### SWE

Generated by Doxygen 1.8.4

Mon May 27 2013 05:17:31

## **Contents**

1	SWI	E - A Sir	nple Shallow Water Code	1				
	1.1	The SI	nallow Water Equations	1				
		1.1.1	Finite Volume Discretisation	2				
		1.1.2	Computing Numerical Fluxes at Edges and Euler Time-Stepping	2				
	1.2	Impler	nentation and base class SWE_Block	2				
		1.2.1	Parallelisation and Different Models	3				
		1.2.2	Multiple Blocks	3				
		1.2.3		4				
		1.2.4		4				
		1.2.5		4				
2	Tod	o List		5				
3	Dep	recated	List	7				
4	Hier	archica	I Index	9				
	4.1	Class	Hierarchy	9				
5	Clas	ss Index		11				
	5.1	Class	List	11				
6	File	Index		13				
	6.1	File Lis	st	13				
7	Clas	Class Documentation 1						
	7.1	Came	a Class Reference	15				
		7.1.1	Constructor & Destructor Documentation	15				
			7.1.1.1 Camera	15				
		7.1.2	Member Function Documentation	15				
			7.1.2.1 displayImage	15				
			7.1.2.2 orient	15				
			7.1.2.3 panning	16				
			7.1.2.4 setCamera	16				
			7.1.2.5 startPanning	16				

iv CONTENTS

		7.1.2.6	viewDistance	. 16
		7.1.2.7	zoomln	. 16
		7.1.2.8	zoomOut	. 16
7.2	Contro	ller Class F	Reference	. 16
	7.2.1	Construct	for & Destructor Documentation	. 17
		7.2.1.1	Controller	. 17
	7.2.2	Member F	Function Documentation	. 18
		7.2.2.1	handleEvents	. 18
		7.2.2.2	hasFocus	. 18
		7.2.2.3	isPaused	. 18
7.3	Dimen	sionalSplitti	ingTest Class Reference	. 18
7.4	Float1	D Class Re	ference	. 19
	7.4.1	Detailed D	Description	. 19
7.5	Float2l	D Class Re	ference	. 19
	7.5.1	Detailed D	Description	. 19
	7.5.2	Construct	or & Destructor Documentation	. 19
		7.5.2.1	Float2D	. 19
7.6	FWave	SolverTest	Class Reference	. 20
	7.6.1	Detailed D	Description	. 20
	7.6.2	Member F	Function Documentation	. 20
		7.6.2.1	testBathymetry	. 20
		7.6.2.2	testEigencoefficients	. 20
		7.6.2.3	testEigenvalues	. 20
		7.6.2.4	testFlux	. 21
		7.6.2.5	testSteadyState	. 21
		7.6.2.6	testSupersonic	. 21
7.7	io::Bou	ındarySize	Struct Reference	. 21
	7.7.1	Detailed D	Description	. 21
	7.7.2	Member E	Data Documentation	. 21
		7.7.2.1	boundarySize	. 21
7.8	io::Net	CdfWriter C	Class Reference	. 22
	7.8.1	Construct	for & Destructor Documentation	. 22
		7.8.1.1	NetCdfWriter	. 22
		7.8.1.2	$\sim\! NetCdfWriter \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots$	. 22
		7.8.1.3	NetCdfWriter	. 23
	7.8.2	Member F	Function Documentation	. 24
		7.8.2.1	writeTimeStep	. 24
		7.8.2.2	writeTimeStep	. 24
7.9	io::Vtk	Writer Class	s Reference	. 24
	7.9.1	Construct	or & Destructor Documentation	. 25

CONTENTS

		7.9.1.1	VtkWriter	25
	7.9.2	Member F	Function Documentation	25
		7.9.2.1	writeTimeStep	25
7.10	io::Writ	er Class Re	eference	26
	7.10.1	Constructo	or & Destructor Documentation	26
		7.10.1.1	Writer	26
	7.10.2	Member F	function Documentation	26
		7.10.2.1	writeTimeStep	26
7.11	scenari	ios::Consta	ntFlow Class Reference	27
	7.11.1	Member F	function Documentation	27
		7.11.1.1	getBathymetry	27
		7.11.1.2	getCellSize	27
		7.11.1.3	getHeight	27
		7.11.1.4	getMomentum	27
7.12	scenari	ios::DamBre	eak Class Reference	28
	7.12.1	Member F	function Documentation	28
		7.12.1.1	getBathymetry	28
		7.12.1.2	getCellSize	28
		7.12.1.3	getCellSize	28
		7.12.1.4	getHeight	28
		7.12.1.5	getHeight	28
		7.12.1.6	getMomentum	28
		7.12.1.7	getMomentum	29
7.13	scenari	ios::Eisbach	h Class Reference	29
	7.13.1	Member F	function Documentation	29
		7.13.1.1	getBathymetry	29
		7.13.1.2	getCellSize	29
		7.13.1.3	getHeight	29
		7.13.1.4	getMomentum	29
7.14	scenari	ios::RareRa	are Class Reference	30
	7.14.1	Member F	function Documentation	30
		7.14.1.1	getCellSize	30
		7.14.1.2	getHeight	30
		7.14.1.3	getMomentum	30
7.15	scenari	ios::ShockS	Shock Class Reference	30
	7.15.1	Member F	function Documentation	30
		7.15.1.1	getCellSize	30
		7.15.1.2	getHeight	31
		7.15.1.3	getMomentum	31
7.16	scenari	ios::Subcrit	Class Reference	31

vi CONTENTS

	7.16.1	Member	Function Documentation	31
		7.16.1.1	getBathymetry	31
		7.16.1.2	getCellSize	31
		7.16.1.3	getHeight	31
		7.16.1.4	getMomentum	31
7.17	scenari	os::Super	crit Class Reference	32
	7.17.1	Member	Function Documentation	32
		7.17.1.1	getBathymetry	32
		7.17.1.2	getCellSize	32
		7.17.1.3	getHeight	32
		7.17.1.4	getMomentum	32
7.18	Shader	Class Re	ference	32
	7.18.1	Construc	tor & Destructor Documentation	33
		7.18.1.1	Shader	33
		7.18.1.2	~Shader	34
	7.18.2	Member	Function Documentation	34
		7.18.2.1	disableShader	34
		7.18.2.2	enableShader	34
		7.18.2.3	getUniformLocation	34
		7.18.2.4	setUniform	34
		7.18.2.5	shadersLoaded	34
7.19	Simula	tion Class	Reference	34
	7.19.1	Construc	tor & Destructor Documentation	35
		7.19.1.1	Simulation	35
		7.19.1.2	$\sim$ Simulation	35
	7.19.2	Member	Function Documentation	35
		7.19.2.1	getScalingApproximation	35
		7.19.2.2	restart	35
		7.19.2.3	runCuda	35
		7.19.2.4	setBathBuffer	35
7.20	solver::	FWave<	T > Class Template Reference	35
	7.20.1	Detailed	Description	36
	7.20.2	Member	Function Documentation	36
		7.20.2.1	computeNetUpdates	36
7.21	SWE_A	ArtificialTs	unamiScenario Class Reference	36
7.22	SWE_A	AsagiGrid	Class Reference	37
7.23	SWE_A	AsagiJapa	nSmallVisInfo Class Reference	37
	7.23.1	Member	Function Documentation	37
		7.23.1.1	bathyVerticalScaling	37
		7.23.1.2	waterVerticalScaling	38

CONTENTS vii

7.24	SWE_A	AsagiScen	ario Class Reference	38
	7.24.1	Construct	tor & Destructor Documentation	38
		7.24.1.1	SWE_AsagiScenario	38
	7.24.2	Member F	Function Documentation	39
		7.24.2.1	dynamicDisplacementAvailable	39
		7.24.2.2	endSimulation	39
		7.24.2.3	getBathymetry	39
		7.24.2.4	getBathymetryAndDynamicDisplacement	39
		7.24.2.5	getBoundaryPos	40
		7.24.2.6	getBoundaryType	40
		7.24.2.7	getWaterHeight	40
7.25	SWE_E	Bathymetry	/DamBreakScenario Class Reference	41
	7.25.1	Detailed I	Description	41
	7.25.2	Member I	Function Documentation	41
		7.25.2.1	getBoundaryPos	41
		7.25.2.2	getWaterHeight	41
7.26	SWE_E	Bathymetry	DamBreakVisInfo Class Reference	42
	7.26.1	Detailed I	Description	42
	7.26.2	Member I	Function Documentation	42
		7.26.2.1	bathyVerticalOffset	42
7.27	SWE_E	Block Class	s Reference	42
	7.27.1	Detailed I	Description	44
	7.27.2	Construct	tor & Destructor Documentation	45
		7.27.2.1	SWE_Block	46
		7.27.2.2	$\sim$ SWE_Block	46
	7.27.3	Member I	Function Documentation	46
		7.27.3.1	computeMaxTimestep	46
		7.27.3.2	computeNumericalFluxes	46
		7.27.3.3	getBathymetry	46
		7.27.3.4	getDischarge_hu	46
		7.27.3.5	getDischarge_hv	46
		7.27.3.6	getMaxTimestep	46
		7.27.3.7	getWaterHeight	47
		7.27.3.8	grabGhostLayer	47
		7.27.3.9	initScenario	47
		7.27.3.10	registerCopyLayer	47
		7.27.3.11	setBathymetry	48
		7.27.3.12	setBathymetry	48
		7.27.3.13	setBoundaryBathymetry	48
		7.27.3.14	setBoundaryConditions	48

