## Cloth simulation

Generated by Doxygen 1.8.9.1

Thu Jan 28 2016 00:29:57

## **Contents**

1	Proj	ect Ove	rview		1
	1.1	Introdu	ction		2
	1.2	Installa	tion		2
		1.2.1	Minimal F	Requirements	2
		1.2.2	Depende	encies	2
		1.2.3	Compiling	g	2
2	Nam	nespace	Index		3
	2.1	Names	pace List		3
3	Clas	s Index			5
	3.1	Class L	ist		5
4	File	Index			7
	4.1	File Lis	t		7
5	Nam	nespace	Documer	ntation	9
	5.1	std Nar	mespace F	Reference	9
		5.1.1	Function	Documentation	10
			5.1.1.1	baryVertexInTriangle	10
			5.1.1.2	collisionDetectionAndResponse	10
			5.1.1.3	f_int	11
			5.1.1.4	force	11
			5.1.1.5	fwd_euler	12
			5.1.1.6	gravity	12
			5.1.1.7	leapfrog	13
			5.1.1.8	midpoint	13
			5.1.1.9	symplectic	14
			5.1.1.10	vertexInTriangle	15
		5.1.2	Variable I	Documentation	15
			5.1.2.1	floorLevel	15
			5.1.2.2	leapFrogInitialized	15
			5123	model3dFwdIndex	15

iv CONTENTS

	5.1.2.4	model3dFwdIndexLength	15
	5.1.2.5	model3dMasses	15
	5.1.2.6	model3dMassFwdOffs	15
	5.1.2.7	model3dMassRevOffs	15
	5.1.2.8	model3dMassRevOffsOrig	15
	5.1.2.9	model3dMassVertices	15
	5.1.2.10	model3dNormals	15
	5.1.2.11	model3dObjectLength	16
	5.1.2.12	model3dObjectNames	16
	5.1.2.13	model3dObjectOffset	16
	5.1.2.14	model3dObjects	16
	5.1.2.15	model3dObjectsWithMass	16
	5.1.2.16	model3dPositions	16
	5.1.2.17	model3dRevIndex	16
	5.1.2.18	model3dTexels	16
	5.1.2.19	model3dTextureFilePath	16
	5.1.2.20	model3dVertices	16
	5.1.2.21	repulsiveSpringConst	16
	5.1.2.22	$t \ldots \ldots \ldots \ldots$	16
Class D	ocumentation	n	17
			17
6.1		Description	18
6.1 6.1	.1 Detailed	Description	18 19
	.1 Detailed	ctor & Destructor Documentation	19
	.1 Detailed .2 Construct 6.1.2.1	Contractor & Destructor Documentation	19 19
	.1 Detailed .2 Construct 6.1.2.1 6.1.2.2	Accelerometer	19 19 19
	.1 Detailed .2 Construct 6.1.2.1 6.1.2.2 6.1.2.3	Accelerometer	19 19
6.1	.1 Detailed .2 Construct 6.1.2.1 6.1.2.2 6.1.2.3	Accelerometer	19 19 19
6.1	.1 Detailed .2 Construct 6.1.2.1 6.1.2.2 6.1.2.3 .3 Member	Accelerometer	19 19 19 19 20
6.1	.1 Detailed .2 Construct 6.1.2.1 6.1.2.2 6.1.2.3 .3 Member 6.1.3.1	Accelerometer  Accelerometer  Accelerometer  Accelerometer  Function Documentation  accelerometerInitDevice  closeUpdateThread	19 19 19 19 20 20
6.1	.1 Detailed .2 Construct 6.1.2.1 6.1.2.2 6.1.2.3 .3 Member 6.1.3.1 6.1.3.2	Accelerometer  Accelerometer  Accelerometer  Accelerometer  Function Documentation  accelerometerInitDevice	19 19 19 20 20
6.1	.1 Detailed .2 Construct 6.1.2.1 6.1.2.2 6.1.2.3 .3 Member 6.1.3.1 6.1.3.2 6.1.3.3	Accelerometer  Accelerometer  Accelerometer  Accelerometer  Function Documentation  accelerometerInitDevice  closeUpdateThread  deviceConfigFromFile	19 19 19 20 20 20
6.1	.1 Detailed .2 Construct 6.1.2.1 6.1.2.2 6.1.2.3 .3 Member 6.1.3.1 6.1.3.2 6.1.3.3 6.1.3.4	Accelerometer  Accelerometer  Accelerometer  Accelerometer  Function Documentation  accelerometerInitDevice  closeUpdateThread  deviceConfigFromFile  getX	19 19 19 20 20 20 20
6.1	.1 Detailed .2 Construct 6.1.2.1 6.1.2.2 6.1.2.3 .3 Member 6.1.3.1 6.1.3.2 6.1.3.3 6.1.3.4 6.1.3.5	ctor & Destructor Documentation  Accelerometer  Accelerometer  ~Accelerometer  Function Documentation  accelerometerInitDevice  closeUpdateThread  deviceConfigFromFile  getX  getY	19 19 19 20 20 20 21 21
6.1	.1 Detailed .2 Construct 6.1.2.1 6.1.2.2 6.1.2.3 .3 Member 6.1.3.1 6.1.3.2 6.1.3.3 6.1.3.4 6.1.3.5 6.1.3.6	ctor & Destructor Documentation  Accelerometer  Accelerometer  ~Accelerometer  Function Documentation  accelerometerInitDevice  closeUpdateThread  deviceConfigFromFile  getX  getY  getZ	19 19 19 20 20 20 21 21
6.1	.1 Detailed .2 Construct 6.1.2.1 6.1.2.2 6.1.2.3 .3 Member 6.1.3.1 6.1.3.2 6.1.3.3 6.1.3.4 6.1.3.5 6.1.3.6 6.1.3.7	ctor & Destructor Documentation  Accelerometer  Accelerometer  ~Accelerometer  Function Documentation  accelerometerInitDevice  closeUpdateThread  deviceConfigFromFile  getX  getY  getZ  initUpdateThread	19 19 19 20 20 20 21 21 21
6.1	.1 Detailed .2 Construct 6.1.2.1 6.1.2.2 6.1.2.3 .3 Member 6.1.3.1 6.1.3.2 6.1.3.3 6.1.3.4 6.1.3.5 6.1.3.6 6.1.3.7 6.1.3.8	Accelerometer Accelerometer  Accelerometer  Accelerometer  Function Documentation  accelerometerInitDevice  closeUpdateThread  deviceConfigFromFile  getX  getY  getZ  initUpdateThread  isAvailable  objectCounter	19 19 19 20 20 20 21 21 21 21
6.1	.1 Detailed .2 Construct 6.1.2.1 6.1.2.2 6.1.2.3 .3 Member 6.1.3.1 6.1.3.2 6.1.3.3 6.1.3.4 6.1.3.5 6.1.3.6 6.1.3.7 6.1.3.8 6.1.3.9 6.1.3.10	Accelerometer Accelerometer  Accelerometer  Accelerometer  Function Documentation  accelerometerInitDevice  closeUpdateThread  deviceConfigFromFile  getX  getY  getZ  initUpdateThread  isAvailable  objectCounter	19 19 19 20 20 20 21 21 21 21 21 22
		5.1.2.6 5.1.2.7 5.1.2.8 5.1.2.9 5.1.2.10 5.1.2.11 5.1.2.12 5.1.2.13 5.1.2.14 5.1.2.15 5.1.2.16 5.1.2.17 5.1.2.18 5.1.2.19 5.1.2.20 5.1.2.21 5.1.2.22 Class Documentation	5.1.2.6       model3dMassRevOffs         5.1.2.7       model3dMassRevOffs         5.1.2.8       model3dMassRevOffsOrig         5.1.2.9       model3dMassVertices         5.1.2.10       model3dNormals         5.1.2.11       model3dObjectLength         5.1.2.12       model3dObjectNames         5.1.2.13       model3dObjectOffset         5.1.2.14       model3dObjects         5.1.2.15       model3dObjectsWithMass         5.1.2.16       model3dRevIndex         5.1.2.17       model3dRevIndex         5.1.2.18       model3dTexels         5.1.2.21       model3dTextureFilePath         5.1.2.21       repulsiveSpringConst         5.1.2.22       t

CONTENTS

		6.1.4.2	devicePath	22
		6.1.4.3	done	22
		6.1.4.4	mtx	22
		6.1.4.5	present	22
		6.1.4.6	sensitivity	22
		6.1.4.7	updaterThread	22
		6.1.4.8	x	22
		6.1.4.9	y	22
		6.1.4.10	z	22
6.2	Image	Struct Ref	erence	23
	6.2.1	Member	Data Documentation	23
		6.2.1.1	colorMap	23
		6.2.1.2	dim	23
		6.2.1.3	file	23
		6.2.1.4	imagic	23
		6.2.1.5	max	24
		6.2.1.6	min	24
		6.2.1.7	name	24
		6.2.1.8	rleEnd	24
		6.2.1.9	rowSize	24
		6.2.1.10	rowStart	24
		6.2.1.11	sizeX	24
		6.2.1.12	sizeY	24
		6.2.1.13	sizeZ	24
		6.2.1.14	tmp	24
		6.2.1.15	type	24
		6.2.1.16	wasteBytes	24
6.3	IMAGE	Struct Re	eference	24
	6.3.1	Member	Data Documentation	25
		6.3.1.1	data	25
		6.3.1.2	dim	25
		6.3.1.3	imagic	25
		6.3.1.4	name	25
		6.3.1.5	sizeX	25
		6.3.1.6	sizeY	25
		6.3.1.7	sizeZ	25
		6.3.1.8	type	25
6.4	Mass (	Class Refe	rence	25
	6.4.1	Detailed	Description	27
	6.4.2	Construc	tor & Destructor Documentation	27

vi CONTENTS

		6.4.2.1	Mass	27
		6.4.2.2	Mass	27
		6.4.2.3	Mass	27
		6.4.2.4	Mass	27
		6.4.2.5	~Mass	28
	6.4.3	Member	Function Documentation	28
		6.4.3.1	addForce	28
		6.4.3.2	getForce	28
		6.4.3.3	getPos	28
		6.4.3.4	getUserForce	28
		6.4.3.5	getVel	28
		6.4.3.6	registerVertex	29
		6.4.3.7	setForce	30
		6.4.3.8	setPos	30
		6.4.3.9	setUserForce	30
		6.4.3.10	setVel	30
		6.4.3.11	setX	30
		6.4.3.12	setY	30
		6.4.3.13	setZ	31
	6.4.4	Member	Data Documentation	31
		6.4.4.1	damping	31
		6.4.4.2	fixed	31
		6.4.4.3	force	31
		6.4.4.4	mass	31
		6.4.4.5	posPtr	31
		6.4.4.6	userForce	31
		6.4.4.7	velocity	31
6.5	std::Mip	oMap Clas	ss Reference	31
	6.5.1	Detailed	Description	32
	6.5.2	Construc	tor & Destructor Documentation	32
		6.5.2.1	MipMap	32
		6.5.2.2	~MipMap	33
	6.5.3	Member	Function Documentation	33
		6.5.3.1	load	33
		6.5.3.2	loadPNG	33
		6.5.3.3	loadSGI	33
	6.5.4	Member	Data Documentation	34
		6.5.4.1	height	34
		6.5.4.2	texture	34
		6.5.4.3	width	34

