

SDL2Me

Generated by Doxygen 1.8.5

Sun Jan 19 2014 03:09:23



# Contents

<b>1</b>	<b>SDL2Me Tutorial</b>	<b>1</b>
1.1	Introduction . . . . .	1
1.2	Actual use example . . . . .	1
1.3	IDEAL use example . . . . .	1
1.3.1	Step 1: Opening the box . . . . .	2
<b>2</b>	<b>page1</b>	<b>3</b>
<b>3</b>	<b>Functions</b>	<b>5</b>
<b>4</b>	<b>Namespace Index</b>	<b>7</b>
4.1	Namespace List . . . . .	7
<b>5</b>	<b>Hierarchical Index</b>	<b>9</b>
5.1	Class Hierarchy . . . . .	9
<b>6</b>	<b>Class Index</b>	<b>11</b>
6.1	Class List . . . . .	11
<b>7</b>	<b>File Index</b>	<b>13</b>
7.1	File List . . . . .	13
<b>8</b>	<b>Namespace Documentation</b>	<b>15</b>
8.1	S2M_Room Namespace Reference . . . . .	15
8.1.1	Detailed Description . . . . .	15
8.1.2	Function Documentation . . . . .	15
8.1.2.1	AddBackground . . . . .	15
8.1.2.2	LoadScript . . . . .	15
8.1.2.3	LoadScript . . . . .	15
8.2	S2M_Script Namespace Reference . . . . .	15
8.2.1	Detailed Description . . . . .	16
8.2.2	Function Documentation . . . . .	16
8.2.2.1	FindAllStrings . . . . .	16
8.2.2.2	ParseCommand . . . . .	16

8.2.2.3	ParseFile	16
8.2.2.4	ReadFile	16
8.2.2.5	SplitString	16
8.2.2.6	SplitString	16
<b>9</b>	<b>Class Documentation</b>	<b>17</b>
9.1	Background Class Reference	17
9.1.1	Detailed Description	17
9.1.2	Constructor & Destructor Documentation	18
9.1.2.1	Background	18
9.1.2.2	Background	19
9.1.2.3	~Background	19
9.1.3	Member Function Documentation	19
9.1.3.1	getHeight	19
9.1.3.2	getWidth	19
9.1.3.3	update	19
9.1.4	Friends And Related Function Documentation	19
9.1.4.1	Graphics	19
9.1.5	Member Data Documentation	19
9.1.5.1	x	19
9.1.5.2	xspeed	19
9.1.5.3	y	19
9.1.5.4	yspeed	19
9.2	Camera Class Reference	19
9.2.1	Detailed Description	20
9.2.2	Constructor & Destructor Documentation	20
9.2.2.1	Camera	20
9.2.2.2	Camera	20
9.2.2.3	~Camera	21
9.2.3	Member Function Documentation	21
9.2.3.1	goTo	21
9.2.3.2	goTo	21
9.2.3.3	move	21
9.2.3.4	setSpeed	21
9.2.3.5	update	21
9.2.4	Friends And Related Function Documentation	22
9.2.4.1	Background	22
9.2.4.2	Graphics	22
9.2.5	Member Data Documentation	22
9.2.5.1	x	22