viii CONTENTS

	7.27.3.15 setBoundaryType	48
	7.27.3.16 setDischarge	48
	7.27.3.17 setGhostLayer	48
	7.27.3.18 setWaterHeight	49
	7.27.3.19 simulate	49
	7.27.3.20 synchAfterWrite	49
	7.27.3.21 synchBathymetryAfterWrite	49
	7.27.3.22 synchBathymetryBeforeRead	49
	7.27.3.23 synchBeforeRead	49
	7.27.3.24 synchCopyLayerBeforeRead	49
	7.27.3.25 synchDischargeAfterWrite	50
	7.27.3.26 synchDischargeBeforeRead	50
	7.27.3.27 synchGhostLayerAfterWrite	50
	7.27.3.28 synchWaterHeightAfterWrite	50
	7.27.3.29 synchWaterHeightBeforeRead	50
	7.27.3.30 updateUnknowns	50
7.27.4	Member Data Documentation	50
	7.27.4.1 maxTimestep	50
7.28 SWE_E	Block1D Struct Reference	51
7.28.1	Detailed Description	51
7.29 SWE_E	BlockCUDA Class Reference	51
7.29.1	Detailed Description	52
7.29.2	Constructor & Destructor Documentation	52
	7.29.2.1 SWE_BlockCUDA	52
	7.29.2.2 ~SWE_BlockCUDA	53
7.29.3	Member Function Documentation	53
	7.29.3.1 finalize	53
	7.29.3.2 getCUDA_bathymetry	53
	7.29.3.3 getCUDA_waterHeight	53
	7.29.3.4 grabGhostLayer	53
	7.29.3.5 init	53
	7.29.3.6 printDeviceInformation	53
	7.29.3.7 registerCopyLayer	54
	7.29.3.8 setBoundaryConditions	54
	7.29.3.9 synchAfterWrite	54
	7.29.3.10 synchBathymetryAfterWrite	54
	7.29.3.11 synchBathymetryBeforeRead	54
	7.29.3.12 synchBeforeRead	54
	7.29.3.13 synchCopyLayerBeforeRead	54
	7.29.3.14 synchDischargeAfterWrite	55

CONTENTS

	7.29.3.15 synchDischargeBeforeRead	55
	7.29.3.16 synchGhostLayerAfterWrite	55
	7.29.3.17 synchWaterHeightAfterWrite	55
	7.29.3.18 synchWaterHeightBeforeRead	55
SWE_I	DamBreakScenario Class Reference	55
7.30.1	Member Function Documentation	56
	7.30.1.1 getBoundaryPos	56
SWE_I	DimensionalSplitting Class Reference	56
7.31.1	Member Function Documentation	56
	7.31.1.1 computeNumericalFluxes	56
	7.31.1.2 runTimestep	57
	7.31.1.3 simulate	57
	7.31.1.4 updateUnknowns	57
SWE_I	NetCDFCheckpointScenario Class Reference	57
7.32.1	Constructor & Destructor Documentation	57
	7.32.1.1 SWE_NetCDFCheckpointScenario	57
7.32.2	Member Function Documentation	58
	7.32.2.1 readNetCDF	58
SWE_I	NetCDFScenario Class Reference	58
SWE_F	RadialDamBreakScenario Class Reference	59
7.34.1	Detailed Description	59
7.34.2	Member Function Documentation	59
	7.34.2.1 getBoundaryPos	59
SWE_F	RusanovBlock Class Reference	59
7.35.1	Detailed Description	60
7.35.2	Constructor & Destructor Documentation	61
	7.35.2.1 SWE_RusanovBlock	61
	7.35.2.2 ~SWE_RusanovBlock	61
7.35.3	Member Function Documentation	61
	7.35.3.1 computeBathymetrySources	61
	7.35.3.2 computeFlux	61
	7.35.3.3 computeLocalSV	61
	7.35.3.4 computeNumericalFluxes	61
	7.35.3.5 simulate	61
	7.35.3.6 simulateTimestep	62
	7.35.3.7 updateUnknowns	62
7.35.4	Friends And Related Function Documentation	62
	7.35.4.1 operator<<	62
SWE_F	RusanovBlockCUDA Class Reference	62
7.36.1	Detailed Description	63
	7.30.1  SWE_I 7.31.1  SWE_I 7.32.1  7.32.2  SWE_I 7.34.1  7.34.2  SWE_I 7.35.1  7.35.3	7.29.3.16 synchGhostLayerAfterWrite 7.29.3.17 synchWaterHeightAfterWrite 7.29.3.18 synchWaterHeightBeforeRead  SWE_DamBreakScenario Class Reference 7.30.1 Member Function Documentation 7.30.1.1 getBoundaryPos  SWE_DimensionalSplitting Class Reference 7.31.1 Member Function Documentation 7.31.1.1 computeNumericalFluxes 7.31.1.2 runTimestep 7.31.1.3 simulate 7.31.1.4 updateUnknowns  SWE_NetCDFCheckpointScenario Class Reference 7.32.1 Constructor & Destructor Documentation 7.32.1.1 SWE_NetCDFCheckpointScenario 7.32.2 Member Function Documentation 7.32.2.1 meatNetCDF  SWE_NetCDFScenario Class Reference  SWE_RadialDamBreakScenario Class Reference 7.34.1 Detailed Description 7.34.2 Member Function Documentation 7.35.3.1 Detailed Description 7.35.3.2 computeNation 7.35.3.1 original Description 7.35.3.2 computeDathymetrySources 7.35.3.2 computeDathymetrySources 7.35.3.3 computeLocalSV 7.35.3.4 computeDathymetrySources 7.35.3.5 simulate 7.35.3.5 simulate 7.35.3.7 updateUnknowns 7.35.3.1 pudateUnknowns 7.35.3.1 pudateUnknowns 7.35.3.1 pudateUnknowns 7.35.3.1 Friends And Related Function Documentation

CONTENTS

	7.36.2	Constructor & Destructor Documentation	63
		7.36.2.1 SWE_RusanovBlockCUDA	63
		7.36.2.2 ~SWE_RusanovBlockCUDA	63
	7.36.3	Member Function Documentation	63
		7.36.3.1 computeNumericalFluxes	63
		7.36.3.2 simulate	63
		7.36.3.3 simulateTimestep	64
		7.36.3.4 updateUnknowns	64
	7.36.4	Friends And Related Function Documentation	64
		7.36.4.1 operator<<	64
7.37	SWE_S	Scenario Class Reference	64
	7.37.1	Detailed Description	65
7.38	SWE_S	SeaAtRestScenario Class Reference	65
	7.38.1	Detailed Description	66
7.39	SWE_S	SplashingConeScenario Class Reference	66
	7.39.1	Detailed Description	66
7.40	SWE_S	SplashingPoolScenario Class Reference	67
	7.40.1	Detailed Description	67
	7.40.2	Member Function Documentation	67
		7.40.2.1 getBoundaryPos	67
7.41	SWE_	TestingScenario Class Reference	67
	7.41.1	Member Function Documentation	68
		7.41.1.1 getBoundaryPos	68
7.42	SWE_	TsunamiScenario Class Reference	68
	7.42.1	Constructor & Destructor Documentation	69
		7.42.1.1 SWE_TsunamiScenario	69
	7.42.2	Member Function Documentation	70
		7.42.2.1 readNetCDF	70
7.43	SWE_	SunamiScenarioTest Class Reference	70
7.44	SWE_\	VisInfo Class Reference	71
	7.44.1	Detailed Description	71
	7.44.2	Constructor & Destructor Documentation	71
		7.44.2.1 ~SWE_VisInfo	71
	7.44.3	Member Function Documentation	71
		7.44.3.1 bathyVerticalOffset	71
		7.44.3.2 bathyVerticalScaling	71
		7.44.3.3 waterVerticalScaling	72
7.45	SWE_\	NavePropagationBlock Class Reference	72
	7.45.1	Detailed Description	72
	7.45.2	Constructor & Destructor Documentation	72