CONTENTS vii

6.6	std::Sc	ene Class	Reference	34
	6.6.1	Detailed	Description	36
	6.6.2	Construc	tor & Destructor Documentation	36
		6.6.2.1	Scene	36
		6.6.2.2	Scene	36
		6.6.2.3	~Scene	36
	6.6.3	Member	Function Documentation	36
		6.6.3.1	detectCollisions	36
		6.6.3.2	getStep	37
		6.6.3.3	initialize	37
		6.6.3.4	initPointsVector	37
		6.6.3.5	initSpringsVector	37
		6.6.3.6	render	37
		6.6.3.7	update	38
		6.6.3.8	updateNormals	38
	6.6.4	Member	Data Documentation	38
		6.6.4.1	damping	38
		6.6.4.2	mass	38
		6.6.4.3	points	38
		6.6.4.4	springs	38
		6.6.4.5	step	38
		6.6.4.6	stiffness	38
		6.6.4.7	weak_stiff	38
6.7	std::Sir	mulation C	lass Reference	39
	6.7.1	Detailed	Description	39
	6.7.2	Member	Enumeration Documentation	39
		6.7.2.1	Method	39
	6.7.3	Member	Function Documentation	39
		6.7.3.1	step	40
6.8	Spring	Class Refe	erence	41
	6.8.1	Detailed	Description	43
	6.8.2	Construc	tor & Destructor Documentation	43
		6.8.2.1	Spring	43
		6.8.2.2	Spring	43
		6.8.2.3	Spring	43
		6.8.2.4	Spring	43
		6.8.2.5	$\sim$ Spring	44
	6.8.3	Member	Function Documentation	44
		6.8.3.1	getMass	44
		6.8.3.2	init	44

viii CONTENTS

		6.8.4	Member	Data Documentation	. 44
			6.8.4.1	m0	. 44
			6.8.4.2	m1	. 44
			6.8.4.3	restLength	. 44
			6.8.4.4	stiffness	. 44
7	File	Docum	entation		47
	7.1	src/Ac	celeromete	er.cpp File Reference	. 47
		7.1.1	Macro De	efinition Documentation	. 47
			7.1.1.1	_PRIV_LINE_LEN	. 47
	7.2	src/Acc	celeromete	er.h File Reference	. 47
		7.2.1	Macro De	efinition Documentation	. 48
			7.2.1.1	CFG_FILENAME	. 48
	7.3	src/Co	llision.cpp	File Reference	. 49
	7.4	src/Co	llision.h Fil	le Reference	. 49
		7.4.1	Detailed	Description	. 50
	7.5	src/dic	e.c File Re	eference	. 51
		7.5.1	Detailed	Description	. 51
		7.5.2	Variable	Documentation	. 52
			7.5.2.1	diceFwdIndex	. 52
			7.5.2.2	diceFwdIndexI	. 52
			7.5.2.3	diceFwdIndexLength	. 52
			7.5.2.4	diceMasses	. 53
			7.5.2.5	diceMassFwdOffs	. 53
			7.5.2.6	diceMassRevOffs	. 53
			7.5.2.7	diceMassRevOffsOrig	. 53
			7.5.2.8	diceMassVertices	. 53
			7.5.2.9	diceNormals	. 53
			7.5.2.10	diceObjectLength	. 53
			7.5.2.11	diceObjectNames	. 54
			7.5.2.12	diceObjectNamesString	. 54
			7.5.2.13	diceObjectOffset	. 54
			7.5.2.14	diceObjects	. 54
			7.5.2.15	diceObjectsWithMass	. 54
			7.5.2.16	dicePositions	. 54
			7.5.2.17	diceRevIndex	. 54
			7.5.2.18	diceTexels	. 55
			7.5.2.19	diceTextureFilePath	. 55
			7.5.2.20	diceVertices	. 55
	7.6	src/dic	e.h File Re	eference	. 55

CONTENTS

	7.6.1	Detailed	Description	56
	7.6.2	Variable	Documentation	56
		7.6.2.1	diceFwdIndex	56
		7.6.2.2	diceFwdIndexI	56
		7.6.2.3	diceFwdIndexLength	56
		7.6.2.4	diceMasses	56
		7.6.2.5	diceMassFwdOffs	56
		7.6.2.6	diceMassRevOffs	56
		7.6.2.7	diceMassRevOffsOrig	56
		7.6.2.8	diceMassVertices	56
		7.6.2.9	diceNormals	56
		7.6.2.10	diceObjectLength	56
		7.6.2.11	diceObjectNames	56
		7.6.2.12	diceObjectNamesString	56
		7.6.2.13	diceObjectOffset	56
		7.6.2.14	diceObjects	56
		7.6.2.15	diceObjectsWithMass	56
		7.6.2.16	dicePositions	56
		7.6.2.17	diceRevIndex	56
		7.6.2.18	diceTexels	56
		7.6.2.19	diceTextureFilePath	57
		7.6.2.20	diceVertices	57
7.7	src/ma	in.cpp File	Reference	57
	7.7.1	Function	Documentation	57
		7.7.1.1	displayCallback	57
		7.7.1.2	getTime	58
		7.7.1.3	idleCallback	58
		7.7.1.4	initialize	58
		7.7.1.5	keyboardCallback	58
		7.7.1.6	main	58
		7.7.1.7	motionCallback	59
		7.7.1.8	mouseCallback	59
		7.7.1.9	reshapeCallback	59
		7.7.1.10	updateSimulation	59
	7.7.2	Variable	Documentation	60
		7.7.2.1	MAX_UPDATE_TIME	60
		7.7.2.2	scene	60
7.8	src/Ma	ss.cpp File	Reference	60
	7.8.1	Macro De	efinition Documentation	61
		7.8.1.1	_DEBUG_MSG	61

X CONTENTS

		7.8.1.2	_DEBUG_MSG_C	. 61
7.9	src/Ma	ss.h File F	Reference	. 61
7.	0 src/Mip	оМар.срр Г	File Reference	. 61
7.	1 src/Mip	oMap.h File	e Reference	. 62
7.	2 src/mo	del_mappi	ing.cpp File Reference	. 63
	7.12.1	Detailed	Description	. 65
	7.12.2	Macro De	efinition Documentation	. 65
		7.12.2.1	MODEL	. 65
7.	3 src/mo	del_mappi	ing.h File Reference	. 65
	7.13.1	Detailed	Description	. 66
7.	4 src/Re	sourcePatl	h.h File Reference	. 66
	7.14.1	Function	Documentation	. 67
		7.14.1.1	getBasePath	. 67
		7.14.1.2	getResourcePath	. 67
7.	5 src/Sc	ene.cpp Fil	ile Reference	. 67
7.	6 src/Sc	ene.h File	Reference	. 68
7.	7 src/SG	ilimage.c F	File Reference	. 69
	7.17.1	Macro De	efinition Documentation	. 70
		7.17.1.1	IMAGIC	. 70
		7.17.1.2	IMAGIC_SWAP	. 70
		7.17.1.3	SWAP_LONG_BYTES	. 70
		7.17.1.4	SWAP_SHORT_BYTES	. 71
	7.17.2	Function	Documentation	. 71
		7.17.2.1	ImageClose	. 71
		7.17.2.2	ImageGetRawData	. 71
		7.17.2.3	ImageGetRow	. 71
		7.17.2.4	ImageLoad	. 71
		7.17.2.5	ImageOpen	. 71
7.	8 src/SG	ilimage.h F	File Reference	. 72
	7.18.1	Function	Documentation	. 72
		7.18.1.1	ImageLoad	. 72
7.	9 src/Sin	nulation.cp	pp File Reference	. 73
7.2	20 src/Sin	nulation.h l	File Reference	. 74
7.2	21 src/ski	rt_sphere.c	c File Reference	. 74
	7.21.1	Detailed	Description	. 75
	7.21.2	Variable	Documentation	. 75
		7.21.2.1	skirt_sphereFwdIndex	. 75
		7.21.2.2		
		7.21.2.3		
		7.21.2.4		

CONTENTS xi

7	.21.2.5	skirt_sphereMassFwdOffs	 75
7	.21.2.6	skirt_sphereMassRevOffs	 76
7	.21.2.7	skirt_sphereMassRevOffsOrig	 76
7	.21.2.8	skirt_sphereMassVertices	 76
7	.21.2.9	skirt_sphereNormals	 76
7	.21.2.10	skirt_sphereObjectLength	 76
7	.21.2.11	skirt_sphereObjectNames	 76
7	.21.2.12	skirt_sphereObjectNamesString	 76
7	.21.2.13	skirt_sphereObjectOffset	 77
7	.21.2.14	skirt_sphereObjects	 77
7	.21.2.15	skirt_sphereObjectsWithMass	 77
7	.21.2.16	skirt_spherePositions	 77
7	.21.2.17	skirt_sphereRevIndex	 77
7	.21.2.18	skirt_sphereTexels	 77
7	.21.2.19	skirt_sphereTextureFilePath	 77
7	.21.2.20	skirt_sphereVertices	 77
7.22 src/skirt_	sphere.h	File Reference	 77
7.22.1 D	Detailed D	Description	 78
7.22.2 V	/ariable D	Occumentation	 78
7	.22.2.1	skirt_sphereFwdIndex	 78
7	.22.2.2	skirt_sphereFwdIndexI	 78
7	.22.2.3	skirt_sphereFwdIndexLength	 78
7	.22.2.4	skirt_sphereMasses	 79
7	.22.2.5	skirt_sphereMassFwdOffs	 79
7	.22.2.6	skirt_sphereMassRevOffs	 79
7	.22.2.7	skirt_sphereMassRevOffsOrig	 79
7	.22.2.8	skirt_sphereMassVertices	 79
7	.22.2.9	skirt_sphereNormals	 79
7	.22.2.10	skirt_sphereObjectLength	 79
7	.22.2.11	skirt_sphereObjectNames	 79
7	.22.2.12	skirt_sphereObjectNamesString	 79
7	.22.2.13	skirt_sphereObjectOffset	 79
7	.22.2.14	skirt_sphereObjects	 79
7	.22.2.15	skirt_sphereObjectsWithMass	 79
7	.22.2.16	skirt_spherePositions	 79
7	.22.2.17	skirt_sphereRevIndex	 79
7	.22.2.18	skirt_sphereTexels	 79
7	.22.2.19	skirt_sphereTextureFilePath	 79
7	.22.2.20	skirt_sphereVertices	 79
7.23 src/Spring	g.cpp File	e Reference	 79

xii CONTENTS

7.23.1	Macro Definition Documentation	80
	7.23.1.1 _DEBUG_MSG	80
	7.23.1.2 _DEBUG_MSG_C	80
7.24 src/Sp	ring.h File Reference	80

# Chapter 1

# **Project Overview**

## Author

Elias Zischg Daly Chea Gerhard Aigner 2 Project Overview

## 1.1 Introduction

Here goes the introduction of the project

## 1.2 Installation

## 1.2.1 Minimal Requirements

- · Linux based OS
- · Memory min. 4 MB RAM
- CPU min. speed 1 GHz with 2 cores e.g. i5-2540 or better
- approx 1 MB free hard-disk space

## 1.2.2 Dependencies

- OpenGL and utility libraries (package libglu\*\*\*-dev e.g. libglu-mesa-dev; depends on graphics hardware)
- OpenGL utilities toolkit (package freeglut3-dev)
- libPNG PNG file handling (package libpng12-dev)
- Eigen math utilities for vector and matrix arithmetics (package libeigen3-dev)
- C++ compiler with C++ 11 support e.g. GNU GCC g++ V4.7 or better

## 1.2.3 Compiling

Go to the sub-folder build of the project folder and run make all.