9.2.5.2	<a href="#">xspeed</a>	22
9.2.5.3	<a href="#">y</a>	22
9.2.5.4	<a href="#">yspeed</a>	22
9.3	<a href="#">Entity Class Reference</a>	22
9.3.1	<a href="#">Detailed Description</a>	22
9.3.2	<a href="#">Constructor &amp; Destructor Documentation</a>	23
9.3.2.1	<a href="#">Entity</a>	23
9.3.2.2	<a href="#">~Entity</a>	23
9.3.3	<a href="#">Member Function Documentation</a>	23
9.3.3.1	<a href="#">update</a>	23
9.4	<a href="#">Graphics Class Reference</a>	23
9.4.1	<a href="#">Detailed Description</a>	24
9.4.2	<a href="#">Constructor &amp; Destructor Documentation</a>	24
9.4.2.1	<a href="#">Graphics</a>	24
9.4.2.2	<a href="#">~Graphics</a>	24
9.4.3	<a href="#">Member Function Documentation</a>	25
9.4.3.1	<a href="#">addSprite</a>	25
9.4.3.2	<a href="#">blitTexture</a>	26
9.4.3.3	<a href="#">blitTexture</a>	26
9.4.3.4	<a href="#">blitTexture</a>	26
9.4.3.5	<a href="#">drawTexture</a>	26
9.4.3.6	<a href="#">loadTexture</a>	27
9.4.3.7	<a href="#">update</a>	27
9.4.4	<a href="#">Member Data Documentation</a>	27
9.4.4.1	<a href="#">gameHeight</a>	27
9.4.4.2	<a href="#">gameWidth</a>	27
9.4.4.3	<a href="#">renderer</a>	27
9.4.4.4	<a href="#">window</a>	27
9.5	<a href="#">Hero Class Reference</a>	27
9.5.1	<a href="#">Detailed Description</a>	28
9.6	<a href="#">Joystick Class Reference</a>	28
9.6.1	<a href="#">Detailed Description</a>	28
9.6.2	<a href="#">Member Function Documentation</a>	28
9.6.2.1	<a href="#">getBtn</a>	28
9.6.2.2	<a href="#">getDir</a>	29
9.6.2.3	<a href="#">update</a>	29
9.7	<a href="#">NPC Class Reference</a>	29
9.7.1	<a href="#">Detailed Description</a>	29
9.8	<a href="#">Object Class Reference</a>	29
9.8.1	<a href="#">Detailed Description</a>	30

9.8.2	Constructor & Destructor Documentation	30
9.8.2.1	Object	30
9.8.2.2	~Object	30
9.8.3	Member Function Documentation	30
9.8.3.1	getAnimation	30
9.8.3.2	getDepth	30
9.8.3.3	setAnimation	30
9.8.3.4	setDepth	30
9.8.3.5	update	30
9.8.4	Friends And Related Function Documentation	30
9.8.4.1	Graphics	30
9.8.4.2	Room::update	30
9.8.5	Member Data Documentation	30
9.8.5.1	sprite	31
9.8.5.2	x	31
9.8.5.3	y	31
9.9	Options Class Reference	31
9.9.1	Detailed Description	31
9.9.2	Constructor & Destructor Documentation	31
9.9.2.1	Options	31
9.9.3	Member Function Documentation	31
9.9.3.1	getScale	31
9.9.3.2	setScale	32
9.10	Room Class Reference	33
9.10.1	Detailed Description	33
9.10.2	Constructor & Destructor Documentation	34
9.10.2.1	Room	34
9.10.2.2	Room	34
9.10.2.3	Room	34
9.10.2.4	~Room	34
9.10.3	Member Function Documentation	34
9.10.3.1	addObject	34
9.10.3.2	createObject	34
9.10.3.3	getHeight	34
9.10.3.4	getWidth	34
9.10.3.5	setCamera	34
9.10.3.6	update	35
9.10.4	Friends And Related Function Documentation	35
9.10.4.1	compareObjectsByDepth	35
9.10.4.2	Graphics	35

9.10.4.3	S2M_Room::AddBackground	35
9.10.4.4	S2M_Room::LoadScript	35
9.10.4.5	S2M_Room::LoadScript	35
9.10.5	Member Data Documentation	35
9.10.5.1	camera	35
9.11	Sprite Class Reference	35
9.11.1	Detailed Description	36
9.11.2	Constructor & Destructor Documentation	36
9.11.2.1	Sprite	36
9.11.2.2	~Sprite	36
9.11.3	Member Function Documentation	36
9.11.3.1	addAnimation	36
9.11.3.2	addObject	37
9.11.3.3	getAnimationsSize	37
9.11.3.4	getFrame	37
9.11.3.5	getHeight	37
9.11.3.6	getRect	37
9.11.3.7	getWidth	37
9.11.3.8	operator[]	38
9.11.3.9	retAnimation	38
9.11.3.10	setFrame	38
9.11.3.11	update	38
9.11.4	Friends And Related Function Documentation	38
9.11.4.1	Graphics::update	38
9.11.4.2	Object::update	38
9.12	VirtualJoystick Class Reference	39
9.12.1	Detailed Description	39
<b>10</b>	<b>File Documentation</b>	<b>41</b>
10.1	src/graphics.cpp File Reference	41
10.1.1	Function Documentation	41
10.1.1.1	S2M_CreateGraphics	41
10.1.1.2	S2M_CreateSprite	41
10.1.1.3	S2M_UpdateGraphics	42
10.1.2	Variable Documentation	42
10.1.2.1	gGraphics	42
10.1.2.2	gOptions	42
10.2	src/graphics.h File Reference	42
10.2.1	Function Documentation	42
10.2.1.1	S2M_CreateGraphics	42