CONTENTS xi

		7.45.2.1	SWE_WavePropagationBlock	72
		7.45.2.2	$\sim\!\!SWE\_WavePropagationBlock\ .\ .\ .\ .\ .\ .\ .\ .\ .\ .\ .\ .\ .\$	73
	7.45.3	Member	Function Documentation	73
		7.45.3.1	computeNumericalFluxes	73
		7.45.3.2	simulate	73
		7.45.3.3	simulateTimestep	74
		7.45.3.4	updateUnknowns	74
7.46	SWE_V	<b>WavePropa</b>	agationBlockCuda Class Reference	74
	7.46.1	Detailed	Description	75
	7.46.2	Construc	tor & Destructor Documentation	75
		7.46.2.1	SWE_WavePropagationBlockCuda	75
		7.46.2.2	~SWE_WavePropagationBlockCuda	77
	7.46.3	Member	Function Documentation	77
		7.46.3.1	computeNumericalFluxes	77
		7.46.3.2	simulate	78
		7.46.3.3	simulateTimestep	78
		7.46.3.4	updateUnknowns	78
7.47	Text Cla	ass Refere	ence	78
	7.47.1	Member	Function Documentation	79
		7.47.1.1	showNextText	79
7.48	tools::A	Args Class	Reference	79
	7.48.1	Detailed	Description	79
7.49	tools::L	ogger Cla	ss Reference	79
	7.49.1	Construc	tor & Destructor Documentation	80
		7.49.1.1	Logger	81
		7.49.1.2	~Logger	82
	7.49.2	Member	Function Documentation	82
		7.49.2.1	cout	82
		7.49.2.2	initWallClockTime	82
		7.49.2.3	operator<<	82
		7.49.2.4	operator<<	82
		7.49.2.5	printCellSize	82
		7.49.2.6	printCpuCommunicationTime	83
		7.49.2.7	printCpuTime	83
		7.49.2.8	printFinishMessage	83
		7.49.2.9	printIterationsDone	83
			printNumberOfBlocks	83
			printNumberOfCells	83
			? printNumberOfCellsPerProcess	84
		7.49.2.13	B printNumberOfProcesses	84

xii CONTENTS

		7.49.2.14	4 printOutputFileCreation	84
		7.49.2.15	5 printOutputTime	84
		7.49.2.16	5 printSimulationTime	84
		7.49.2.17	7 printSolverStatistics	85
		7.49.2.18	3 printStartMessage	85
		7.49.2.19	9 printStatisticsMessage	85
		7.49.2.20	D printString	85
		7.49.2.21	1 printWallClockTime	85
		7.49.2.22	2 printWelcomeMessage	86
		7.49.2.23	3 setProcessRank	86
		7.49.2.24	4 updateCpuCommunicationTime	86
		7.49.2.25	5 updateCpuTime	86
	7.49.3	Member I	Data Documentation	86
		7.49.3.1	logger	86
7.50	tools::P	rogressBa	ar Class Reference	86
	7.50.1	Member I	Function Documentation	86
		7.50.1.1	update	86
7.51	VBO C	lass Refer	rence	87
	7.51.1	Member I	Function Documentation	87
		7.51.1.1	finialize	87
		7.51.1.2	getName	87
			init	
7.52	Visualiz	zation Clas	ss Reference	87
	7.52.1	Construct	ctor & Destructor Documentation	88
		7.52.1.1	Visualization	88
		7.52.1.2	$\sim$ Visualization	88
	7.52.2	Member I	Function Documentation	88
		7.52.2.1	cleanUp	88
		7.52.2.2	getCudaNormalsPtr	88
		7.52.2.3	getCudaWaterSurfacePtr	88
		7.52.2.4	init	88
		7.52.2.5	isExtensionSupported	89
		7.52.2.6	renderDisplay	89
		7.52.2.7	resizeWindow	89
		7.52.2.8	setRenderingMode	89
		7.52.2.9	toggleRenderingMode	89
7.53	WaveP	ropagation	n Class Reference	89
			Description	
	7.53.2		ctor & Destructor Documentation	
		7.53.2.1	WavePropagation	90

CONTENTS xiii

		7.53.3	Member Function Documentation	90
			7.53.3.1 computeNumericalFluxes	90
			7.53.3.2 setOutflowBoundaryConditions	91
			7.53.3.3 updateUnknowns	91
	7.54	writer::	ConsoleWriter Class Reference	91
		7.54.1	Detailed Description	91
		7.54.2	Member Function Documentation	91
			7.54.2.1 write	91
	7.55	writer::	VtkWriter Class Reference	91
		7.55.1	Detailed Description	92
		7.55.2	Member Function Documentation	92
			7.55.2.1 write	92
8	File I	Docume	entation	93
•	8.1			93
		8.1.1		93
		8.1.2		93
		8.1.3		93
	8.2	/home/	raphael/Programmieren/BPraktikum/SWE/src/blocks/cuda/SWE_BlockCUDA.cu File Refer-	93
		8.2.1		94
		8.2.2	•	94
		8.2.3		94
	8.3		raphael/Programmieren/BPraktikum/SWE/src/blocks/cuda/SWE_BlockCUDA.hh_File_Refer-	
			_	94
		8.3.1	Detailed Description	95
		8.3.2	LICENSE	95
		8.3.3	DESCRIPTION	95
		8.3.4	Function Documentation	95
			8.3.4.1 getBathyCoord	95
			8.3.4.2 getCellCoord	95
			8.3.4.3 getEdgeCoord	95
	8.4		raphael/Programmieren/BPraktikum/SWE/src/blocks/cuda/SWE_BlockCUDA_kernels.cu	96
		8.4.1	Detailed Description	96
		8.4.2	LICENSE	96
		8.4.3	DESCRIPTION	96
		8.4.4	Function Documentation	97
			8.4.4.1 kernelBottomBoundary	97
			8.4.4.2 kernelBottomCopyLayer	97
			8.4.4.3 kernelBottomGhostBoundary	97

XIV

		8.4.4.4	kernelHdBufferEdges	97
		8.4.4.5	kernelLeftBoundary	97
		8.4.4.6	kernelRightBoundary	97
		8.4.4.7	kernelTopBoundary	97
		8.4.4.8	kernelTopCopyLayer	97
		8.4.4.9	kernelTopGhostBoundary	98
8.5		•	rogrammieren/BPraktikum/SWE/src/blocks/cuda/SWE_BlockCUDA_kernels.hh	98
	8.5.1	Detailed	Description	98
	8.5.2	LICENSE		98
	8.5.3	DESCRI	PTION	98
	8.5.4	Function	Documentation	99
		8.5.4.1	kernelBottomBoundary	99
		8.5.4.2	kernelBottomCopyLayer	99
		8.5.4.3	kernelBottomGhostBoundary	99
		8.5.4.4	kernelHdBufferEdges	99
		8.5.4.5	kernelLeftBoundary	99
		8.5.4.6	kernelMaximum	99
		8.5.4.7	kernelRightBoundary	99
		8.5.4.8	kernelTopBoundary	99
		8.5.4.9	kernelTopCopyLayer	99
		8.5.4.10		100
8.6	/home/	raphael/Pi	rogrammieren/BPraktikum/SWE/src/blocks/cuda/SWE_WavePropagationBlock-	
	Cuda.c	u File Ref	erence	100
	8.6.1	Detailed	Description	100
	8.6.2	LICENSE	≣	100
	8.6.3	DESCRI	PTION	100
8.7			rogrammieren/BPraktikum/SWE/src/blocks/cuda/SWE_WavePropagationBlock-	404
			erence	101
	8.7.1		·	101
	8.7.2			101
	8.7.3			101
8.8		-	rogrammieren/BPraktikum/SWE/src/blocks/cuda/SWE_WavePropagationBlock-File Reference	101
	8.8.1	Detailed	Description	102
	8.8.2	LICENSE	≣	102
	8.8.3	DESCRI	PTION	102
	8.8.4	Function	Documentation	102
		8.8.4.1	computeNetUpdatesKernel	102
		8.8.4.2	computeOneDPositionKernel	104
		8.8.4.3	updateUnknownsKernel	104