If you want to compile directly on the console use the following options:

- -std=c++11 or -std=c++0x (use the C++ 11 dialect)
- -lGL (the OpenGL library)
- -lGLU (OpenGL utility library)
- -lglut (the GL utility Toolkit make sure 'freeglut' is installed, some extensions not present in the GLUT API are used)
- -lpthread (or -lthread; manage threads)
- -lpng (library to handle PNG files)

# Chapter 2

# Namespace Index

2.1	Namespace List
-----	----------------

Here is a list of all namespaces with brief descriptions:	
std	ç

Namespace Index

# **Chapter 3**

## **Class Index**

## 3.1 Class List

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

Accelerometer			 																				17
Image			 																				23
IMAGE			 									 											24
Mass			 									 											25
std::MipMap .			 									 											31
std::Scene .			 																				34
std::Simulation			 									 											39
Spring			 									 											41

6 Class Index

# Chapter 4

## File Index

## 4.1 File List

Here is a list of all files with brief descriptions:

src/Accelerometer.cpp	 47
src/Accelerometer.h	 47
src/Collision.cpp	 49
src/Collision.h	 49
src/dice.c	 51
src/dice.h	 55
src/main.cpp	 57
src/Mass.cpp	 60
src/Mass.h	 61
src/MipMap.cpp	 61
src/MipMap.h	 62
src/model_mapping.cpp	 63
src/model_mapping.h	 65
src/ResourcePath.h	 66
src/Scene.cpp	 67
src/Scene.h	 68
src/SGlimage.c	69
src/SGlimage.h	 72
src/Simulation.cpp	 73
src/Simulation.h	 74
src/skirt_sphere.c	 74
src/skirt_sphere.h	 77
src/Spring.cpp	 79
src/Spring.h	 80

8 File Index

## **Chapter 5**

## **Namespace Documentation**

## 5.1 std Namespace Reference

### Classes

- class MipMap
- class Scene
- · class Simulation

### **Functions**

- bool vertexInTriangle (const Eigen::Vector3d &P, const Eigen::Vector3d &A, const Eigen::Vector3d &B, const Eigen::Vector3d &C, const float epsilon, float &dist, Eigen::Vector3d &N)
- bool baryVertexInTriangle (const Eigen::Vector3d &P, const Eigen::Vector3d &A, const Eigen::Vector3d &B, const Eigen::Vector3d &C)
- void collisionDetectionAndResponse (vector < Mass > &points, size\_t offsP, size\_t lenP, GLfloat object\_
   mesh[], size\_t offsO, size\_t lenO)
- void fwd\_euler (double dt, vector < Mass > &points, vector < Spring > &springs, bool interaction)
- Eigen::Vector3d f\_int (Mass &pt, vector < Spring > &springs)
- Eigen::Vector3d gravity ()
- void symplectic (double dt, vector < Mass > &points, vector < Spring > &springs, bool interaction)
- void leapfrog (double dt, vector < Mass > &points, vector < Spring > &springs, bool interaction)
- void midpoint (double dt, vector < Mass > &points, vector < Spring > &springs, bool interaction)
- void force (vector < Mass > &points, vector < Spring > &springs, bool interaction)

## **Variables**

- const size\_t model3dVertices = MODEL(Vertices)
  - number of vertices
- GLfloat \* model3dPositions = MODEL(Positions)
  - all vertex positions
- GLfloat \* model3dTexels = MODEL(Texels)
  - all texture coordinates
- GLfloat \* model3dNormals = MODEL(Normals)
  - all normals of the face surfaces
- const size\_t model3dObjectsWithMass = MODEL(ObjectsWithMass)
  - number of objects to apply to the mass-spring simulation
- const size\_t \* model3dMasses = MODEL(Masses)

array of 3D-objects with mass

const size\_t \* model3dMassFwdOffs = MODEL(MassFwdOffs)

offset in the points array

const size\_t \* model3dMassVertices = MODEL(MassVertices)

number of vertices per object with mass

const size\_t \* model3dMassRevOffs = MODEL(MassRevOffs)

offsets for the model3dRevIndex array

const size\_t \* model3dMassRevOffsOrig = MODEL(MassRevOffsOrig)

offset to the first vertex of an object with mass

const size\_t \*\* model3dFwdIndex = MODEL(FwdIndex)

indices of the mass-points in the positions array

const size t \* model3dFwdIndexLength = MODEL(FwdIndexLength)

number of indices for each mass-point

const size t \* model3dRevIndex = MODEL(RevIndex)

index of a vertex in the mass-points array

const size t model3dObjects = MODEL(Objects)

number of 3D-objects

const size t \* model3dObjectOffset = MODEL(ObjectOffset)

offset to the first vertex of a 3D-object

const size\_t \* model3dObjectLength = MODEL(ObjectLength)

number of vertices for each 3D-object

const char \*\* model3dObjectNames = (const char\*\*) MODEL(ObjectNames)

names of the objects (for identification)

const char \* model3dTextureFilePath = "textures/textures\_all.rgb"

path to the texture-image file

- bool leapFrogInitialized = false
- double t = 0
- double floorLevel = -1.
- double repulsiveSpringConst = -50.

## 5.1.1 Function Documentation

- 5.1.1.1 bool std::baryVertexInTriangle ( const Eigen::Vector3d & P, const Eigen::Vector3d & A, const Eigen::Vector3d & B, const Eigen::Vector3d & C )
- 5.1.1.2 void std::collisionDetectionAndResponse ( vector< Mass > & points, size\_t offsP, size\_t lenP, GLfloat object\_mesh[], size\_t offsO, size\_t lenO )

Checks if there is a collision with mass spring points and object mesh. Adds penalty forces for collisions.

This algorithm can't deal with self collision, so assure the object\_mesh doesn't contain vertices owned by points Parameters

points	array of mass points
offsP	offset in the points array
lenP	number of mass points
object_mesh	vertices array with all triangles (except them present in points)
offsO	offset in terms of vertices

lenO	number of vertices to compute
10110	number of vertices to compute

Here is the call graph for this function:



## 5.1.1.3 Eigen::Vector3d std::f\_int ( Mass & pt, vector < Spring > & springs )

Compute internal forces of a point.

This is a subroutine for the forward Euler method.

## **Parameters**

pt	the actual point
springs	vector of springs

## Returns

sum of forces of all springs connected to point pt

## 5.1.1.4 void std::force ( vector < Mass > & points, vector < Spring > & springs, bool interaction )

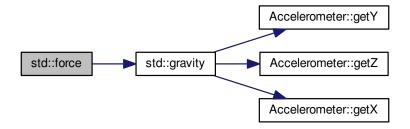
Calculate force for all points at time t according to x(t) stored in the points.

This is a subroutine for the midpoint method.

## **Parameters**

points	all mass-points
springs	all springs
interaction	true = apply external forces other than gravity

Here is the call graph for this function:



5.1.1.5 void std::fwd\_euler ( double dt, vector < Mass > & points, vector < Spring > & springs, bool interaction )

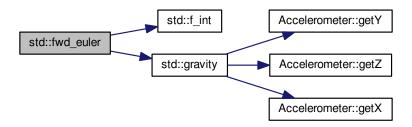
Euler Method - Implementation.

This function calculates the position and velocity for the next time step.

### **Parameters**

dt	time step
points	all mass-points
springs	all springs
interaction	true = apply external forces other than gravity

Here is the call graph for this function:



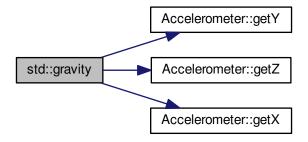
## 5.1.1.6 Eigen::Vector3d std::gravity ( )

The gravity. Either it is determined from an acceleration sensor or if not detected a constant in Z axis.

## Returns

the gravity

Here is the call graph for this function:



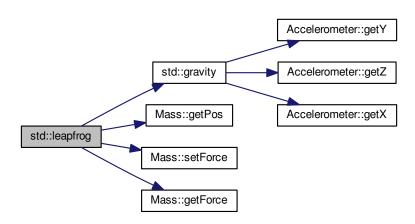
5.1.1.7 void std::leapfrog ( double dt, vector < Mass > & points, vector < Spring > & springs, bool interaction )

## Leapfrog method.

## **Parameters**

dt	time step
points	all mass-points
springs	all springs
interaction	true = apply external forces other than gravity

Here is the call graph for this function:



5.1.1.8 void std::midpoint ( double dt, vector < Mass > & points, vector < Spring > & springs, bool interaction )

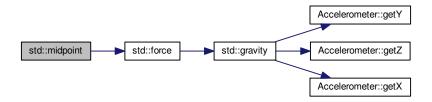
Midpoint method.

```
Formula:
    a(t) = (f(t) - gamma*v(t))/m
    v_half(t+h/2) = v(t) + h/2*a(t)
    x_half(t+h/2) = x(t) + h/2*v(t)
    a_half(t+h/2) = (f_half(t+h/2) - gamma*v_half(t+h/2))/m
    x(t+h) = x(t) + h*v_half(t+h/2)
    v(t+h) = v(t) + h*a_half(t+h/2)
```

### **Parameters**

dt	time step
points	all mass-points
springs	all springs
interaction	true = apply external forces other than gravity

Here is the call graph for this function:



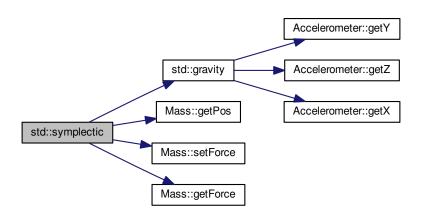
5.1.1.9 void std::symplectic ( double dt, vector < Mass > & points, vector < Spring > & springs, bool interaction )

Symplectic euler method.

### **Parameters**

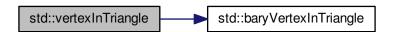
dt	time step
points	all mass-points
springs	all springs
interaction	true = apply external forces other than gravity

Here is the call graph for this function:



5.1.1.10 bool std::vertexInTriangle ( const Eigen::Vector3d & P, const Eigen::Vector3d & A, const Eigen::Vector3d & B, const Eigen::Vector3d & C, const float epsilon, float & dist, Eigen::Vector3d & N)

Here is the call graph for this function:



- 5.1.2 Variable Documentation
- 5.1.2.1 double std::floorLevel = -1.
- 5.1.2.2 bool std::leapFrogInitialized = false
- 5.1.2.3 const size\_t \*\* std::model3dFwdIndex = MODEL(FwdIndex)

indices of the mass-points in the positions array

5.1.2.4 const size\_t \* std::model3dFwdIndexLength = MODEL(FwdIndexLength)

number of indices for each mass-point

5.1.2.5 const size\_t \* std::model3dMasses = MODEL(Masses)

array of 3D-objects with mass

5.1.2.6 const size\_t \* std::model3dMassFwdOffs = MODEL(MassFwdOffs)

offset in the points array

5.1.2.7 const size\_t \* std::model3dMassRevOffs = MODEL(MassRevOffs)

offsets for the model3dRevIndex array

 $5.1.2.8 \quad const \ size\_t * std::model3dMassRevOffsOrig = MODEL(MassRevOffsOrig)$ 

offset to the first vertex of an object with mass

5.1.2.9 const size\_t \* std::model3dMassVertices = MODEL(MassVertices)

number of vertices per object with mass

5.1.2.10 GLfloat \* std::model3dNormals = MODEL(Normals)

all normals of the face surfaces

```
5.1.2.11 const size_t * std::model3dObjectLength = MODEL(ObjectLength)
number of vertices for each 3D-object
5.1.2.12 const char ** std::model3dObjectNames = (const char**) MODEL(ObjectNames)
names of the objects (for identification)
5.1.2.13 const size_t * std::model3dObjectOffset = MODEL(ObjectOffset)
offset to the first vertex of a 3D-object
5.1.2.14 const size_t std::model3dObjects = MODEL(Objects)
number of 3D-objects
5.1.2.15 const size_t std::model3dObjectsWithMass = MODEL(ObjectsWithMass)
number of objects to apply to the mass-spring simulation
5.1.2.16 GLfloat * std::model3dPositions = MODEL(Positions)
all vertex positions
5.1.2.17 const size_t * std::model3dRevIndex = MODEL(RevIndex)
index of a vertex in the mass-points array
5.1.2.18 GLfloat * std::model3dTexels = MODEL(Texels)
all texture coordinates
5.1.2.19 const char * std::model3dTextureFilePath = "textures/textures_all.rgb"
path to the texture-image file
5.1.2.20 const size_t std::model3dVertices = MODEL(Vertices)
number of vertices
5.1.2.21 double std::repulsiveSpringConst = -50.
5.1.2.22 double std::t = 0
```

## **Chapter 6**

## **Class Documentation**

## 6.1 Accelerometer Class Reference

#include <Accelerometer.h>

Collaboration diagram for Accelerometer:

## Accelerometer

- devicePath
- updaterThread
- mtx
- device
- sensitivity
- present
- done
- X
- y - Z
- + Accelerometer()
- + Accelerometer()
- + ~Accelerometer()
- + getX()
- + getY()
- + getZ()
- + isAvailable()
- + accelerometerInitDevice()

- objectCounter()updateValues()initUpdateThread()
- closeUpdateThread()
- deviceConfigFromFile()

18 Class Documentation

## **Public Member Functions**

- Accelerometer ()
- Accelerometer (int argc, char \*argv[])
- ∼Accelerometer ()
- double getX ()
- double getY ()
- double getZ ()

## **Static Public Member Functions**

- static bool isAvailable ()
- static void accelerometerInitDevice (int argc, char \*argv[])

## **Static Private Member Functions**

- static int objectCounter (int count)
- static void updateValues ()
- static void initUpdateThread ()
- static void closeUpdateThread ()
- static void deviceConfigFromFile (std::string configPath)

## **Static Private Attributes**

- · static std::string devicePath
- static std::thread updaterThread
- static std::mutex mtx
- static std::ifstream device
- static double sensitivity = 1.0
- static bool present = false
- static bool done = false
- static double x = 0.0
- static double y = 0.0
- static double z = 0.0

## 6.1.1 Detailed Description

This class handles the connection to the accelerometer device of a notebook.