10.2.1.2	S2M_CreateSprite	42
10.2.1.3	S2M_UpdateGraphics	43
10.2.2	Variable Documentation	43
10.2.2.1	gGraphics	43
10.2.2.2	gOptions	43
10.3	src/joystick.cpp File Reference	43
10.3.1	Variable Documentation	43
10.3.1.1	gJoystick	43
10.4	src/joystick.h File Reference	43
10.4.1	Variable Documentation	44
10.4.1.1	gJoystick	44
10.5	src/object.cpp File Reference	44
10.6	src/object.h File Reference	44
10.6.1	Function Documentation	44
10.6.1.1	compareObjectsByDepth	44
10.7	src/options.cpp File Reference	44
10.8	src/options.h File Reference	44
10.9	src/pause.cpp File Reference	45
10.9.1	Function Documentation	45
10.9.1.1	S2M_PauseGame	45
10.9.1.2	S2M_UnpauseGame	45
10.9.2	Variable Documentation	45
10.9.2.1	gameInPause	45
10.10	src/pause.h File Reference	45
10.10.1	Function Documentation	45
10.10.1.1	S2M_PauseGame	45
10.10.1.2	S2M_UnpauseGame	45
10.10.2	Variable Documentation	45
10.10.2.1	gameInPause	45
10.11	src/platformer/entity.cpp File Reference	46
10.11.1	Function Documentation	46
10.11.1.1	S2M_CreateEntity	46
10.12	src/platformer/entity.h File Reference	46
10.12.1	Function Documentation	46
10.12.1.1	S2M_CreateEntity	46
10.13	src/platformer/hero.h File Reference	46
10.14	src/room.cpp File Reference	46
10.14.1	Function Documentation	47
10.14.1.1	compareObjectsByDepth	47
10.14.1.2	S2M_CreateRoom	47



10.14.1.3 S2M_CreateRoom . . . . .	47
10.14.1.4 S2M_SetRoom . . . . .	47
10.14.1.5 S2M_UpdateRoom . . . . .	47
10.14.2 Variable Documentation . . . . .	47
10.14.2.1 gRoom . . . . .	47
10.15src/room.h File Reference . . . . .	47
10.15.1 Macro Definition Documentation . . . . .	48
10.15.1.1 STYLE_CENTER . . . . .	48
10.15.1.2 STYLE_FILL . . . . .	48
10.15.1.3 STYLE_MOSAIC . . . . .	48
10.15.1.4 STYLE_MOSAIC_PARALLAX . . . . .	48
10.15.1.5 STYLE_PARALLAX . . . . .	48
10.15.1.6 STYLE_STATIC . . . . .	48
10.15.2 Function Documentation . . . . .	48
10.15.2.1 compareObjectsByDepth . . . . .	48
10.15.2.2 S2M_CreateRoom . . . . .	48
10.15.2.3 S2M_CreateRoom . . . . .	49
10.15.2.4 S2M_SetRoom . . . . .	49
10.15.2.5 S2M_UpdateRoom . . . . .	49
10.15.3 Variable Documentation . . . . .	49
10.15.3.1 gRoom . . . . .	49
10.16src/S2M.h File Reference . . . . .	49
10.16.1 Function Documentation . . . . .	49
10.16.1.1 S2M_Update . . . . .	49
10.17src/S2M_Platformer.h File Reference . . . . .	49
10.18src/script.cpp File Reference . . . . .	49
10.18.1 Variable Documentation . . . . .	50
10.18.1.1 itnMap . . . . .	50
10.19src/script.h File Reference . . . . .	50
10.19.1 Variable Documentation . . . . .	51
10.19.1.1 itnMap . . . . .	51



# 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:

<a href="#">S2M_Room</a>	Every <a href="#">Room</a> operation is included here . . . . .	<a href="#">15</a>
<a href="#">S2M_Script</a>	Scripting operations are included here . . . . .	<a href="#">15</a>