CONTENTS xv

8.9 /home/raphael/Programmieren/BPraktikum/SWE/src/blocks/cuda/SWE_WavePropagation Cuda_kernels.hh File Reference			
	8.9.1	Detailed Description	105
	8.9.2	LICENSE	106
	8.9.3	DESCRIPTION	106
	8.9.4	Function Documentation	106
		8.9.4.1 computeNetUpdatesKernel	106
		8.9.4.2 computeOneDPositionKernel	108
		8.9.4.3 updateUnknownsKernel	108
8.10		raphael/Programmieren/BPraktikum/SWE/src/blocks/rusanov/SWE_RusanovBlock.cpp File	109
	8.10.1	Detailed Description	109
	8.10.2	LICENSE	109
		DESCRIPTION	110
		Function Documentation	
		8.10.4.1 operator <<	110
8.11		raphael/Programmieren/BPraktikum/SWE/src/blocks/rusanov/SWE_RusanovBlock.hh File	
			110
		Detailed Description	
		LICENSE	
			110
	8.11.4		111
		•	111
8.12		raphael/Programmieren/BPraktikum/SWE/src/blocks/rusanov/SWE_RusanovBlockCUDA.cu ference	111
	8.12.1	Detailed Description	111
	8.12.2	LICENSE	111
	8.12.3	DESCRIPTION	111
	8.12.4	Function Documentation	111
		8.12.4.1 operator<<	111
8.13		raphael/Programmieren/BPraktikum/SWE/src/blocks/rusanov/SWE_RusanovBlockCUDA.hh	112
	8.13.1	Detailed Description	112
	8.13.2	LICENSE	112
	8.13.3	DESCRIPTION	112
	8.13.4	Function Documentation	112
		8.13.4.1 operator <<	112
8.14		raphael/Programmieren/BPraktikum/SWE/src/blocks/rusanov/SWE_RusanovBlockCUDA	
		cu File Reference	113
		•	113
			113
	8.14.3	DESCRIPTION	113

xvi CONTENTS

	8.14.4	Function Documentation	13
		8.14.4.1 kernelComputeBathymetrySources	13
		8.14.4.2 kernelComputeFluxesF	14
		8.14.4.3 kernelComputeFluxesG	14
		8.14.4.4 kernelEulerTimestep	14
		8.14.4.5 kernelMaximum	14
8.15		raphael/Programmieren/BPraktikum/SWE/src/blocks/rusanov/SWE_RusanovBlockCUDAhh File Reference	14
	8.15.1	Detailed Description	14
	8.15.2	LICENSE	14
	8.15.3	DESCRIPTION	15
	8.15.4	Function Documentation	15
		8.15.4.1 kernelComputeBathymetrySources	15
		8.15.4.2 kernelComputeFluxesF	15
		8.15.4.3 kernelComputeFluxesG	15
		8.15.4.4 kernelEulerTimestep	15
		8.15.4.5 kernelMaximum	15
8.16	/home/i	raphael/Programmieren/BPraktikum/SWE/src/blocks/SWE_Block.cpp File Reference 1	15
	8.16.1	Detailed Description	15
	8.16.2	LICENSE	16
	8.16.3	DESCRIPTION	16
8.17	/home/i	raphael/Programmieren/BPraktikum/SWE/src/blocks/SWE_Block.hh File Reference 1	16
	8.17.1	Detailed Description	16
	8.17.2	LICENSE	17
	8.17.3	DESCRIPTION	17
8.18		raphael/Programmieren/BPraktikum/SWE/src/blocks/SWE_WavePropagationBlock.cpp File nce	
			17
		·	17
			17
8.19	/home/i	raphael/Programmieren/BPraktikum/SWE/src/blocks/SWE_WavePropagationBlock.hh File	18
	8.19.1	Detailed Description	18
	8.19.2	LICENSE	18
	8.19.3	DESCRIPTION	18
8.20		raphael/Programmieren/BPraktikum/SWE/src/examples/swe_DimensionalSplitting.cpp File	19
			19
			19
			19
			19
	0.20.4	runction Documentation	18

CONTENTS xvii

		8.20.4.1 main	119
8.21	/home/i	raphael/Programmieren/BPraktikum/SWE/src/examples/swe_mpi.cpp File Reference	120
	8.21.1	Detailed Description	120
	8.21.2	LICENSE	121
	8.21.3	DESCRIPTION	121
	8.21.4	Function Documentation	121
		8.21.4.1 computeNumberOfBlockRows	121
		8.21.4.2 exchangeBottomTopGhostLayers	121
		8.21.4.3 exchangeLeftRightGhostLayers	122
		8.21.4.4 main	122
8.22	/home/i	raphael/Programmieren/BPraktikum/SWE/src/examples/swe_simple.cpp File Reference	123
	8.22.1	Detailed Description	123
	8.22.2	LICENSE	123
	8.22.3	DESCRIPTION	123
	8.22.4	Function Documentation	124
		8.22.4.1 main	124
8.23	/home/i	raphael/Programmieren/BPraktikum/SWE/src/opengl/vbo.cpp File Reference	124
	8.23.1	Detailed Description	124
	8.23.2	LICENSE	124
8.24	/home/ı	raphael/Programmieren/BPraktikum/SWE/src/opengl/vbo.h File Reference	124
	8.24.1	Detailed Description	125
	8.24.2	LICENSE	125
	8.24.3	DESCRIPTION	125
8.25		raphael/Programmieren/BPraktikum/SWE/src/scenarios/SWE_ArtificialTsunamiScenario.hh	125
		Detailed Description	
		LICENSE	125
		DESCRIPTION	
0.00			126
8.26		raphael/Programmieren/BPraktikum/SWE/src/scenarios/SWE_AsagiScenario.cpp File Ref-	126
	8.26.1	Detailed Description	126
	8.26.2	LICENSE	126
8.27	/home/i	raphael/Programmieren/BPraktikum/SWE/src/scenarios/SWE_AsagiScenario.hh File Refer-	
			126
	8.27.1	Detailed Description	127
	8.27.2	LICENSE	127
		DESCRIPTION	127
8.28		raphael/Programmieren/BPraktikum/SWE/src/scenarios/SWE_AsagiScenario_vis.hh File	127
		Detailed Description	
	0.40.4	LICENSE	120

xviii CONTENTS

	8.28.3	DESCRIPTION	128
8.29		raphael/Programmieren/BPraktikum/SWE/src/scenarios/SWE_NetCDFCheckpointScenario.ht	
	8.29.1	Detailed Description	128
	8.29.2	LICENSE	128
	8.29.3	DESCRIPTION	129
8.30	/home/	raphael/Programmieren/BPraktikum/SWE/src/scenarios/SWE_Scenario.hh File Reference .	129
	8.30.1	Detailed Description	129
	8.30.2	LICENSE	129
	8.30.3	DESCRIPTION	129
	8.30.4	Typedef Documentation	130
		8.30.4.1 BoundaryEdge	130
		8.30.4.2 BoundaryType	130
	8.30.5	Enumeration Type Documentation	130
		8.30.5.1 BoundaryEdge	130
		8.30.5.2 BoundaryType	130
8.31		raphael/Programmieren/BPraktikum/SWE/src/scenarios/SWE_simple_scenarios.hh File Ref-	
			130
		Detailed Description	
		LICENSE	
		DESCRIPTION	131
8.32		raphael/Programmieren/BPraktikum/SWE/src/scenarios/SWE_TsunamiScenario.hh File nce	131
	8.32.1	Detailed Description	131
	8.32.2	LICENSE	131
	8.32.3	DESCRIPTION	131
8.33	/home/	raphael/Programmieren/BPraktikum/SWE/src/scenarios/SWE_VisInfo.hh File Reference	132
	8.33.1	Detailed Description	132
	8.33.2	LICENSE	132
	8.33.3	DESCRIPTION	132
8.34	/home/	raphael/Programmieren/BPraktikum/SWE/src/testing/testing_scenario.hh File Reference	132
	8.34.1	Detailed Description	132
	8.34.2	LICENSE	133
	8.34.3	DESCRIPTION	133
8.35	/home/	raphael/Programmieren/BPraktikum/SWE/src/tools/help.hh File Reference	133
	8.35.1	Detailed Description	133
	8.35.2	LICENSE	134
	8.35.3	DESCRIPTION	134
	8.35.4	Function Documentation	134
		8.35.4.1 generateBaseFileName	134
		8.35.4.2 generateContainerFileName	134