Actually it supports the accelerometer of type lis3lv02d. Other devices are unsupported, but they may work with proper settings in the configuration file.

## Configuration file:

The file follows the specifications of a windows INI file. Here is an example setup:

```
1 [accelerometer]; this section must be present 2 device = "/sys/devices/platform/lis31v02d/position"; must be an absolute path to the unix device 3 sensitivity = 0.001; multiplier to scale the accelerometer output to fit it to gravity force
```

## 6.1.2 Constructor & Destructor Documentation

### 6.1.2.1 Accelerometer::Accelerometer ( )

Construct an accelerometer device handle.

This constructor fails if no device has been initialized previously by using the constructor which takes command line arguments.

Here is the call graph for this function:



## 6.1.2.2 Accelerometer::Accelerometer ( int argc, char \* argv[] )

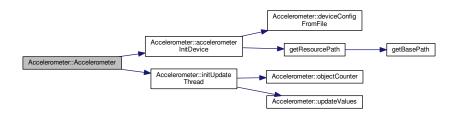
Construct an accelerometer device handle.

This constructor initializes a helping thread which determines periodically data from the device, if it is called the first time.

## **Parameters**

argc	command line arguments counter from main ()
argv	arguments vector

Here is the call graph for this function:



## 6.1.2.3 Accelerometer::~Accelerometer ( )

Destroy the device handle.

If the last handle is destroyed, also the connection to the device is closed and the appertaining thread is destroyed.

20 Class Documentation

Here is the call graph for this function:



## 6.1.3 Member Function Documentation

**6.1.3.1** void Accelerometer::accelerometerInitDevice (int argc, char \* argv[]) [static]

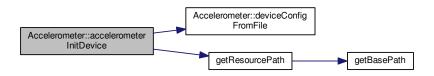
Initialize the device.

This method is called by the constructor of the Accelerometer class.

#### **Parameters**

argc	arguments counter
argv	arguments vector

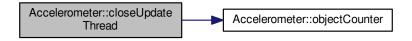
Here is the call graph for this function:



**6.1.3.2 void Accelerometer::closeUpdateThread()** [static], [private]

Close the updater thread if the last instance of Accelerometer is destroyed. This method is called by the destructor of the Accelerometer class.

Here is the call graph for this function:



**6.1.3.3** void Accelerometer::deviceConfigFromFile ( std::string configPath ) [static], [private]

Read configuration for accelerometer device from a config file.

### **Parameters**

configPath	path to the config file

## 6.1.3.4 double Accelerometer::getX ( )

Get the force into X direction.

### Returns

the force determined from the device.

## 6.1.3.5 double Accelerometer::getY()

Get the force into Y direction.

### Returns

the force determined from the device.

## 6.1.3.6 double Accelerometer::getZ ( )

Get the force into Z direction.

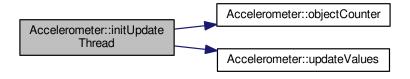
### Returns

the force determined from the device.

## **6.1.3.7 void Accelerometer::initUpdateThread()** [static], [private]

Initialize the updater thread, which reads periodically the state of the accelerometer. This method is called each times an Accelerometer object is created.

Here is the call graph for this function:



## 6.1.3.8 bool Accelerometer::isAvailable( ) [static]

Checks if the device is present.

22 Class Documentation

### Return values

true	if the device is online
false	otherwise

**6.1.3.9** int Accelerometer::objectCounter(int count) [static], [private]

Count the number of open accelerometers

## **Parameters**

count	1 count up, -1 count down or 0 to retrieve without change

## Returns

the counter after change

**6.1.3.10 void Accelerometer::updateValues()** [static], [private]

Updater function delivered to the device-handling thread.

## 6.1.4 Member Data Documentation

- **6.1.4.1 std::ifstream Accelerometer::device** [static], [private]
- **6.1.4.2 std::string Accelerometer::devicePath** [static], [private]
- **6.1.4.3** bool Accelerometer::done = false [static], [private]
- **6.1.4.4 std::mutex Accelerometer::mtx** [static], [private]
- **6.1.4.5** bool Accelerometer::present = false [static], [private]
- **6.1.4.6** double Accelerometer::sensitivity = 1.0 [static], [private]
- **6.1.4.7 std::thread Accelerometer::updaterThread** [static], [private]
- **6.1.4.8 double Accelerometer::x = 0.0** [static], [private]
- **6.1.4.9** double Accelerometer::y = 0.0 [static], [private]
- **6.1.4.10** double Accelerometer::z = 0.0 [static], [private]

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

- · src/Accelerometer.h
- src/Accelerometer.cpp

#### **Image Struct Reference** 6.2

Collaboration diagram for Image:

## Image

- + imagic
- + type
- + dim
- + sizeX
- + sizeY
- + sizeZ + min
- + max
- + wasteBytes
- + name
- and 6 more...

## **Public Attributes**

- · unsigned short imagic
- · unsigned short type
- · unsigned short dim
- unsigned short sizeX
- unsigned short sizeY
- unsigned short sizeZ
- unsigned int min
- · unsigned int max
- unsigned int wasteBytes
- char name [80]
- unsigned int colorMap
- FILE \* file
- unsigned char \* tmp [5]
- unsigned int rleEnd
- unsigned int \* rowStart
- unsigned int \* rowSize

#### 6.2.1 **Member Data Documentation**

- 6.2.1.1 unsigned int Image::colorMap
- 6.2.1.2 unsigned short Image::dim
- 6.2.1.3 FILE\* Image::file
- 6.2.1.4 unsigned short Image::imagic

24 Class Documentation

6.2.1.5 unsigned int Image::max 6.2.1.6 unsigned int Image::min char Image::name[80] 6.2.1.7 6.2.1.8 unsigned int Image::rleEnd 6.2.1.9 unsigned int\* Image::rowSize 6.2.1.10 unsigned int \* Image::rowStart 6.2.1.11 unsigned short Image::sizeX 6.2.1.12 unsigned short Image::sizeY 6.2.1.13 unsigned short Image::sizeZ 6.2.1.14 unsigned char\* Image::tmp[5] 6.2.1.15 unsigned short Image::type

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

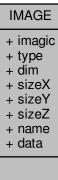
• src/SGlimage.c

## 6.3 IMAGE Struct Reference

6.2.1.16 unsigned int Image::wasteBytes

#include <SGIimage.h>

Collaboration diagram for IMAGE:



6.4 Mass Class Reference 25

# **Public Attributes**

- · unsigned short imagic
- unsigned short type
- unsigned short dim
- unsigned short sizeX
- unsigned short sizeY
- unsigned short sizeZ
- char name [128]
- unsigned char \* data

# 6.3.1 Member Data Documentation

- 6.3.1.1 unsigned char\* IMAGE::data
- 6.3.1.2 unsigned short IMAGE::dim
- 6.3.1.3 unsigned short IMAGE::imagic
- 6.3.1.4 char IMAGE::name[128]
- 6.3.1.5 unsigned short IMAGE::sizeX
- 6.3.1.6 unsigned short IMAGE::sizeY
- 6.3.1.7 unsigned short IMAGE::sizeZ
- 6.3.1.8 unsigned short IMAGE::type

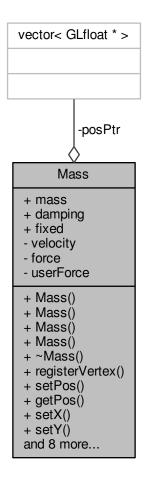
The documentation for this struct was generated from the following file:

• src/SGlimage.h

# 6.4 Mass Class Reference

#include <Mass.h>

## Collaboration diagram for Mass:



## **Public Member Functions**

- Mass (std::vector< GLfloat \* > posPtr=vector< GLfloat \* >(), double mass=0.0, double damp=0.0)
- Mass (double mass, double damp=0.0)
- Mass (const Mass &m)
- Mass (Mass &&)
- virtual ∼Mass ()
- void registerVertex (GLfloat \*ptr)
- void setPos (Eigen::Vector3d p)
- Eigen::Vector3d getPos ()
- void setX (double x)
- void setY (double y)
- void setZ (double z)
- void setVel (Eigen::Vector3d v)
- Eigen::Vector3d getVel ()
- void setForce (Eigen::Vector3d f)
- Eigen::Vector3d getForce ()
- void addForce (Eigen::Vector3d f)
- void setUserForce (Eigen::Vector3d f)
- Eigen::Vector3d getUserForce ()

6.4 Mass Class Reference 27

# **Public Attributes**

- · double mass
- · double damping
- bool fixed

### **Private Attributes**

- vector< GLfloat \* > posPtr
- Eigen::Vector3d velocity
- Eigen::Vector3d force
- Eigen::Vector3d userForce

# 6.4.1 Detailed Description

This class contains all properties of a mass-point which are necessary to run a physically based simulation of a mass-spring system.

## 6.4.2 Constructor & Destructor Documentation

6.4.2.1 Mass::Mass ( std::vector < GLfloat \* > posPtr = vector < GLfloat \* > (), double mass = 0.0, double damp = 0.0)

### Default constructor.

### **Parameters**

posPtr	vector of pointers to the vertices in the positions array
mass	the mass of the mass-point
damp	the damping factor of the mass-point (to simulate friction and air resistance)

# 6.4.2.2 Mass::Mass ( double mass, double damp = 0.0 )

Constructor which takes at least the mass argument.

### **Parameters**

mass	the mass of the mass-point
damp	the damping factor of the mass-point (to simulate friction and air resistance)

# 6.4.2.3 Mass::Mass ( const Mass & m )

## Copy constructor

### **Parameters**

m	source object

## 6.4.2.4 Mass::Mass ( Mass && m )

Move Constructor

**Parameters** m **6.4.2.5** Mass::∼Mass( ) [virtual] Default destructor. 6.4.3 Member Function Documentation 6.4.3.1 void Mass::addForce ( Eigen::Vector3d f ) Add a force to the stored value. **Parameters** force vector 6.4.3.2 Eigen::Vector3d Mass::getForce ( ) Get the force. Returns force vector 6.4.3.3 Eigen::Vector3d Mass::getPos() Get position vector of mass. Don't write to the elements of the returned vector. This will lead to inconsistencies in the vertices array. To set a single value use setX(), setY() and setZ() instead. Returns the position vector or if no vertices are registered the 0 vector. 6.4.3.4 Eigen::Vector3d Mass::getUserForce ( ) Get external forces (usually without gravity). Returns force vector 6.4.3.5 Eigen::Vector3d Mass::getVel()

Get the velocity.

velocity vector

Returns

6.4 Mass Class Reference 29

6.4.3.6 void Mass::registerVertex ( GLfloat \* ptr )

Register the pointer to a vertex as position.

### **Parameters**

ptr	pointer targeting the x-coordinate of the vertex position. The y- and z-coordinates must be in
	consecutive order in the array.

6.4.3.7 void Mass::setForce ( Eigen::Vector3d f )

Set the force.

This vector is reserved for the solver. Use setUserForce() to apply edxternal forces.

**Parameters** 

f	force vector.

6.4.3.8 void Mass::setPos ( Eigen::Vector3d pos )

Set position of the mass object.

This setter spreads the position to all registered vertices.

**Parameters** 

pos	the position vector
-----	---------------------

6.4.3.9 void Mass::setUserForce ( Eigen::Vector3d f )

Set external forces.

Use this setter to apply all external forces except the gravity, the solver applies it separately.

**Parameters** 

f	force vector

6.4.3.10 void Mass::setVel ( Eigen::Vector3d v )

Set the velocity.

**Parameters** 

ν	velocity vector

6.4.3.11 void Mass::setX ( double x )

Set x-coordinate of the mass object.

This setter spreads the x-coordinate to all registered vertices. Don't use getPos.x to write a coordinate, it will mess up the vertices array.