## Chapter 5

# Hierarchical Index

### 5.1 Class Hierarchy

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

Background . . . . .	17
Camera . . . . .	19
Graphics . . . . .	23
Joystick . . . . .	28
VirtualJoystick . . . . .	39
Object . . . . .	29
Entity . . . . .	22
Hero . . . . .	27
NPC . . . . .	29
Options . . . . .	31
Room . . . . .	33
Sprite . . . . .	35



## Chapter 6

# Class Index

### 6.1 Class List

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

<a href="#">Background</a>	A <a href="#">Room</a> 's background . . . . .	17
<a href="#">Camera</a>	Manages the Screen's viewport on a <a href="#">Room</a> . . . . .	19
<a href="#">Entity</a>	Any <a href="#">Object</a> that is affected by gravity or capable of random motion . . . . .	22
<a href="#">Graphics</a>	Controls everything graphics-related . . . . .	23
<a href="#">Hero</a>	An <a href="#">Entity</a> that can be controlled directly with user input . . . . .	27
<a href="#">Joystick</a>	Representation of game buttons/keys . . . . .	28
<a href="#">NPC</a>	An <a href="#">Entity</a> that can't be directly controlled with user input . . . . .	29
<a href="#">Object</a>	Anything that can be positioned within a <a href="#">Room</a> . . . . .	29
<a href="#">Options</a>	Serves as a deposit for all the game parameters . . . . .	31
<a href="#">Room</a>	An abstraction of a certain space within a game . . . . .	33
<a href="#">Sprite</a>	A simple abstraction of a set of images and animations . . . . .	35
<a href="#">VirtualJoystick</a>	Virtual representation of game buttons/keys . . . . .	39



## Chapter 7

# File Index

### 7.1 File List

Here is a list of all files with brief descriptions:

src/ <a href="#">graphics.cpp</a>	41
src/ <a href="#">graphics.h</a>	42
src/ <a href="#">joystick.cpp</a>	43
src/ <a href="#">joystick.h</a>	43
src/ <a href="#">object.cpp</a>	44
src/ <a href="#">object.h</a>	44
src/ <a href="#">options.cpp</a>	44
src/ <a href="#">options.h</a>	44
src/ <a href="#">pause.cpp</a>	45
src/ <a href="#">pause.h</a>	45
src/ <a href="#">room.cpp</a>	46
src/ <a href="#">room.h</a>	47
src/ <a href="#">S2M.h</a>	49
src/ <a href="#">S2M_Platformer.h</a>	49
src/ <a href="#">script.cpp</a>	49
src/ <a href="#">script.h</a>	50
src/platformer/ <a href="#">entity.cpp</a>	46
src/platformer/ <a href="#">entity.h</a>	46
src/platformer/ <a href="#">hero.h</a>	46





## 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

<i>filename</i>	the image to load
<i>style</i>	

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

<i>object</i>	the <a href="#">Object</a> to follow around
<i>graphics</i>	a pointer to a <a href="#">Graphics</a> 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

<i>x</i>	the horizontal position on <a href="#">Room</a> of the <a href="#">Camera</a>
<i>y</i>	the vertical position on <a href="#">Room</a> of the <a href="#">Camera</a>
<i>graphics</i>	a pointer to a <a href="#">Graphics</a> 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

<i>object</i>	the object to go to.
<i>smooth</i>	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

<i>x</i>	the horizontal coordinate of the destination point
<i>y</i>	the vertical coordinate of the destination point
<i>smooth</i>	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

<i>xspe</i>	horizontal speed
<i>yspe</i>	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 within 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:

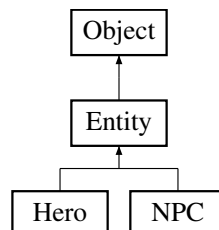
- [src/room.h](#)
- [src/room.cpp](#)

## 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](#))  
*Default constructor. Works just like it would work with an [Object](#).*
- [~Entity](#) ()  
*Default 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* )

Default constructor. Works just like it would work with an [Object](#).