CONTENTS xix

	8.35.4.3 generateFileName	134
	8.35.4.4 generateFileName	134
	8.35.4.5 generateFileName	135
8.36	/home/raphael/Programmieren/BPraktikum/SWE/src/tools/Logger.cpp File Reference	135
	8.36.1 Detailed Description	135
	8.36.2 LICENSE	135
8.37	/home/raphael/Programmieren/BPraktikum/SWE/src/tools/Logger.hh File Reference	135
	8.37.1 Detailed Description	136
	8.37.2 LICENSE	136
	8.37.3 DESCRIPTION	136
8.38	/home/raphael/Programmieren/BPraktikum/SWE/src/tools/ProgressBar.hh File Reference	136
	8.38.1 Detailed Description	136
	8.38.2 LICENSE	137
	8.38.3 DESCRIPTION	137
8.39	$/home/raphael/Programmieren/BPraktikum/SWE/src/writer/NetCdfWriter.cpp\ File\ Reference\ .\ .\ .\ .$	137
	8.39.1 Detailed Description	137
	8.39.2 LICENSE	137
	8.39.3 DESCRIPTION	138
8.40	/home/raphael/Programmieren/BPraktikum/SWE/src/writer/NetCdfWriter.hh File Reference	138
	8.40.1 Detailed Description	138
	8.40.2 LICENSE	138
	8.40.3 DESCRIPTION	138
8.41	/home/raphael/Programmieren/BPraktikum/SWE/src/writer/NetCdfWriterCP.cpp File Reference	139
	8.41.1 Detailed Description	139
	8.41.2 LICENSE	139
	8.41.3 DESCRIPTION	139
8.42	/home/raphael/Programmieren/BPraktikum/SWE/src/writer/NetCdfWriterCP.hh File Reference	139
	8.42.1 Detailed Description	140
	8.42.2 LICENSE	140
	8.42.3 DESCRIPTION	140
8.43	/home/raphael/Programmieren/BPraktikum/SWE/src/writer/VtkWriter.cpp File Reference	140
	8.43.1 Detailed Description	140
	8.43.2 LICENSE	140
	8.43.3 DESCRIPTION	141
8.44	/home/raphael/Programmieren/BPraktikum/SWE/src/writer/VtkWriter.hh File Reference	141
	8.44.1 Detailed Description	141
	8.44.2 LICENSE	141
	8.44.3 DESCRIPTION	141
8.45		141
	8.45.1 Detailed Description	141

CONTENTS

	8.45.2 LICENSE	142
	8.45.3 DESCRIPTION	142
8.46	/home/raphael/Programmieren/BPraktikum/SWE/submodules/f-wave-solver/src/FWave.cpp File Reference	142
	8.46.1 Detailed Description	142
8.47	/home/raphael/Programmieren/BPraktikum/SWE/submodules/f-wave-solver/src/FWave.hpp File Reference	142
	8.47.1 Detailed Description	142
8.48	/home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/main.cpp File Reference	143
	8.48.1 Detailed Description	143
8.49	/home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/scenarios/constantflow.h File Reference	143
	8.49.1 Detailed Description	144
8.50	/home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/scenarios/dambreak.h	144
	8.50.1 Detailed Description	144
8.51	/home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/scenarios/dambreakbathy.h File Reference	145
	8.51.1 Detailed Description	145
8.52	/home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/scenarios/eisbach.h File Reference	146
	8.52.1 Detailed Description	146
8.53	/home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/scenarios/rarerare.h File Reference	147
	8.53.1 Detailed Description	147
8.54	/home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/scenarios/ShockShock.h File Reference	147
	8.54.1 Detailed Description	148
8.55	/home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/scenarios/Subcriticalflow.h File Reference	148
	8.55.1 Detailed Description	148
8.56	/home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/scenarios/Supercritical_flow.h File Reference	149
	8.56.1 Detailed Description	149
8.57	/home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/tools/args.cpp File Reference	150
	8.57.1 Detailed Description	150
8.58	/home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/tools/args.h File Reference	e150
	8.58.1 Detailed Description	151
8.59	/home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/tools/logger.cpp File Reference	151
	8.59.1 Detailed Description	151
8 60	/home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/tools/logger.h File Refer-	.01
5.00	ence	152

CONTENTS xxi

Index		157
	8.65.1 Detailed Description	156
8.65	/home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/writer/VtkWriter.h File Reference	156
	8.64.1 Detailed Description	155
8.64	/home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/writer/ConsoleWriter.h File Reference	155
	8.63.1 Detailed Description	154
8.63	/home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/WavePropagation.h File Reference	
	8.62.1 Detailed Description	153
8.62	/home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/WavePropagation.cpp File Reference	153
	8.61.1 Detailed Description	153
8.61	$/home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/types.h\ File\ Reference \ .$	153
	8.60.1 Detailed Description	152

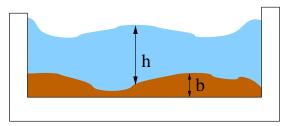
### **Chapter 1**

### **SWE - A Simple Shallow Water Code**

SWE is a teaching code that implements simple Finite Volumes models that solve the shallow water equations - in a problem setting as it would be used for tsunami simulation.

### 1.1 The Shallow Water Equations

The shallow water equations describe the behaviour of a fluid, in particular water, of a certain (possibly varying) depth h in a two-dimensional domain – imagine, for example, a puddle of water or a shallow pond (and compare the 1D sketch given below). The main modelling assumption is that we can neglect effects of flow in vertical direction. The resulting model proved to be useful for the simulation of tsunami propagation (with appropriate extensions). While an ocean can hardly be considered as "shallow" in the usual sense, tsunami waves (in contrast to regular waves induced by wind, e.g.) affect the entire water column, such that effects of vertical flow can again be neglected. To allow for a non-even sea bottom (as required for accurate modelling of tsunamis), we include the elevation b of the sea floor in our model:



The shallow water equations describe the changes of water depth h and horizontal velocities  $v_X$  and  $v_y$  (in the resp. coordinate directions) over time, depending on some initial conditions – in the case of tsunami simulation, these initial conditions could, for example, result from an initial elevation of the sea floor caused by an earthquake. The respective changes in time can be described via a system of partial differential equations:

$$\frac{\partial h}{\partial t} + \frac{\partial (v_x h)}{\partial x} + \frac{\partial (v_y h)}{\partial y} = 0$$

$$\frac{\partial (hv_x)}{\partial t} + \frac{\partial (hv_x v_x)}{\partial x} + \frac{\partial (hv_y v_x)}{\partial y} + \frac{1}{2}g \frac{\partial (h^2)}{\partial x} = -gh \frac{\partial b}{\partial x},$$

$$\frac{\partial (hv_y)}{\partial t} + \frac{\partial (hv_x v_y)}{\partial x} + \frac{\partial (hv_y v_y)}{\partial y} + \frac{1}{2}g \frac{\partial (h^2)}{\partial y} = -gh \frac{\partial b}{\partial y},$$

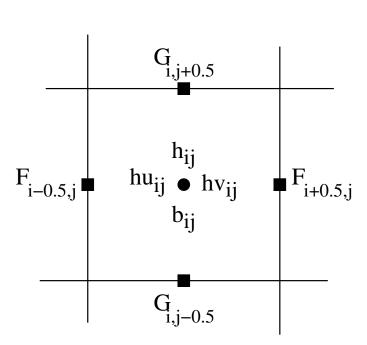
The equation for h is obtained, if we examine the conservation of mass in a control volume. The equations for  $hv_X$  and  $hv_Y$  result from conservation of momentum (note that h is directly related to the volume, and thus the mass of the water – thus  $hv_X$  can be interpreted as a momentum).

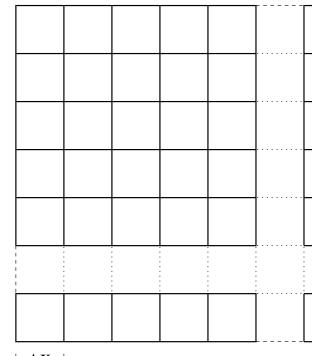
The two terms involving g model a gravity-induced force (g being the constant for the gravitational acceleration,  $g = 9.81 \text{ ms}^{-2}$ ), which results from the hydrostatic pressure. The right-hand-side source terms model the effect of an uneven ocean floor (b obtained from respective bathymetry data).

#### 1.1.1 Finite Volume Discretisation

The shallow water equations are usually too difficult to be solved exactly - hence, SWE implements simple discrete models as an approximation. As the applied numerical method (typically a Finite Volume discretization) may vary, we will stick to the basics at this point.

First, SWE assumes that the unknown functions h(t,x,y),  $hu(t,x,y) := h(t,x,y) v_X(t,x,y)$ ,  $hv(t,x,y) := h(t,x,y) v_Y(t,x,y)$ , as well as the given sea bottom level h(t,x,y), are approximated on a Cartesian mesh of grid cells, as illustrated below. In each grid cell, with indices h(t,y), the unknowns have constant values h(t,x),  $hv(t,x,y) := h(t,x,y) v_Y(t,x,y)$ , as well as the given sea bottom level h(t,x), are approximated on a Cartesian mesh of grid cells, as illustrated below. In each grid cell, with indices h(t,x), the unknowns have constant values h(t,x), hv(t,x), and h(t,x).