**Parameters** 

X	the coordinate

6.4.3.12 void Mass::setY ( double *y* )

Set y-coordinate of the mass object.

This setter spreads the y-coordinate to all registered vertices. Don't use getPos.y to write a coordinate, it will mess up the vertices array.

### **Parameters**

V	the coordinate
У	the coordinate

### 6.4.3.13 void Mass::setZ ( double z )

Set z-coordinate of the mass object.

This setter spreads the z-coordinate to all registered vertices. Don't use getPos.z to write a coordinate, it will mess up the vertices array.

### **Parameters**

Z	the coordinate

### 6.4.4 Member Data Documentation

- 6.4.4.1 double Mass::damping
- 6.4.4.2 bool Mass::fixed
- **6.4.4.3 Eigen::Vector3d Mass::force** [private]
- 6.4.4.4 double Mass::mass
- **6.4.4.5 vector**<**GLfloat**\*> **Mass::posPtr** [private]
- **6.4.4.6 Eigen::Vector3d Mass::userForce** [private]
- **6.4.4.7 Eigen::Vector3d Mass::velocity** [private]

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

- src/Mass.h
- src/Mass.cpp

# 6.5 std::MipMap Class Reference

#include <MipMap.h>

Collaboration diagram for std::MipMap:

## std::MipMap

- texture
- width
- height
- + MipMap()
- + ~MipMap() + loadPNG()
- + loadSGI()
- + load()

### **Public Member Functions**

- MipMap ()
- virtual ∼MipMap ()
- void loadPNG (const string fileName)

load texture from PNG

void loadSGI (const string fileName)

load texture from PNG

void load (const string fileName)

load a texture with a selected import filter

## **Private Attributes**

- GLuint texture = 0
- size t width = 0
- size t height = 0

# 6.5.1 Detailed Description

This class is a container for a texture mipmap.

It also handles the loading of an image file.

Supported file formats:

PNG images with three colors and no alpha channel SGI RGB images with uncompressed data and without alpha channel.

#### 6.5.2 **Constructor & Destructor Documentation**

6.5.2.1 std::MipMap::MipMap()

Default constructor.

6.5.2.2 std::MipMap::~MipMap( ) [virtual]

Default destructor.

## 6.5.3 Member Function Documentation

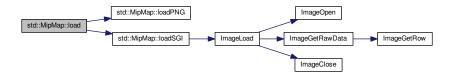
6.5.3.1 void std::MipMap::load ( const string fileName )

load a texture with a selected import filter

### **Parameters**

fileName	the image file to be loaded	

Here is the call graph for this function:



## 6.5.3.2 void std::MipMap::loadPNG ( const string fileName )

## load texture from PNG

# **Parameters**

fileName	the png file to be loaded
----------	---------------------------

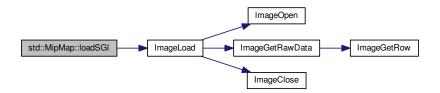
### 6.5.3.3 void std::MipMap::loadSGI ( const string fileName )

## load texture from PNG

# **Parameters**

fileName	the SGI-rgb file to be loaded
----------	-------------------------------

Here is the call graph for this function:



# 6.5.4 Member Data Documentation

- **6.5.4.1** size\_t std::MipMap::height = 0 [private]
- **6.5.4.2 GLuint std::MipMap::texture = 0** [private]
- 6.5.4.3 size\_t std::MipMap::width = 0 [private]

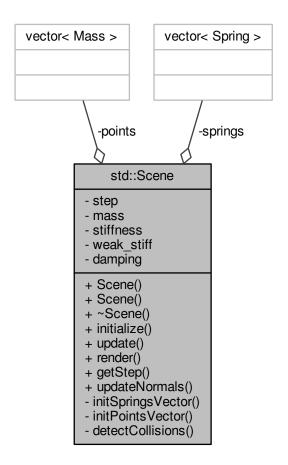
The documentation for this class was generated from the following files:

- src/MipMap.h
- src/MipMap.cpp

# 6.6 std::Scene Class Reference

#include <Scene.h>

Collaboration diagram for std::Scene:



# **Public Member Functions**

- Scene ()
- Scene (int argc, char \*argv[])
- virtual ∼Scene ()
- void initialize ()
- void update ()
- void render ()
- double getStep () const
- void updateNormals ()

# **Private Member Functions**

- void initSpringsVector (size\_t i)
- void initPointsVector (size\_t i, size\_t \*n)
- void detectCollisions ()

## **Private Attributes**

- double step
- · double mass
- · double stiffness
- · double weak\_stiff
- · double damping
- vector< Mass > points
- vector< Spring > springs

# 6.6.1 Detailed Description

This is a class which handles all tasks to set up, simulate and render a scene with a physical model.

Usually one scene object instance exists program-wide.

## 6.6.2 Constructor & Destructor Documentation

```
6.6.2.1 std::Scene::Scene()
```

Default constructor

```
6.6.2.2 std::Scene::Scene ( int argc, char * argv[] )
```

Constructor which initializes the objectives with values given by command line parameters.

### **Parameters**

argc	arguments counter supplied by the main function
argv	arguments vector of length argc

```
6.6.2.3 std::Scene::∼Scene( ) [virtual]
```

Default destructor.

## 6.6.3 Member Function Documentation

```
6.6.3.1 void std::Scene::detectCollisions() [private]
```

Call the collision detection algorithm for each 3D-object with mass.

Here is the call graph for this function:



6.6.3.2 double std::Scene::getStep ( ) const

Get the time step for the simulation.

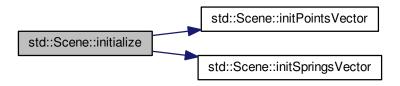
Returns

step in seconds

6.6.3.3 void std::Scene::initialize ( )

Initialize the scene for rendering and physical simulation.

Here is the call graph for this function:



6.6.3.4 void std::Scene::initPointsVector( size\_t i, size\_t \* n ) [private]

Initialize all mass-points.

This method connects all mass-points processed by the physics simulation to the vertices in the ...Positions vector for the renderer. The vector of points must be filled with the correct number of Mass objects initialized with default values.

## **Parameters**

i	the i-th 3d-object
n	a pointer to a counter value, it is incremented with each initialized mass-point. it must be
	initialized with 0 before the first call of this method.

**6.6.3.5 void std::Scene::initSpringsVector(size\_t** *i* **)** [private]

Initialize structural (strong) and bending (weak) springs. No diagonal springs, because of triangular grid Parameters

i	the i-th 3D-object to process

6.6.3.6 void std::Scene::render ( )

Render the scene.

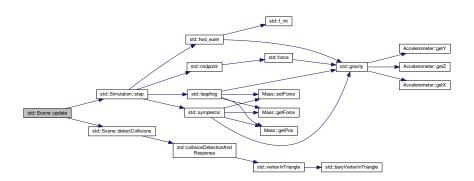
Here is the call graph for this function:



6.6.3.7 void std::Scene::update ( )

Update the simulation step and collision detection.

Here is the call graph for this function:



6.6.3.8 void std::Scene::updateNormals ( )

Recalculate all normals for all faces modified by the simulaton step.

## 6.6.4 Member Data Documentation

- **6.6.4.1** double std::Scene::damping [private]
- **6.6.4.2 double std::Scene::mass** [private]
- **6.6.4.3** vector<Mass> std::Scene::points [private]
- **6.6.4.4 vector**<**Spring**> **std::Scene::springs** [private]
- **6.6.4.5** double std::Scene::step [private]
- **6.6.4.6 double std::Scene::stiffness** [private]
- **6.6.4.7** double std::Scene::weak\_stiff [private]

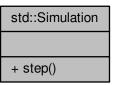
The documentation for this class was generated from the following files:

- · src/Scene.h
- src/Scene.cpp

# 6.7 std::Simulation Class Reference

#include <Simulation.h>

Collaboration diagram for std::Simulation:



# **Public Types**

• enum Method { EULER, SYMPLECTIC, LEAPFROG, MIDPOINT }

# **Static Public Member Functions**

static void step (double dt, Method method, vector < Mass > &points, vector < Spring > &springs)

# 6.7.1 Detailed Description

Simulation class. Solver for mass-spring systems.

This class provides some numerical solvers. They are programmed during a assignment of the pro-seminar physically based simulations preceding to this project.

## 6.7.2 Member Enumeration Documentation

6.7.2.1 enum std::Simulation::Method

Enumerator

EULER

**SYMPLECTIC** 

**LEAPFROG** 

**MIDPOINT** 

## 6.7.3 Member Function Documentation

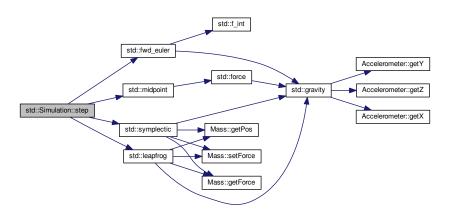
6.7.3.1 void std::Simulation::step ( double dt, Method method, vector < Mass > & points, vector < Springs ) [static]

Execute one step of the simulation.

### **Parameters**

dt	time delay between two steps
method	select the method to use; The value can be EULER, SYMPLECTIC, LEAPFROG or MIDP↔
	OINT.
points	all mass-points
springs	all springs

Here is the call graph for this function:



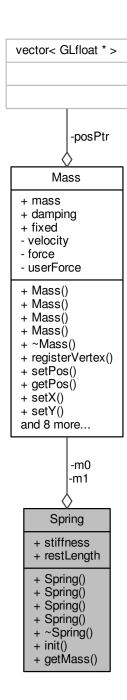
The documentation for this class was generated from the following files:

- src/Simulation.h
- src/Simulation.cpp

# 6.8 Spring Class Reference

#include <Spring.h>

Collaboration diagram for Spring:



### **Public Member Functions**

- Spring (double stiff=0.0, double restLen=0.0)
- Spring (Mass \*mass0, Mass \*mass1, double stiff=0.0, double restLen=0.0)
- Spring (Spring &s)
- Spring (Spring &&s)
- virtual ∼Spring (void)

- void init (Mass \*mass0, Mass \*mass1)
- Mass \* getMass (int i)

## **Public Attributes**

- · double stiffness
- · double restLength

## **Private Attributes**

- Mass \* m0
- Mass \* m1

## 6.8.1 Detailed Description

This class contains all properties of a spring which are necessary to run a physically based simulation of a mass-spring system.

## 6.8.2 Constructor & Destructor Documentation

6.8.2.1 Spring::Spring ( double stiff = 0.0, double restLen = 0.0 )

Default constructor.

The end-points are left uninitialized, so after calling this constructor a call to init () is necessary.

### **Parameters**

s	tiff	stiffness factor of the spring
restL	en	the rest length = length of the spring without any forces taking effect to it

## 6.8.2.2 Spring::Spring ( Mass \* mass0, Mass \* mass1, double stiff = 0 . 0, double restLen = 0 . 0 )

Constructor which initializes the endpoints of the spring with two mass-points

# **Parameters**

mass0	mass-point on the first end of the spring
mass1	mass-point on the second end
stiff	stiffness factor of the spring
restLen	the rest length = length of the spring without any forces taking effect to it

# 6.8.2.3 Spring::Spring (Spring & s)

## Copy Constructor

## **Parameters**

S	source spring

# 6.8.2.4 Spring::Spring (Spring && s)

Move constructor

### **Parameters**

s	source spring
---	---------------

**6.8.2.5 Spring::∼Spring(void)** [virtual]

Default destructor.

## 6.8.3 Member Function Documentation

6.8.3.1 Mass \* Spring::getMass ( int i )

Get a mass.

**Parameters** 

i	0 or 1 to select the mass

### Returns

the selected mass

6.8.3.2 void Spring::init ( Mass \* mass0, Mass \* mass1 )

Initialize spring with two given masses and set the rest length to the distance of the two points.