#### 9.3.2.2 Entity::~Entity ( )

Default 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

Controls 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](#) \*src, [SDL\\_Texture](#) \*tdest, int x, int y, bool clear)  
*Blits a source texture into a destination texture.*
- void [blitTexture](#) ([SDL\\_Texture](#) \*src, [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 \*src, 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

Controls 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

<i>winWidth</i>	the game width
<i>winHeight</i>	the game height
<i>winCaption</i>	the window caption
<i>options</i>	a pointer to an <a href="#">Options</a> 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

<i>sprite</i>	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

<i>tsrc</i>	the source texture
<i>tdest</i>	the destination texture
<i>x</i>	destination horizontal position
<i>y</i>	destination vertical position
<i>clear</i>	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

<i>tsrc</i>	the source texture
<i>tdest</i>	the destination texture
<i>x</i>	destination horizontal position
<i>y</i>	destination vertical position
<i>w</i>	width to scale the given texture to
<i>h</i>	height to scale the given texture to
<i>clear</i>	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

<i>tsrc</i>	the source texture
<i>tdest</i>	the destination texture
<i>src</i>	SDL_Rect of the source texture
<i>dest</i>	SDL_Rect of the destination texture
<i>clear</i>	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

--

<i>source</i>	the texture to blit
<i>x</i>	horizontal position on screen
<i>y</i>	vertical position on screen
<i>w</i>	width to scale the given texture to
<i>h</i>	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

<i>filename</i>	the bitmap to load (extension included)
<i>drawable</i>	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:

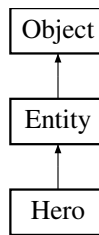
- [src/graphics.h](#)
- [src/graphics.cpp](#)

## 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:

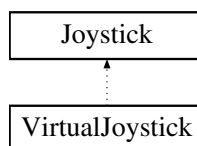
- `src/platformer/entity.h`

## 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:

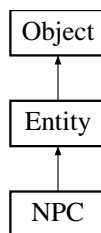
- [src/joystick.h](#)
- [src/joystick.cpp](#)

## 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:

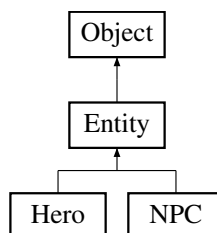
- [src/platformer/entity.h](#)

## 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

<i>filename</i>	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

<code>s</code>	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

<i>filename</i>	the design file
<i>scriptname</i>	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

<i>w</i>	the <a href="#">Room</a> 's width
<i>h</i>	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

<i>camera</i>	a pointer to a <a href="#">Camera</a> 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 override 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

<i>filename</i>	the bitmap filename where all the sprite images are to be loaded
<i>w</i>	the width of each image
<i>h</i>	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

<i>animation</i>	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>i</i>	the animation index
----------	---------------------

Gets 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>i</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

<i>idx</i>	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>i</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>i</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](#)

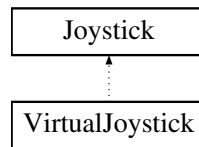


## 9.12 VirtualJoystick Class Reference

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

<i>filename</i>	the bitmap filename where all the sprite images are to be loaded
<i>w</i>	the width of each image
<i>h</i>	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

<i>filename</i>	the bitmap filename where all the sprite images are to be loaded
<i>w</i>	the width of each image
<i>h</i>	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

- ~Background
  - Background, [19](#)
- ~Camera
  - Camera, [21](#)
- ~Entity
  - Entity, [23](#)
- ~Graphics
  - Graphics, [24](#)
- ~Object
  - Object, [30](#)
- ~Room
  - Room, [34](#)
- ~Sprite
  - Sprite, [36](#)
- addAnimation
  - Sprite, [36](#)
- AddBackground
  - S2M\_Room, [15](#)
- addObject
  - Room, [34](#)
  - Sprite, [37](#)
- addSprite
  - Graphics, [25](#)
- Background, [17](#)
  - ~Background, [19](#)
  - Background, [18](#), [19](#)
  - Camera, [22](#)
  - getHeight, [19](#)
  - getWidth, [19](#)
  - Graphics, [19](#)
  - update, [19](#)
  - x, [19](#)
  - xspeed, [19](#)
  - y, [19](#)
  - yspeed, [19](#)
- blitTexture
  - Graphics, [26](#)
- Camera, [19](#)
  - ~Camera, [21](#)
  - Background, [22](#)
  - Camera, [20](#)
  - goTo, [21](#)
  - Graphics, [22](#)
  - move, [21](#)
  - setSpeed, [21](#)
  - update, [21](#)
  - x, [22](#)
  - xspeed, [22](#)
  - y, [22](#)
  - yspeed, [22](#)
- camera
  - Room, [35](#)
- compareObjectsByDepth
  - object.h, [44](#)
  - Room, [35](#)
  - room.cpp, [47](#)
  - room.h, [48](#)
- createObject
  - Room, [34](#)
- drawTexture
  - Graphics, [26](#)
- Entity, [22](#)
  - ~Entity, [23](#)
  - Entity, [23](#)
  - update, [23](#)
- entity.cpp
  - S2M\_CreateEntity, [46](#)
- entity.h
  - S2M\_CreateEntity, [46](#)
- FindAllStrings
  - S2M\_Script, [16](#)
- gGraphics
  - graphics.cpp, [42](#)
  - graphics.h, [43](#)
- gJoystick
  - joystick.cpp, [43](#)
  - joystick.h, [44](#)
- gOptions
  - graphics.cpp, [42](#)
  - graphics.h, [43](#)
- gRoom
  - room.cpp, [47](#)
  - room.h, [49](#)
- gameHeight
  - Graphics, [27](#)
- gameInPause
  - pause.cpp, [45](#)
  - pause.h, [45](#)
- gameWidth
  - Graphics, [27](#)
- getAnimation
  - Object, [30](#)
- getAnimationsSize