### $\vdash \Delta X \dashv$

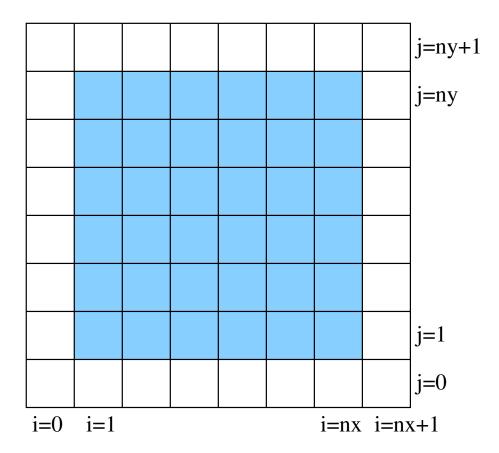
#### 1.1.2 Computing Numerical Fluxes at Edges and Euler Time-Stepping

The details of the numerical schemes are too complicated to be described in this overview. Please refer to the accompanying material. To put it short, we successively perform two main computational steps:

- we compute so-called **numerical fluxes** on each edge of the grid (which approximate the transfer of mass or momentum between grid cells),
- based on these numerical fluxes, we then update the unknowns in each cell.

#### 1.2 Implementation and base class SWE Block

For the simulation of the shallow water model, we thus require a regular Cartesian grid, where each grid cell carries the respective unknowns - water level, momentum in x- and y-direction, and bathymetry data. The central data structures for Cartesian grid and arrays of unknowns are provided with the abstract base class SWE\_Block, which has four 2D arrays SWE\_Block::h, SWE\_Block::hu, SWE\_Block::hv, and SWE\_Block::b. To implement the behaviour of the fluid at boundaries, and also to allow the connection of several grid blocks (for parallelization or just to build more complicated computational domains), each array has an additional layer of so-called *ghost cells*, as illustrated in the following figure:



#### 1.2.1 Parallelisation and Different Models

In each time step, our numerical algorithm will compute the flux terms for each edge of the computational domain. To compute the fluxes, we require the values of the unknowns in both adjacent cells. At the boundaries of the fluid domain, the ghost layer makes sure that we also have two adjacent cells for the cell edges on the domain boundary. The values in the ghost layer cells will be set to values depending on the values in the adjacent fluid domain. We will model three different situations: {description} {Outflow:} {h}, {u}, and {v} in the ghost cell are set to the same value as in the adjacent fluid cell. This models the situation that the unknowns do not change across the domain boundary (undisturbed outflow). {Wall:} At a wall, the velocity component normal to the boundary should be \$0\$, such that no fluid can cross the boundary. To model this case, we set the normal velocity, e.g. {u[0]} at the left boundary, to the negative value of the adjacent cell: {-u[1]}. The interpolated value at the boundary edge will then be \$0\$ ({h} is identical in both cells due to the imposed boundary condition). The other two variables are set in the same way as for the outflow situation. {Connect:} With the connect case, we can connect a domain at two boundaries. If we connect the left and right boundary, we will obtain a periodically repeated domain. Here, all ghost values are determined by the values of the unknowns in the fluid cell adjacent to the connected boundary. {description}

To implement the boundary conditions, the class {SWE\_Block} contains an array of four enum variables, {boundary[4]} (for left/right/bottom/top boundary), that can take the values OUTFLOW, WALL, and CONNECT.

#### 1.2.2 Multiple Blocks

Via the connect boundary condition, it is also possible to connect several Cartesian grid blocks to build a more complicated domain. Figure fig:connect} illustrates the exchange of ghost values for two connected blocks.

To store the neighbour block in case of a CONNECT boundary, SWE\_Block contains a further array of four pointers, neighbour [4] (for left/right/bottom/top boundary), that will store a pointer to the connected adjacent SWE\_Block.

The respective block approach can also be exploited for parallelisation: the different blocks would then be assigned to the available processors (or processor cores) – each processor (core) works on its share of blocks, while the program has to make sure to keep the values in the ghost cells up to date (which requires explicit communication in

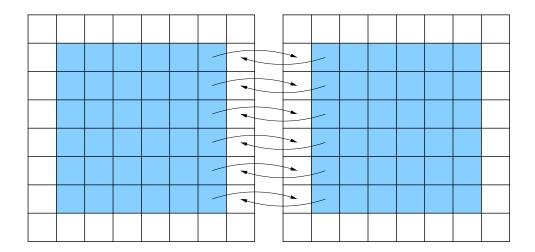


Figure 1.1: Exchange of values in ghost layers between two connected SWE\_Blocks.

the case of distributed-memory computers).

#### 1.2.3

For each time step, our solver thus performs the following steps – each step is implemented via a separate member function of the class {SWE\_Block}: {enumerate} set the values at the boundaries: {setBoundaryLayer()}; compute the flux terms for all edges: {computeFluxes()}; from the flux terms, compute the in/outflow balance for each cell, and compute the new values of the unknowns for the next time step: {eulerTimestep()}. {enumerate}

#### 1.2.4

The class {SWE\_Block} contains further methods that will write the numerical solution into a sequence of files that can be read by the visualisation package ParaView (just enter the respective folder from ParaView – the files will be recognised and displayed as one project). ParaView allows to visualise the computed time-dependent solution (as "movie" or in single-step mode). ParaView is pretty self-explanatory for our purposes, but provides an online help for further instructions.

#### 1.2.5

We also provide a CUDA implementation of the simulation code (requires a computer with a CUDA-capable GPU, together with the respective drivers – visit NVIDIA's website on CUDA for details on implementation). Apart from the fact that the simulation usually runs a lot faster on the GPU, the program is also capable of plotting the computing solution (water surface) "on the fly".

Finally: whoever thinks that they can do a better (faster, ...) implementation (visualisation, ...) of the provided code is more than welcome to do so! Feel free to contribute to SWE - for questions, just contact Michael Bader (bader@in.tum.de).

### **Chapter 2**

## **Todo List**

Member io::VtkWriter::VtkWriter (const std::string &i\_fileName, const Float2D &i\_b, const BoundarySize &i\_boundarySize, int i\_nY, int i\_nY, float i\_dX, float i\_dY, int i\_offsetX=0, int i\_offsetY=0)

This version can only handle a boundary layer of size 1

6 **Todo List** 

### **Chapter 3**

# **Deprecated List**

Member generateFileName (std::string baseName, int timeStep)

Member generateFileName (std::string baseName, int timeStep, int block\_X, int block\_Y, std::string i\_file-Extension=".vts")

Member generateFileName (std::string i\_baseName, int i\_blockPositionX, int i\_blockPositionY, std::string i\_fileExtension=".nc")

8 Deprecated List

# Chapter 4

## **Hierarchical Index**

### 4.1 Class Hierarchy

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

Camera
Controller
TestSuite
DimensionalSplittingTest
FWaveSolverTest
SWE_TsunamiScenarioTest
SWE_TsunamiScenarioTest
Float1D
Float2D
solver::FWave < float >
io::BoundarySize
io::Writer
io::NetCdfWriter
io::NetCdfWriter
io::VtkWriter
scenarios::ConstantFlow
scenarios::DamBreak
scenarios::Eisbach
scenarios::RareRare
scenarios::ShockShock
scenarios::Subcrit
scenarios::Supercrit
Shader
Simulation
solver::FWave < T >
SWE AsagiGrid
_ 0
SWE_BlockCUDA
SWE_RusanovBlockCUDA
SWE_WavePropagationBlockCuda
SWE_DimensionalSplitting
SWE_RusanovBlock
SWE_WavePropagationBlock
SWE Block1D
SWE_Scenario
SWE ArtificialTsunamiScenario
SWE_AsagiScenario

10 Hierarchical Index

SWE_BathymetryDamBreakScenario	. 41
SWE_DamBreakScenario	. 55
SWE_NetCDFScenario	. 58
SWE_NetCDFCheckpointScenario	. 57
SWE_TsunamiScenario	. 68
SWE_RadialDamBreakScenario	. 59
SWE_SeaAtRestScenario	. 65
SWE_SplashingConeScenario	. 66
SWE_SplashingPoolScenario	. 67
SWE_TestingScenario	. 67
SWE_VisInfo	71
SWE AsagiJapanSmallVisInfo	. 37
SWE_BathymetryDamBreakVisInfo	. 42
Text	78
ools::Args	
ools::Logger	79
ools::ProgressBar	86
/BO	87
Visualization	87
NavePropagation	89
writer::ConsoleWriter	91
writer··VtkWriter	91