## **Parameters**

mass0	mass at end-point
mass1	mass at end-point

Here is the call graph for this function:



## 6.8.4 Member Data Documentation

**6.8.4.1 Mass\* Spring::m0** [private]

**6.8.4.2 Mass** \* Spring::m1 [private]

6.8.4.3 double Spring::restLength

6.8.4.4 double Spring::stiffness

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

• src/Spring.h

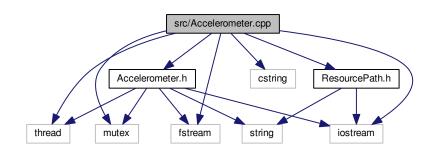
• src/Spring.cpp

# **Chapter 7**

# **File Documentation**

# 7.1 src/Accelerometer.cpp File Reference

```
#include <thread>
#include <mutex>
#include <iostream>
#include <fstream>
#include <cstring>
#include "ResourcePath.h"
#include "Accelerometer.h"
Include dependency graph for Accelerometer.cpp:
```



# **Macros**

• #define \_PRIV\_LINE\_LEN 100

# 7.1.1 Macro Definition Documentation

7.1.1.1 #define \_PRIV\_LINE\_LEN 100

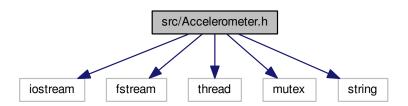
# 7.2 src/Accelerometer.h File Reference

#include <iostream>

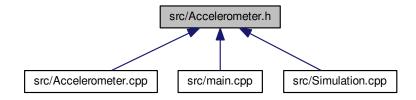
48 File Documentation

```
#include <fstream>
#include <thread>
#include <mutex>
#include <string>
```

Include dependency graph for Accelerometer.h:



This graph shows which files directly or indirectly include this file:



### Classes

• class Accelerometer

# Macros

• #define CFG\_FILENAME "accelerometer.config"

## 7.2.1 Macro Definition Documentation

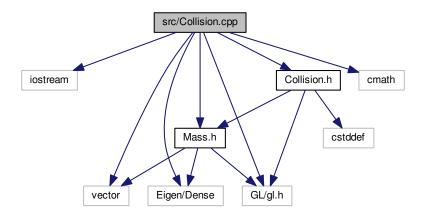
7.2.1.1 #define CFG\_FILENAME "accelerometer.config"

This definition contains the hard-coded path to the device-config-file. TODO. do it in a generic way - some changes may be necessary to maintain multiple devices

# 7.3 src/Collision.cpp File Reference

```
#include <iostream>
#include <vector>
#include <cmath>
#include <Eigen/Dense>
#include <GL/gl.h>
#include "Collision.h"
#include "Mass.h"
```

Include dependency graph for Collision.cpp:



## **Namespaces**

• std

## **Functions**

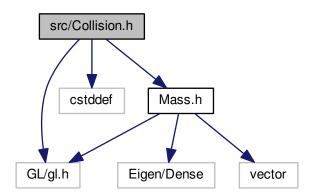
- bool std::vertexInTriangle (const Eigen::Vector3d &P, const Eigen::Vector3d &A, const Eigen::Vector3d &B, const Eigen::Vector3d &C, const float epsilon, float &dist, Eigen::Vector3d &N)
- bool std::baryVertexInTriangle (const Eigen::Vector3d &P, const Eigen::Vector3d &A, const Eigen::Vector3d &B, const Eigen::Vector3d &C)
- void std::collisionDetectionAndResponse (vector < Mass > &points, size\_t offsP, size\_t lenP, GLfloat object ←
   \_mesh[], size\_t offsO, size\_t lenO)

# 7.4 src/Collision.h File Reference

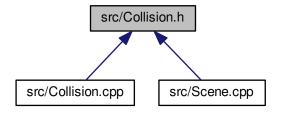
```
#include <GL/gl.h>
#include <cstddef>
#include "Mass.h"
```

50 File Documentation

Include dependency graph for Collision.h:



This graph shows which files directly or indirectly include this file:



# **Namespaces**

• std

# **Functions**

void std::collisionDetectionAndResponse (vector < Mass > &points, size\_t offsP, size\_t lenP, GLfloat object ← mesh[], size\_t offsO, size\_t lenO)

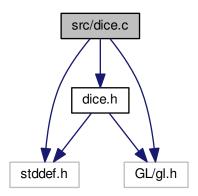
# 7.4.1 Detailed Description

This is a header file to include the functions needed for the collision detection algorithm.

## 7.5 src/dice.c File Reference

```
#include <stddef.h>
#include <GL/gl.h>
#include "dice.h"
```

Include dependency graph for dice.c:



## **Variables**

- const size\_t diceVertices = 78
- GLfloat dicePositions [234]
- GLfloat diceTexels [156]
- GLfloat diceNormals [234]
- const size\_t diceObjectsWithMass = 2
- const size\_t diceMasses [2]
- const size\_t diceMassFwdOffs [2]
- const size t diceMassVertices [2]
- const size t diceMassRevOffs [2]
- const size\_t diceMassRevOffsOrig [2]
- const size\_t diceFwdIndexI [72]
- const size\_t \* diceFwdIndex [16]
- const size\_t diceFwdIndexLength [16]
- const size\_t diceRevIndex [72]
- const size\_t diceObjects = 3
- const size\_t diceObjectOffset [3]
- const size\_t diceObjectLength [3]
- const char diceObjectNamesString [36]
- const char \* diceObjectNames [3]
- const char diceTextureFilePath [34] = "/not\_implemented\_yet/edit/by.hand"

## 7.5.1 Detailed Description

This is a C-source file (.c) for the models "dice\_Plane", "dice\_RollingDie", "dice\_Die" Don't edit! This is an autogenerated file by blender2oGL. Modifications are not permanent.

52 File Documentation

## 7.5.2 Variable Documentation

## 7.5.2.1 const size\_t\* diceFwdIndex[16]

### Initial value:

### 7.5.2.2 const size\_t diceFwdIndexI[72]

### Initial value:

### 7.5.2.3 const size\_t diceFwdIndexLength[16]

# Initial value:

```
= {
    5, 4, 5, 4, 5, 4, 5, 4, 5, 4, 5, 4, 5, 4, 5, 4, 5, 4, 5, 4, 5, 4, 5, 4, }
```

7.5.2.4 const size\_t diceMasses[2]

Initial value:

```
= {
    8,
8,
```

7.5.2.5 const size\_t diceMassFwdOffs[2]

Initial value:

```
= {
0,
8,
```

7.5.2.6 const size\_t diceMassRevOffs[2]

Initial value:

```
= {
    0,
    36,
```

7.5.2.7 const size\_t diceMassRevOffsOrig[2]

Initial value:

```
= {
    6,
    42,
}
```

7.5.2.8 const size\_t diceMassVertices[2]

Initial value:

```
= {
36,
36,
```

7.5.2.9 GLfloat diceNormals[234]

7.5.2.10 const size\_t diceObjectLength[3]

Initial value:

```
= {
    6,
    36,
    36,
```

54 File Documentation

## 7.5.2.11 const char\* diceObjectNames[3]

### Initial value:

```
= {
    &diceObjectNamesString[0],
    &diceObjectNamesString[11],
    &diceObjectNamesString[27],
}
```

7.5.2.12 const char diceObjectNamesString[36]

### Initial value:

```
= {
    "dice_Plane\0"
    "dice_RollingDie\0"
    "dice_Die\0"
}
```

7.5.2.13 const size\_t diceObjectOffset[3]

### Initial value:

```
= {
    0,
    6,
    42,
```

7.5.2.14 const size\_t diceObjects = 3

7.5.2.15 const size\_t diceObjectsWithMass = 2

7.5.2.16 GLfloat dicePositions[234]

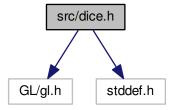
7.5.2.17 const size\_t diceRevIndex[72]

## Initial value:

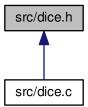
- 7.5.2.18 GLfloat diceTexels[156]
- 7.5.2.19 const char diceTextureFilePath[34] = "/not\_implemented\_yet/edit/by.hand"
- 7.5.2.20 const size\_t diceVertices = 78

# 7.6 src/dice.h File Reference

#include <GL/gl.h>
#include <stddef.h>
Include dependency graph for dice.h:



This graph shows which files directly or indirectly include this file:



## **Variables**

- const size\_t diceVertices
- GLfloat dicePositions [234]
- GLfloat diceTexels [156]
- GLfloat diceNormals [234]
- const size\_t diceObjectsWithMass
- const size\_t diceMasses [2]
- const size\_t diceMassFwdOffs [2]
- const size\_t diceMassVertices [2]
- const size\_t diceMassRevOffs [2]

56 File Documentation

- const size\_t diceMassRevOffsOrig [2]
- const size\_t diceFwdIndexI [72]
- const size t \* diceFwdIndex [16]
- const size\_t diceFwdIndexLength [16]
- const size\_t diceRevIndex [72]
- const size t diceObjects
- const size\_t diceObjectOffset [3]
- const size\_t diceObjectLength [3]
- const char diceObjectNamesString [36]
- const char \* diceObjectNames [3]
- const char diceTextureFilePath [34]

# 7.6.1 Detailed Description

This is a C-header file (.h) for the models "dice\_Plane", "dice\_RollingDie", "dice\_Die" Don't edit! This is an autogenerated file by blender2oGL. Modifications are not permanent.

### 7.6.2 Variable Documentation

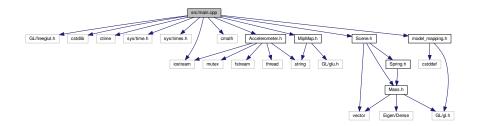
- 7.6.2.1 const size\_t\* diceFwdIndex[16]
- 7.6.2.2 const size\_t diceFwdIndexI[72]
- 7.6.2.3 const size\_t diceFwdIndexLength[16]
- 7.6.2.4 const size\_t diceMasses[2]
- 7.6.2.5 const size\_t diceMassFwdOffs[2]
- 7.6.2.6 const size\_t diceMassRevOffs[2]
- 7.6.2.7 const size\_t diceMassRevOffsOrig[2]
- 7.6.2.8 const size\_t diceMassVertices[2]
- 7.6.2.9 GLfloat diceNormals[234]
- 7.6.2.10 const size\_t diceObjectLength[3]
- 7.6.2.11 const char\* diceObjectNames[3]
- 7.6.2.12 const char diceObjectNamesString[36]
- 7.6.2.13 const size\_t diceObjectOffset[3]
- 7.6.2.14 const size\_t diceObjects
- 7.6.2.15 const size\_t diceObjectsWithMass
- 7.6.2.16 GLfloat dicePositions[234]
- 7.6.2.17 const size\_t diceRevIndex[72]
- 7.6.2.18 GLfloat diceTexels[156]

### 7.6.2.19 const char diceTextureFilePath[34]

7.6.2.20 const size\_t diceVertices

# 7.7 src/main.cpp File Reference

```
#include <GL/freeglut.h>
#include <cstdlib>
#include <ctime>
#include <sys/time.h>
#include <sys/times.h>
#include <iostream>
#include <cmath>
#include "Accelerometer.h"
#include "Scene.h"
#include "MipMap.h"
#include "model_mapping.h"
Include dependency graph for main.cpp:
```



## **Functions**

- ulong getTime ()
- void initialize ()
- void motionCallback (int, int)
- void reshapeCallback (int w, int h)
- void updateSimulation ()
- void displayCallback ()
- void keyboardCallback (unsigned char key, int, int)
- void mouseCallback (int button, int state, int x, int y)
- void idleCallback ()
- int main (int argc, char \*argv[])

# **Variables**

- Scene \* scene =nullptr
- static const double MAX\_UPDATE\_TIME = 0.01

## 7.7.1 Function Documentation

## 7.7.1.1 void displayCallback ( )

Display callback function. Called each times the display needs to be redrawn.

58 File Documentation

Here is the call graph for this function:



# 7.7.1.2 ulong getTime ( )

Get the system time. This function requires \*NIX dependent system functions.

### Returns

the time of now

# 7.7.1.3 void idleCallback ( )

Idle function. Executed if OpenGL is in an idle state.

# 7.7.1.4 void initialize ( )

Initialize the OpenGL system and the scene.

Here is the call graph for this function:



# 7.7.1.5 void keyboardCallback (unsigned char key, int, int)

Keyboard callback function. Processes the input events from the keyboard.

# **Parameters**

key key mapped to an event
----------------------------

# 7.7.1.6 int main ( int argc, char \* argv[])

The main function.