- Sprite, 37
- getBtn
  - Joystick, 28
- getDepth
  - Object, 30
- getDir
  - Joystick, 28
- getFrame
  - Sprite, 37
- getHeight
  - Background, 19
  - Room, 34
  - Sprite, 37
- getRect
  - Sprite, 37
- getScale
  - Options, 31
- getWidth
  - Background, 19
  - Room, 34
  - Sprite, 37
- goTo
  - Camera, 21
- Graphics, 23
  - ~Graphics, 24
  - addSprite, 25
  - Background, 19
  - blitTexture, 26
  - Camera, 22
  - drawTexture, 26
  - gameHeight, 27
  - gameWidth, 27
  - Graphics, 24
  - loadTexture, 27
  - Object, 30
  - renderer, 27
  - Room, 35
  - update, 27
  - window, 27
- graphics.cpp
  - gGraphics, 42
  - gOptions, 42
  - S2M\_CreateGraphics, 41
  - S2M\_CreateSprite, 41
  - S2M\_UpdateGraphics, 41
- graphics.h
  - gGraphics, 43
  - gOptions, 43
  - S2M\_CreateGraphics, 42
  - S2M\_CreateSprite, 42
  - S2M\_UpdateGraphics, 43
- Graphics::update
  - Sprite, 38
- Hero, 27
- itnMap
  - script.cpp, 50
  - script.h, 51
- Joystick, 28
  - getBtn, 28
  - getDir, 28
  - update, 29
- joystick.cpp
  - gJoystick, 43
- joystick.h
  - gJoystick, 44
- LoadScript
  - S2M\_Room, 15
- loadTexture
  - Graphics, 27
- move
  - Camera, 21
- NPC, 29
- Object, 29
  - ~Object, 30
  - getAnimation, 30
  - getDepth, 30
  - Graphics, 30
  - Object, 30
  - Room::update, 30
  - setAnimation, 30
  - setDepth, 30
  - sprite, 30
  - update, 30
  - x, 31
  - y, 31
- object.h
  - compareObjectsByDepth, 44
- Object::update
  - Sprite, 38
- Options, 31
  - getScale, 31
  - Options, 31
  - setScale, 31
- ParseCommand
  - S2M\_Script, 16
- ParseFile
  - S2M\_Script, 16
- pause.cpp
  - gameInPause, 45
  - S2M\_PauseGame, 45
  - S2M\_UnpauseGame, 45
- pause.h
  - gameInPause, 45
  - S2M\_PauseGame, 45
  - S2M\_UnpauseGame, 45
- ReadFile
  - S2M\_Script, 16
- renderer
  - Graphics, 27
- retAnimation
  - Sprite, 38