# **Chapter 5**

## **Class Index**

### 5.1 Class List

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

Camera	5
Controller	
DimensionalSplittingTest	8
Float1D	9
Float2D	9
FWaveSolverTest	0
io::BoundarySize	1
io::NetCdfWriter	2
io::VtkWriter	4
io::Writer	6
scenarios::ConstantFlow	27
scenarios::DamBreak	8
scenarios::Eisbach	9
scenarios::RareRare	0
scenarios::ShockShock	0
scenarios::Subcrit	1
scenarios::Supercrit	2
Shader	2
Simulation	4
solver::FWave< T >	5
SWE_ArtificialTsunamiScenario	6
SWE_AsagiGrid	7
SWE_AsagiJapanSmallVisInfo	7
SWE_AsagiScenario	8
SWE_BathymetryDamBreakScenario	1
SWE_BathymetryDamBreakVisInfo	2
SWE_Block	2
SWE_Block1D	1
SWE_BlockCUDA	1
SWE_DamBreakScenario	5
SWE_DimensionalSplitting	6
SWE_NetCDFCheckpointScenario	7
SWE_NetCDFScenario	8
SWE_RadialDamBreakScenario	9
SWE_RusanovBlock	9
SWE_RusanovBlockCUDA	2
SWE_Scenario	4
SWE_SeaAtRestScenario	5

12 Class Index

SWE_SplashingConeScenario	66
WE_SplashingPoolScenario	67
WE_TestingScenario	67
WE_TsunamiScenario	68
WE_TsunamiScenarioTest	70
WE_VisInfo	71
WE_WavePropagationBlock	72
WE_WavePropagationBlockCuda	74
ext	78
pols::Args	79
pols::Logger	79
pols::ProgressBar	86
/BO	87
'isualization	87
VavePropagation	89
vriter::ConsoleWriter	91
vritor:::VtkWritor	<b>Q1</b>

# **Chapter 6**

## File Index

### 6.1 File List

Here is a list of all documented files with brief descriptions:

/home/raphael/Programmieren/BPraktikum/SWE/src/blocks/SWE_Block.cpp	115
/home/raphael/Programmieren/BPraktikum/SWE/src/blocks/SWE_Block.hh	116
/home/raphael/Programmieren/BPraktikum/SWE/src/blocks/ <b>SWE_DimensionalSplitting.hh</b>	??
/home/raphael/Programmieren/BPraktikum/SWE/src/blocks/SWE_WavePropagationBlock.cpp	117
/home/raphael/Programmieren/BPraktikum/SWE/src/blocks/SWE_WavePropagationBlock.hh	118
/home/raphael/Programmieren/BPraktikum/SWE/src/blocks/cuda/SWE_BlockCUDA.cu	93
/home/raphael/Programmieren/BPraktikum/SWE/src/blocks/cuda/SWE_BlockCUDA.hh	94
/home/raphael/Programmieren/BPraktikum/SWE/src/blocks/cuda/SWE_BlockCUDA_kernels.cu	96
/home/raphael/Programmieren/BPraktikum/SWE/src/blocks/cuda/SWE_BlockCUDA_kernels.hh	98
/home/raphael/Programmieren/BPraktikum/SWE/src/blocks/cuda/SWE_WavePropagationBlockCuda.cu	100
/home/raphael/Programmieren/BPraktikum/SWE/src/blocks/cuda/SWE_WavePropagationBlockCuda.hh	101
/home/raphael/Programmieren/BPraktikum/SWE/src/blocks/cuda/SWE_WavePropagationBlockCuda	
kernels.cu	101
/home/raphael/Programmieren/BPraktikum/SWE/src/blocks/cuda/SWE_WavePropagationBlockCuda	
kernels.hh	105
/home/raphael/Programmieren/BPraktikum/SWE/src/blocks/rusanov/SWE_RusanovBlock.cpp	109
/home/raphael/Programmieren/BPraktikum/SWE/src/blocks/rusanov/SWE_RusanovBlock.hh	110
/home/raphael/Programmieren/BPraktikum/SWE/src/blocks/rusanov/SWE_RusanovBlockCUDA.cu	111
/home/raphael/Programmieren/BPraktikum/SWE/src/blocks/rusanov/SWE_RusanovBlockCUDA.hh	112
/home/raphael/Programmieren/BPraktikum/SWE/src/blocks/rusanov/SWE_RusanovBlockCUDA_kernels	
cu	113
/home/raphael/Programmieren/BPraktikum/SWE/src/blocks/rusanov/SWE_RusanovBlockCUDA_kernels	
hh	114
/home/raphael/Programmieren/BPraktikum/SWE/src/examples/swe_DimensionalSplitting.cpp	119
/home/raphael/Programmieren/BPraktikum/SWE/src/examples/swe_mpi.cpp	120
/home/raphael/Programmieren/BPraktikum/SWE/src/examples/swe_simple.cpp	123
/home/raphael/Programmieren/BPraktikum/SWE/src/opengl/ <b>camera.h</b>	??
/home/raphael/Programmieren/BPraktikum/SWE/src/opengl/ <b>controller.h</b>	??
/home/raphael/Programmieren/BPraktikum/SWE/src/opengl/ <b>shader.h</b>	??
/home/raphael/Programmieren/BPraktikum/SWE/src/opengl/ <b>simulation.h</b>	??
/home/raphael/Programmieren/BPraktikum/SWE/src/opengl/ <b>text.h</b>	??
/home/raphael/Programmieren/BPraktikum/SWE/src/opengl/vbo.cpp	124
/home/raphael/Programmieren/BPraktikum/SWE/src/opengl/vbo.h	124
/home/raphael/Programmieren/BPraktikum/SWE/src/opengl/ <b>visualization.h</b>	??
/home/raphael/Programmieren/BPraktikum/SWE/src/scenarios/SWE_ArtificialTsunamiScenario.hh	125
/home/raphael/Programmieren/BPraktikum/SWE/src/scenarios/SWE_AsagiScenario.cpp	126
/home/raphael/Programmieren/BPraktikum/SWE/src/scenarios/SWE_AsagiScenario.hh	126
/home/raphael/Programmieren/BPraktikum/SWE/src/scenarios/SWE_AsagiScenario_vis.hh	127

14 File Index

/home/raphael/Programmieren/BPraktikum/SWE/src/scenarios/SWE_NetCDFCheckpointScenario.hh	128
/home/raphael/Programmieren/BPraktikum/SWE/src/scenarios/ <b>SWE_NetCDFScenario.hh</b>	??
_	129
/home/raphael/Programmieren/BPraktikum/SWE/src/scenarios/SWE_simple_scenarios.hh	130
/home/raphael/Programmieren/BPraktikum/SWE/src/scenarios/ <b>SWE_simple_scenarios_vis.hh</b>	??
/home/raphael/Programmieren/BPraktikum/SWE/src/scenarios/SWE_TsunamiScenario.hh	131
/home/raphael/Programmieren/BPraktikum/SWE/src/scenarios/ <b>SWE_TsunamiScenarioTest.hh</b>	??
/home/raphael/Programmieren/BPraktikum/SWE/src/scenarios/ <b>SWE_TsunamiScenarioTest.t.h</b>	??
/home/raphael/Programmieren/BPraktikum/SWE/src/scenarios/SWE_VisInfo.hh	132
/home/raphael/Programmieren/BPraktikum/SWE/src/testing/ <b>DimensionalSplittingTest.t.h</b>	??
/home/raphael/Programmieren/BPraktikum/SWE/src/testing/testing_scenario.hh	132
/home/raphael/Programmieren/BPraktikum/SWE/src/tools/help.hh	133
/home/raphael/Programmieren/BPraktikum/SWE/src/tools/Logger.cpp	135
/home/raphael/Programmieren/BPraktikum/SWE/src/tools/Logger.hh	135
/home/raphael/Programmieren/BPraktikum/SWE/src/tools/ProgressBar.hh	136
/home/raphael/Programmieren/BPraktikum/SWE/src/writer/NetCdfWriter.cpp	137
/home/raphael/Programmieren/BPraktikum/SWE/src/writer/NetCdfWriter.hh	138
/home/raphael/Programmieren/BPraktikum/SWE/src/writer/NetCdfWriterCP.cpp	139
/home/raphael/Programmieren/BPraktikum/SWE/src/writer/NetCdfWriterCP.hh	139
/home/raphael/Programmieren/BPraktikum/SWE/src/writer/VtkWriter.cpp	140
/home/raphael/Programmieren/BPraktikum/SWE/src/writer/VtkWriter.hh	141
/home/raphael/Programmieren/BPraktikum/SWE/src/writer/Writer.hh	141
/home/raphael/Programmieren/BPraktikum/SWE/submodules/f-wave-solver/src/FWave.cpp	142
/home/raphael/Programmieren/BPraktikum/SWE/submodules/f-wave-solver/src/FWave.hpp	142
$/home/raphael/Programmieren/BPraktikum/SWE/submodules/f-wave-solver/src/ \textbf{FWaveSolverTest.hpp} \ .$	??
/home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/main.cpp	143
/home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/types.h	153
$/home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/WavePropagation.cpp \ . \ . \ . \ . \ . \ . \ . \ . \ . \$	153
/home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/WavePropagation.h	154
$/home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/scenarios/constant\_flow.h \\$	143
	144
/home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/scenarios/dambreak_bathy.h	145
/home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/scenarios/eisbach.h	146
/home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/scenarios/rarerare.h	147
$/home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/scenarios/ShockShock.h \ . \ . \ . \ .$	147
$/home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/scenarios/Subcritical\_flow.h \\ .$	148
/home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/scenarios/Supercritical_flow.h	149
/home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/tools/args.cpp	150
	150
/home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/tools/logger.cpp	151
1 0	152
	155
/home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/writer/VtkWriter.h	156

