE_MOSAIC_PARALLAX char(4)

10.15.1.5 #define STYLE_PARALLAX char(3)

10.15.1.6 #define STYLE_STATIC char(5)

10.15.2 Function Documentation

10.15.2.1 bool compareObjectsByDepth (Object * *obj1*, Object * *obj2*)

10.15.2.2 Room* S2M_CreateRoom ( )

Create an empty Room.

10.15.2.3 Room* S2M_CreateRoom (int *w*, int *h*)

10.15.2.4 void S2M_SetRoom (Room * *room*)

10.15.2.5 void S2M_UpdateRoom ( )

10.15.3 Variable Documentation

10.15.3.1 Room* gRoom

10.16 src/S2M.h File Reference

```
#include <SDL2/SDL.h>
#include <SDL2/SDL_image.h>
#include "graphics.h"
#include "options.h"
#include "object.h"
#include "room.h"
#include "pause.h"
#include "script.h"
```

Functions

- void S2M_Update ()

10.16.1 Function Documentation

10.16.1.1 void S2M_Update ( )

10.17 src/S2M_Platformer.h File Reference

```
#include "platformer/entity.h"
```

10.18 src/script.cpp File Reference

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <iomanip>
#include <stdlib.h>
#include "script.h"
```

Namespaces

- S2M_Script

  Scripting operations are included here.

Functions

- vector< string > & S2M_Script::SplitString (const string &s, char delim, vector< string > &elems)
- vector< string > S2M_Script::SplitString (const string &s, char delim)
- string S2M_Script::ReadFile (const char *filename)
- vector< string > S2M_Script::FindAllStrings (string line)
- command S2M_Script::ParseCommand (string line)
- vector< event > S2M_Script::ParseFile (string filename)

Variables

- map< string, unsigned char > itnMap

10.18.1 Variable Documentation

10.18.1.1 map<string, unsigned char> itnMap

Initial value:

```
{
    {"lock", 0},
    {"faceplayer", 1},
    {"message", 2},
    {"nod", 3},
    {"transport", 4}
}
```

10.19 src/script.h File Reference

```
#include <string>
#include <map>
#include <vector>
```

Namespaces

- S2M_Script

  Scripting operations are included here.

Functions

- vector< string > & S2M_Script::SplitString (const string &s, char delim, vector< string > &elems)
- vector< string > S2M_Script::SplitString (const string &s, char delim)
- string S2M_Script::ReadFile (const char *filename)
- vector< string > S2M_Script::FindAllStrings (string line)
- command S2M_Script::ParseCommand (string line)
- vector< event > S2M_Script::ParseFile (string filename)

Variables

- map< string, unsigned char > itnMap

10.19.1 Variable Documentation

10.19.1.1 map<string, unsigned char> itnMap

Index