- Room, 33
  - ~Room, 34
  - addObject, 34
  - camera, 35
  - compareObjectsByDepth, 35
  - createObject, 34
  - getHeight, 34
  - getWidth, 34
  - Graphics, 35
  - Room, 34
  - S2M\_Room::AddBackground, 35
  - S2M\_Room::LoadScript, 35
  - setCamera, 34
  - update, 35
- room.cpp
  - compareObjectsByDepth, 47
  - gRoom, 47
  - S2M\_CreateRoom, 47
  - S2M\_SetRoom, 47
  - S2M\_UpdateRoom, 47
- room.h
  - compareObjectsByDepth, 48
  - gRoom, 49
  - S2M\_CreateRoom, 48
  - S2M\_SetRoom, 49
  - S2M\_UpdateRoom, 49
  - STYLE\_CENTER, 48
  - STYLE\_FILL, 48
  - STYLE\_MOSAIC, 48
  - STYLE\_PARALLAX, 48
  - STYLE\_STATIC, 48
- Room::update
  - Object, 30
- S2M.h
  - S2M\_Update, 49
- S2M\_CreateEntity
  - entity.cpp, 46
  - entity.h, 46
- S2M\_CreateGraphics
  - graphics.cpp, 41
  - graphics.h, 42
- S2M\_CreateRoom
  - room.cpp, 47
  - room.h, 48
- S2M\_CreateSprite
  - graphics.cpp, 41
  - graphics.h, 42
- S2M\_PauseGame
  - pause.cpp, 45
  - pause.h, 45
- S2M\_Room, 15
  - AddBackground, 15
  - LoadScript, 15
- S2M\_Room::AddBackground
  - Room, 35
- S2M\_Room::LoadScript
  - Room, 35
- S2M\_Script, 15
  - FindAllStrings, 16
  - ParseCommand, 16
  - ParseFile, 16
  - ReadFile, 16
  - SplitString, 16
- S2M\_SetRoom
  - room.cpp, 47
  - room.h, 49
- S2M\_UnpauseGame
  - pause.cpp, 45
  - pause.h, 45
- S2M\_Update
  - S2M.h, 49
- S2M\_UpdateGraphics
  - graphics.cpp, 41
  - graphics.h, 43
- S2M\_UpdateRoom
  - room.cpp, 47
  - room.h, 49
- STYLE\_CENTER
  - room.h, 48
- STYLE\_FILL
  - room.h, 48
- STYLE\_MOSAIC
  - room.h, 48
- STYLE\_PARALLAX
  - room.h, 48
- STYLE\_STATIC
  - room.h, 48
- script.cpp
  - itnMap, 50
- script.h
  - itnMap, 51
- setAnimation
  - Object, 30
- setCamera
  - Room, 34
- setDepth
  - Object, 30
- setFrame
  - Sprite, 38
- setScale
  - Options, 31
- setSpeed
  - Camera, 21
- SplitString
  - S2M\_Script, 16
- Sprite, 35
  - ~Sprite, 36
  - addAnimation, 36
  - addObject, 37
  - getAnimationsSize, 37
  - getFrame, 37
  - getHeight, 37
  - getRect, 37
  - getWidth, 37
  - Graphics::update, 38
  - Object::update, 38



- [retAnimation](#), 38
  - [setFrame](#), 38
  - [Sprite](#), 36
  - [update](#), 38
- [sprite](#)
  - [Object](#), 30
- [src/S2M.h](#), 49
- [src/S2M\\_Platformer.h](#), 49
- [src/graphics.cpp](#), 41
- [src/graphics.h](#), 42
- [src/joystick.cpp](#), 43
- [src/joystick.h](#), 43
- [src/object.cpp](#), 44
- [src/object.h](#), 44
- [src/options.cpp](#), 44
- [src/options.h](#), 44
- [src/pause.cpp](#), 45
- [src/pause.h](#), 45
- [src/platformer/entity.cpp](#), 46
- [src/platformer/entity.h](#), 46
- [src/platformer/hero.h](#), 46
- [src/room.cpp](#), 46
- [src/room.h](#), 47
- [src/script.cpp](#), 49
- [src/script.h](#), 50
- [update](#)
  - [Background](#), 19
  - [Camera](#), 21
  - [Entity](#), 23
  - [Graphics](#), 27
  - [Joystick](#), 29
  - [Object](#), 30
  - [Room](#), 35
  - [Sprite](#), 38
- [VirtualJoystick](#), 39
- [window](#)
  - [Graphics](#), 27
- [x](#)
  - [Background](#), 19
  - [Camera](#), 22
  - [Object](#), 31
- [xspeed](#)
  - [Background](#), 19
  - [Camera](#), 22
- [y](#)
  - [Background](#), 19
  - [Camera](#), 22
  - [Object](#), 31
- [yspeed](#)
  - [Background](#), 19
  - [Camera](#), 22