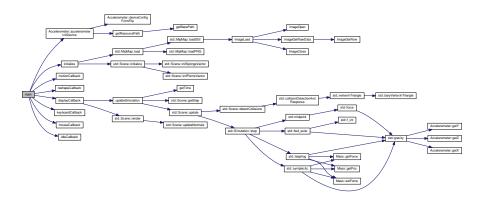
### **Parameters**

argc	arguments counter.
argv	arguments vector. All arguments are passed on to glutInit() and Scene()

### Returns

process return value EXIT\_SUCCESS or EXIT\_FAILURE

Here is the call graph for this function:



# 7.7.1.7 void motionCallback (int, int)

Mouse Motion callback function Processes the input events from the Mouse. actually unused

# 7.7.1.8 void mouseCallback (int button, int state, int x, int y)

Mouse button callback function. Processes the input events from the Mouse.

### **Parameters**

button	
state	
X	
у	

# 7.7.1.9 void reshape Callback ( int w, int h )

Window reshape callback function.

### **Parameters**

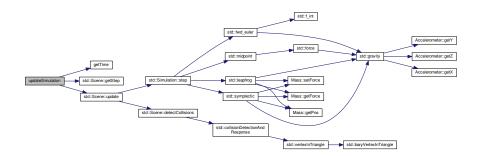
W	window width after size change
h	window height after size change

## 7.7.1.10 void updateSimulation ( )

Update the simulation up to the present time.

60 File Documentation

Here is the call graph for this function:



## 7.7.2 Variable Documentation

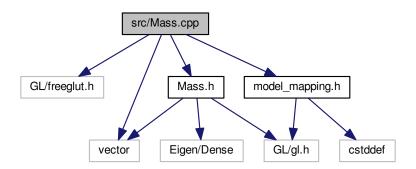
**7.7.2.1** const double MAX\_UPDATE\_TIME = 0.01 [static]

Maximum time allowed between updates when running in real-time mode (prevents performing too many calculations when running slower than real time)

## 7.7.2.2 Scene \* scene = nullptr

# 7.8 src/Mass.cpp File Reference

```
#include <GL/freeglut.h>
#include <vector>
#include "Mass.h"
#include "model_mapping.h"
Include dependency graph for Mass.cpp:
```



## **Macros**

- #define \_DEBUG\_MSG\_C(X)
- #define \_DEBUG\_MSG(X)

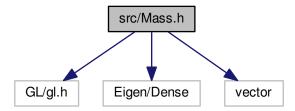
## 7.8.1 Macro Definition Documentation

- 7.8.1.1 #define \_DEBUG\_MSG( *X* )
- 7.8.1.2 #define \_DEBUG\_MSG\_C( X )

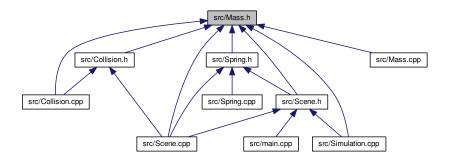
## 7.9 src/Mass.h File Reference

```
#include <GL/gl.h>
#include <Eigen/Dense>
#include <vector>
```

Include dependency graph for Mass.h:



This graph shows which files directly or indirectly include this file:



## Classes

• class Mass

## 7.10 src/MipMap.cpp File Reference

#include "MipMap.h"

```
#include <GL/glu.h>
#include <png.h>
#include <iostream>
#include "SGIimage.h"
Include dependency graph for MipMap.cpp:
```

src/MipMap.cpp

MipMap.h png.h iostream SGlimage.h

string

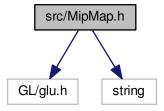
GL/glu.h

## **Namespaces**

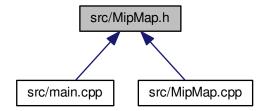
• std

# 7.11 src/MipMap.h File Reference

#include <GL/glu.h>
#include <string>
Include dependency graph for MipMap.h:



This graph shows which files directly or indirectly include this file:



#### Classes

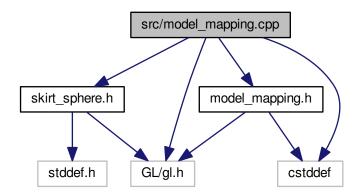
• class std::MipMap

## **Namespaces**

• std

## 7.12 src/model\_mapping.cpp File Reference

```
#include <GL/gl.h>
#include <cstddef>
#include "model_mapping.h"
#include "skirt_sphere.h"
Include dependency graph for model_mapping.cpp:
```



### **Namespaces**

• std

#### **Macros**

#define MODEL(X) skirt\_sphere##X

path to the texture-image file

### **Variables**

```
    const size_t std::model3dVertices = MODEL(Vertices)

     number of vertices

    GLfloat * std::model3dPositions = MODEL(Positions)

     all vertex positions

    GLfloat * std::model3dTexels = MODEL(Texels)

     all texture coordinates

    GLfloat * std::model3dNormals = MODEL(Normals)

     all normals of the face surfaces

    const size_t std::model3dObjectsWithMass = MODEL(ObjectsWithMass)

     number of objects to apply to the mass-spring simulation

    const size_t * std::model3dMasses = MODEL(Masses)

     array of 3D-objects with mass

    const size_t * std::model3dMassFwdOffs = MODEL(MassFwdOffs)

     offset in the points array

    const size_t * std::model3dMassVertices = MODEL(MassVertices)

     number of vertices per object with mass
• const size_t * std::model3dMassRevOffs = MODEL(MassRevOffs)
     offsets for the model3dRevIndex array

    const size_t * std::model3dMassRevOffsOrig = MODEL(MassRevOffsOrig)

     offset to the first vertex of an object with mass

    const size_t ** std::model3dFwdIndex = MODEL(FwdIndex)

     indices of the mass-points in the positions array

    const size_t * std::model3dFwdIndexLength = MODEL(FwdIndexLength)

     number of indices for each mass-point

    const size t * std::model3dRevIndex = MODEL(RevIndex)

     index of a vertex in the mass-points array
• const size_t std::model3dObjects = MODEL(Objects)
     number of 3D-objects

    const size_t * std::model3dObjectOffset = MODEL(ObjectOffset)

     offset to the first vertex of a 3D-object

    const size_t * std::model3dObjectLength = MODEL(ObjectLength)

     number of vertices for each 3D-object

    const char ** std::model3dObjectNames = (const char**) MODEL(ObjectNames)

     names of the objects (for identification)

    const char * std::model3dTextureFilePath = "textures/textures all.rgb"
```

### 7.12.1 Detailed Description

This file includes the vertex, texture and normals coordinates and some 3D object topology informations.

It is a wrapper to make the code relying on these variables independent from varying names supplied by the header file generator tool blender2oGL. Use the following line to generate a customized header file:

```
blender2oGL ./path/to-your/file.obj -m -s -o -r -a file_objectWithMass -a file_otherObjectIfDesired
```

Edit the pointers in this file to fit the code to an exchanged header file. DON'T EDIT them in the generated .h file, they may be overwritten by the generator tool.

#### 7.12.2 Macro Definition Documentation

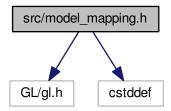
7.12.2.1 #define MODEL( X ) skirt\_sphere##X

define the prefix of the structures for mapping the variable names

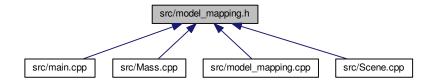
## 7.13 src/model\_mapping.h File Reference

```
#include <GL/gl.h>
#include <cstddef>
```

Include dependency graph for model\_mapping.h:



This graph shows which files directly or indirectly include this file:



## **Namespaces**

std

## 7.13.1 Detailed Description

This file includes the vertex, texture and normals coordinates and some 3D object topology informations.

It is a wrapper to make the code relying on these variables independent from varying names supplied by the header file generator tool 'blender2oGL'

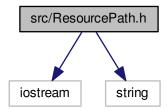
Edit the pointers in the 'model\_mapping.cpp' file to fit the code to an exchanged header file. DON'T EDIT them in the generated .h file, they may be overwritten by the generator tool.

See 'model\_mapping.cpp' for further details.

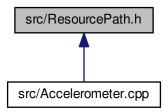
## 7.14 src/ResourcePath.h File Reference

#include <iostream>
#include <string>

Include dependency graph for ResourcePath.h:



This graph shows which files directly or indirectly include this file:



### **Functions**

- string getBasePath (int, char \*argv[]=nullptr)
   Get the path of the application folder.
- string getResourcePath (string filePath, string resourcesFolder="resources", int argc=0, char \*argv[]=nullptr)

  Get path to resource file relative to the BasePath().

### 7.14.1 Function Documentation

7.14.1.1 string getBasePath (int, char \* argv[] = nullptr)

Get the path of the application folder.

The path returned is an absolute path with trailing path separator. The first call of this function initializes the path constant and thus, it needs non-null values for argc and argv. If the initialization fails, the return value is an empty string.

#### **Parameters**

argv	the arguments list provided by the main () function on first call, omit or set to nullptr on
	succeeding calls

#### Returns

the base path or an empty string if not initialized.

7.14.1.2 string getResourcePath ( string filePath, string resourcesFolder = "resources", int argc = 0, char \* argv[] = nullptr )

Get path to resource file relative to the BasePath().

The path returned is an absolute path resolved from ../resources/filePath. If the path to a sub-folder is required use the path separator supported by the target system.

#### **Parameters**

<i></i> 5	
filePath	the path to the requested file in the resources folder
resourcesFolder	name of the folder without path separators or empty string to address the parent folder, default
	is "resources"
argc	the arguments count of the main () function on first call
argv	the arguments list of the main () function on first call

#### Returns

the base path to the resource folder appended by filePath

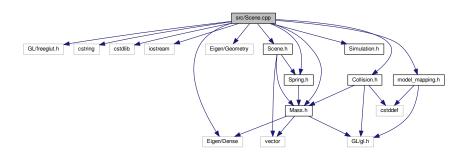
Here is the call graph for this function:



## 7.15 src/Scene.cpp File Reference

#include <GL/freeglut.h>

```
#include <cstring>
#include <cstdlib>
#include <iostream>
#include <Eigen/Dense>
#include <Eigen/Geometry>
#include "Scene.h"
#include "Mass.h"
#include "Spring.h"
#include "Simulation.h"
#include "Collision.h"
#include "model_mapping.h"
Include dependency graph for Scene.cpp:
```



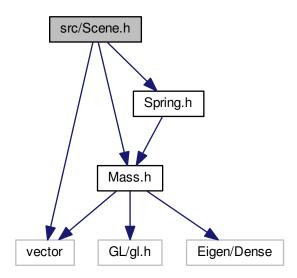
## **Namespaces**

• std

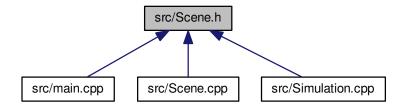
## 7.16 src/Scene.h File Reference

```
#include <vector>
#include "Mass.h"
#include "Spring.h"
```

Include dependency graph for Scene.h:



This graph shows which files directly or indirectly include this file:



## Classes

• class std::Scene

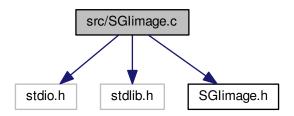
## **Namespaces**

• std

# 7.17 src/SGlimage.c File Reference

#include <stdio.h>

```
#include <stdlib.h>
#include "SGIimage.h"
Include dependency graph for SGIimage.c:
```



## **Classes**

· struct Image

### **Macros**

- #define IMAGIC 0x01da
- #define IMAGIC SWAP 0xda01
- #define SWAP\_SHORT\_BYTES(x) ((((x) & 0xff) << 8) | (((x) & 0xff00) >> 8))
- #define SWAP\_LONG\_BYTES(x)

#### **Functions**

- static Image \* ImageOpen (char \*fileName)
- static void ImageClose (Image \*image)
- static void ImageGetRow (Image \*image, unsigned char \*buf, int y, int z)
- static void ImageGetRawData (Image \*image, unsigned char \*data)
- IMAGE \* ImageLoad (char \*fileName)

### 7.17.1 Macro Definition Documentation

7.17.1.1 #define IMAGIC 0x01da

7.17.1.2 #define IMAGIC\_SWAP 0xda01

7.17.1.3 #define SWAP\_LONG\_BYTES(x)

### Value:

```
(((((x) & 0xff) << 24) | (((x) & 0xff00) << 8)) | \ ((((x) & 0xff0000) >> 8) | (((x) & 0xff000000) >> 24)))
```

7.17.1.4 #define SWAP\_SHORT\_BYTES(x) ((((x) & 0xff) << 8) | (((x) & 0xff00) >> 8))

#### 7.17.2 Function Documentation

**7.17.2.1** static void ImageClose ( Image \* image ) [static]

7.17.2.2 static void ImageGetRawData ( Image \* image, unsigned char \* data ) [static]

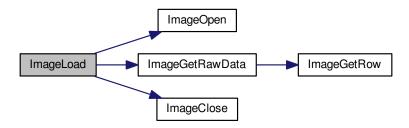
Here is the call graph for this function:



7.17.2.3 static void ImageGetRow ( Image \* image, unsigned char \* buf, int y, int z ) [static]

## 7.17.2.4 IMAGE\* ImageLoad ( char \* fileName )

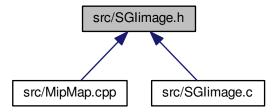
Here is the call graph for this function:



**7.17.2.5 static Image\* ImageOpen ( char \*** *fileName* **)** [static]

# 7.18 src/SGlimage.h File Reference

This graph shows which files directly or indirectly include this file:



## **Classes**

struct IMAGE

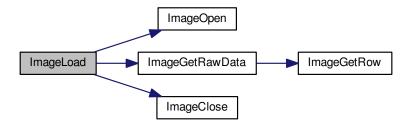
## **Functions**

• IMAGE \* ImageLoad (char \*)

## 7.18.1 Function Documentation

7.18.1.1 IMAGE\* ImageLoad ( char \* )

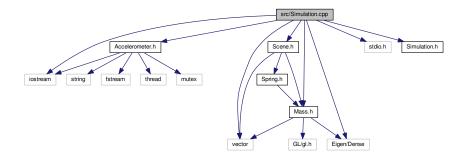
Here is the call graph for this function:



#### 7.19 src/Simulation.cpp File Reference

```
#include <iostream>
#include <vector>
#include <stdio.h>
#include <Eigen/Dense>
#include "Mass.h"
#include "Scene.h"
#include "Accelerometer.h"
#include "Simulation.h"
```

Include dependency graph for Simulation.cpp:



### **Namespaces**

• std

#### **Functions**

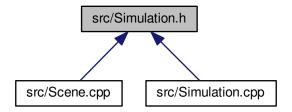
- void std::fwd\_euler (double dt, vector < Mass > &points, vector < Spring > &springs, bool interaction)
- Eigen::Vector3d std::f\_int (Mass &pt, vector < Spring > &springs)
- Eigen::Vector3d std::gravity ()
- void std::symplectic (double dt, vector < Mass > &points, vector < Spring > &springs, bool interaction)
- void std::leapfrog (double dt, vector< Mass > &points, vector< Spring > &springs, bool interaction)
- void std::midpoint (double dt, vector < Mass > &points, vector < Spring > &springs, bool interaction)
- void std::force (vector< Mass > &points, vector< Spring > &springs, bool interaction)

#### **Variables**

- bool std::leapFrogInitialized = false
- double std::t = 0
- double std::floorLevel = -1.
- double std::repulsiveSpringConst = -50.

## 7.20 src/Simulation.h File Reference

This graph shows which files directly or indirectly include this file:



#### Classes

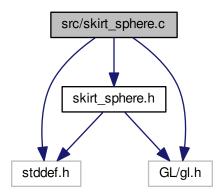
· class std::Simulation

## **Namespaces**

• std

# 7.21 src/skirt\_sphere.c File Reference

```
#include <stddef.h>
#include <GL/gl.h>
#include "skirt_sphere.h"
Include dependency graph for skirt_sphere.c:
```



#### **Variables**

- const size\_t skirt\_sphereVertices = 4008
- GLfloat skirt\_spherePositions [12024]
- GLfloat skirt\_sphereTexels [8016]
- GLfloat skirt\_sphereNormals [12024]
- const size\_t skirt\_sphereObjectsWithMass = 1
- const size\_t skirt\_sphereMasses [1]
- const size t skirt sphereMassFwdOffs [1]
- const size\_t skirt\_sphereMassVertices [1]
- const size\_t skirt\_sphereMassRevOffs [1]
- const size t skirt sphereMassRevOffsOrig [1]
- const size\_t skirt\_sphereFwdIndexI [1824]
- const size\_t \* skirt\_sphereFwdIndex [353]
- const size t skirt sphereFwdIndexLength [353]
- const size\_t skirt\_sphereRevIndex [1824]
- const size\_t skirt\_sphereObjects = 3
- const size\_t skirt\_sphereObjectOffset [3]
- const size\_t skirt\_sphereObjectLength [3]
- const char skirt\_sphereObjectNamesString [59]
- const char \* skirt\_sphereObjectNames [3]
- const char skirt\_sphereTextureFilePath [34] = "/not\_implemented\_yet/edit/by.hand"

### 7.21.1 Detailed Description

This is a C-source file (.c) for the models "skirt\_sphere\_Cone", "skirt\_sphere\_Grid", "skirt\_sphere\_lcosphere" Don't edit! This is an auto-generated file by blender2oGL. Modifications are not permanent.

## 7.21.2 Variable Documentation

```
7.21.2.1 const size_t* skirt_sphereFwdIndex[353]
```

7.21.2.2 const size\_t skirt\_sphereFwdIndexI[1824]

7.21.2.3 const size\_t skirt\_sphereFwdIndexLength[353]

7.21.2.4 const size\_t skirt\_sphereMasses[1]

#### Initial value:

```
= {
353,
```

#### 7.21.2.5 const size\_t skirt\_sphereMassFwdOffs[1]

#### Initial value:

7.21.2.6 const size\_t skirt\_sphereMassRevOffs[1]

Initial value:

```
= {
     0,
}
```

7.21.2.7 const size\_t skirt\_sphereMassRevOffsOrig[1]

Initial value:

```
= { 0,
```

7.21.2.8 const size\_t skirt\_sphereMassVertices[1]

Initial value:

```
= {
 1824,
```

7.21.2.9 GLfloat skirt\_sphereNormals[12024]

7.21.2.10 const size\_t skirt\_sphereObjectLength[3]

Initial value:

7.21.2.11 const char\* skirt\_sphereObjectNames[3]

Initial value:

```
= {
    &skirt_sphereObjectNamesString[0],
    &skirt_sphereObjectNamesString[18],
    &skirt_sphereObjectNamesString[36],
}
```

7.21.2.12 const char skirt\_sphereObjectNamesString[59]

Initial value:

```
= {
    "skirt_sphere_Cone\0"
    "skirt_sphere_Grid\0"
    "skirt_sphere_Icosphere\0"
}
```

7.21.2.13 const size\_t skirt\_sphereObjectOffset[3]

#### Initial value:

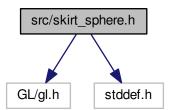
```
0,
1824,
3768,
```

```
7.21.2.14 const size_t skirt_sphereObjects = 3
```

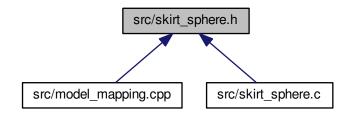
- 7.21.2.15 const size\_t skirt\_sphereObjectsWithMass = 1
- 7.21.2.16 GLfloat skirt\_spherePositions[12024]
- 7.21.2.17 const size\_t skirt\_sphereRevIndex[1824]
- 7.21.2.18 GLfloat skirt\_sphereTexels[8016]
- 7.21.2.19 const char skirt\_sphereTextureFilePath[34] = "/not\_implemented\_yet/edit/by.hand"
- 7.21.2.20 const size\_t skirt\_sphereVertices = 4008

# 7.22 src/skirt\_sphere.h File Reference

```
#include <GL/gl.h>
#include <stddef.h>
Include dependency graph for skirt_sphere.h:
```



This graph shows which files directly or indirectly include this file:



#### **Variables**

- const size\_t skirt\_sphereVertices
- GLfloat skirt\_spherePositions [12024]
- GLfloat skirt sphereTexels [8016]
- GLfloat skirt sphereNormals [12024]
- · const size\_t skirt\_sphereObjectsWithMass
- const size\_t skirt\_sphereMasses [1]
- const size\_t skirt\_sphereMassFwdOffs [1]
- const size\_t skirt\_sphereMassVertices [1]
- const size\_t skirt\_sphereMassRevOffs [1]
- const size\_t skirt\_sphereMassRevOffsOrig [1]
- const size\_t skirt\_sphereFwdIndexI [1824]
- const size\_t \* skirt\_sphereFwdIndex [353]
- const size\_t skirt\_sphereFwdIndexLength [353]
- const size\_t skirt\_sphereRevIndex [1824]
- · const size\_t skirt\_sphereObjects
- const size\_t skirt\_sphereObjectOffset [3]
- const size\_t skirt\_sphereObjectLength [3]
- const char skirt\_sphereObjectNamesString [59]
- const char \* skirt\_sphereObjectNames [3]
- const char skirt\_sphereTextureFilePath [34]

### 7.22.1 Detailed Description

This is a C-header file (.h) for the models "skirt\_sphere\_Cone", "skirt\_sphere\_Grid", "skirt\_sphere\_lcosphere" Don't edit! This is an auto-generated file by blender2oGL. Modifications are not permanent.

## 7.22.2 Variable Documentation

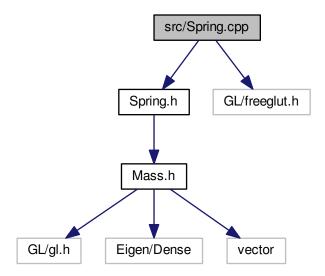
- 7.22.2.1 const size\_t\* skirt\_sphereFwdIndex[353]
- 7.22.2.2 const size\_t skirt\_sphereFwdIndexI[1824]
- 7.22.2.3 const size\_t skirt\_sphereFwdIndexLength[353]

```
7.22.2.4 const size_t skirt_sphereMasses[1]
7.22.2.5 const size_t skirt_sphereMassFwdOffs[1]
7.22.2.6 const size_t skirt_sphereMassRevOffs[1]
7.22.2.7 const size_t skirt_sphereMassRevOffsOrig[1]
7.22.2.8 const size_t skirt_sphereMassVertices[1]
7.22.2.9 GLfloat skirt_sphereNormals[12024]
7.22.2.10 const size_t skirt_sphereObjectLength[3]
7.22.2.11 const char* skirt_sphereObjectNames[3]
7.22.2.12 const char skirt_sphereObjectNamesString[59]
7.22.2.13 const size_t skirt_sphereObjectOffset[3]
7.22.2.14 const size_t skirt_sphereObjects
7.22.2.15 const size_t skirt_sphereObjectsWithMass
7.22.2.16 GLfloat skirt_spherePositions[12024]
7.22.2.17 const size_t skirt_sphereRevIndex[1824]
7.22.2.18 GLfloat skirt_sphereTexels[8016]
7.22.2.19 const char skirt_sphereTextureFilePath[34]
7.22.2.20 const size_t skirt_sphereVertices
```

## 7.23 src/Spring.cpp File Reference

```
#include "Spring.h"
#include <GL/freeglut.h>
```

Include dependency graph for Spring.cpp:



## **Macros**

- #define \_DEBUG\_MSG\_C(X)
- #define \_DEBUG\_MSG(X)

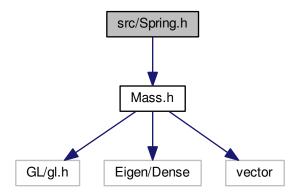
## 7.23.1 Macro Definition Documentation

- 7.23.1.1 #define \_DEBUG\_MSG( X )
- 7.23.1.2 #define \_DEBUG\_MSG\_C( X )

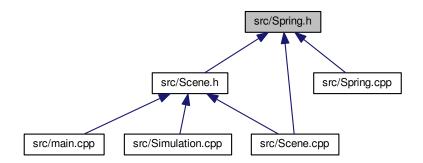
# 7.24 src/Spring.h File Reference

#include "Mass.h"

Include dependency graph for Spring.h:



This graph shows which files directly or indirectly include this file:



## Classes

• class Spring