# **Chapter 7**

# **Class Documentation**

# 7.1 Camera Class Reference

# **Public Member Functions**

- Camera (const char \*window\_title)
- void setCamera ()
- void reset ()
- void viewDistance (float viewDistance)
- void orient (float angX, float angY)
- void zoomln (float scaleFactor)
- void zoomOut (float scaleFactor)
- void startPanning (int xPos, int yPos)
- void panning (int newX, int newY)
- void displayImage ()

# 7.1.1 Constructor & Destructor Documentation

7.1.1.1 Camera::Camera ( const char \* window\_title )

# Constructor

#### **Parameters**

view_distance	initial view distance from the origin
window_title	title of the current window

# 7.1.2 Member Function Documentation

# 7.1.2.1 void Camera::displayImage ( )

Calculates the current framerate, updates the window title and swaps framebuffers to display the new image

7.1.2.2 void Camera::orient (float angle\_dX, float angle\_dY)

Increment viewing orientation of the camera

#### **Parameters**

angle_dX	angle relative to the x-axis
angle dY	angle relative to the rotated y-axis

7.1.2.3 void Camera::panning (int newX, int newY)

User drags our object. Transform screen coordinates into world coordinates and update the objects position

7.1.2.4 void Camera::setCamera ( )

Set the camera via gluLookAt and set the light position afterwards

7.1.2.5 void Camera::startPanning (int xPos, int yPos)

User starts dragging. Remember the old mouse coordinates.

7.1.2.6 void Camera::viewDistance ( float viewDistance )

Set the view distance

7.1.2.7 void Camera::zoomln ( float scaleFactor )

Zoom in

**Parameters** 

scaleFactor	factor which is used for zooming

7.1.2.8 void Camera::zoomOut (float scaleFactor)

Zoom out

**Parameters** 

scaleFactor	factor which is used for zooming

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

- $\bullet \ \ /home/raphael/Programmieren/BPraktikum/SWE/src/opengl/camera.h$
- /home/raphael/Programmieren/BPraktikum/SWE/src/opengl/camera.cpp

# 7.2 Controller Class Reference

# **Public Member Functions**

- Controller (Simulation \*sim, Visualization \*vis)
- bool handleEvents ()
- bool hasFocus ()
- bool isPaused ()

# 7.2.1 Constructor & Destructor Documentation

7.2.1.1 Controller::Controller ( Simulation \* sim, Visualization \* vis )

Constructor

#### **Parameters**

sim	instance of simulation class
vis	instance of visualization class

# 7.2.2 Member Function Documentation

# 7.2.2.1 bool Controller::handleEvents ( )

Process all user events in a loop Returns true, when user wants to quit

7.2.2.2 bool Controller::hasFocus ( )

Returns true, when window has focus

7.2.2.3 bool Controller::isPaused ( )

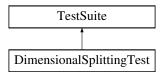
Return whether program is currently paused

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

- · /home/raphael/Programmieren/BPraktikum/SWE/src/opengl/controller.h
- /home/raphael/Programmieren/BPraktikum/SWE/src/opengl/controller.cpp

# 7.3 Dimensional Splitting Test Class Reference

Inheritance diagram for DimensionalSplittingTest:



# **Public Member Functions**

- void testCompareNetUpdates ()
- · void testHorizontalSteadyState ()

# **Static Public Attributes**

- static const int **row** = 0
- static const float dt = 0.01
- static const float accuracy = 1.0E-6
- static const int **nx** = 200
- static const int ny = 1
- static const float dx = 1.f
- static const float dy = 1.f

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

• /home/raphael/Programmieren/BPraktikum/SWE/src/testing/DimensionalSplittingTest.t.h

# 7.4 Float1D Class Reference

```
#include <help.hh>
```

#### **Public Member Functions**

- Float1D (float \*\_elem, int \_rows, int \_stride=1)
- float & operator[] (int i)
- const float & operator[] (int i) const
- float \* elemVector ()
- int getSize () const

# 7.4.1 Detailed Description

class Float1D is a proxy class that can represent, for example, a column or row vector of a Float2D array, where row (sub-)arrays are stored with a respective stride. Besides constructor/deconstructor, the class provides overloading of the []-operator, such that elements can be accessed as v[i] (independent of the stride). The class will never allocate separate memory for the vectors, but point to the interior data structure of Float2D (or other "host" data structures).

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

• /home/raphael/Programmieren/BPraktikum/SWE/src/tools/help.hh

# 7.5 Float2D Class Reference

```
#include <help.hh>
```

# **Public Member Functions**

- Float2D (int \_cols, int \_rows)
- float \* operator[] (int i)
- float const \* operator[] (int i) const
- float \* elemVector ()
- int getRows () const
- int **getCols** () const
- Float1D getColProxy (int i)
- Float1D getRowProxy (int j)

# 7.5.1 Detailed Description

class Float2D is a very basic helper class to deal with 2D float arrays: indices represent columns (1st index, "horizontal"/x-coordinate) and rows (2nd index, "vertical"/y-coordinate) of a 2D grid; values are sequentially ordered in memory using "column major" order. Besides constructor/deconstructor, the class provides overloading of the []-operator, such that elements can be accessed as a[i][i].

# 7.5.2 Constructor & Destructor Documentation

```
7.5.2.1 Float2D::Float2D ( int _cols, int _rows ) [inline]
```

Constructor

#### **Parameters**

_cols	number of columns (i.e., elements in horizontal direction)
_rows	rumber of rows (i.e., elements in vertical directions)

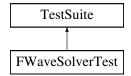
The documentation for this class was generated from the following file:

• /home/raphael/Programmieren/BPraktikum/SWE/src/tools/help.hh

# 7.6 FWaveSolverTest Class Reference

#include <FWaveSolverTest.hpp>

Inheritance diagram for FWaveSolverTest:



# **Public Member Functions**

- void testEigenvalues ()
- void testFlux ()
- void testEigencoefficients ()
- void testSteadyState ()
- void testSupersonic ()
- void testBathymetry ()
- · void testDryCells ()

# 7.6.1 Detailed Description

This is the cxxtest test-suite for the FWave template class.

# 7.6.2 Member Function Documentation

7.6.2.1 void FWaveSolverTest::testBathymetry() [inline]

check the handling of bathymetric data

a steady state will be created, but with different water depths

**7.6.2.2** void FWaveSolverTest::testEigencoefficients() [inline]

testEigencoefficients will do a valueCheck on the function "eigencoeffis" of the template "FWave"

7.6.2.3 void FWaveSolverTest::testEigenvalues ( ) [inline]

this function will test the private function roeEigenvals of the template FWave by performing two value checks

```
7.6.2.4 void FWaveSolverTest::testFlux() [inline]
```

testFlux will perform a fast value check on the private function "flux" of the template "FWave"

```
7.6.2.5 void FWaveSolverTest::testSteadyState() [inline]
```

testSteadyState will calculate the net updates for identical water columns and momentum on both sides, which have to be equal to zero

```
7.6.2.6 void FWaveSolverTest::testSupersonic ( ) [inline]
```

testSupersonic will check the function computeNetUpdates for correct behavior in the supersonic case (both eigenvalues greater/less than zero)

- · greater than zero: the net-updates on the left have to be zero
- · less than zero: the net-updates on the right have to be zero

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

/home/raphael/Programmieren/BPraktikum/SWE/submodules/f-wave-solver/src/FWaveSolverTest.hpp

# 7.7 io::BoundarySize Struct Reference

```
#include <Writer.hh>
```

# **Public Member Functions**

- int & operator[] (unsigned int i)
- int operator[] (unsigned int i) const

# **Public Attributes**

• int boundarySize [4]

# 7.7.1 Detailed Description

This struct is used so we can initialize this array in the constructor.

# 7.7.2 Member Data Documentation

# 7.7.2.1 int io::BoundarySize::boundarySize[4]

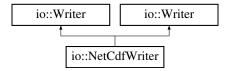
boundarySize[0] == left boundarySize[1] == right boundarySize[2] == bottom boundarySize[3] == top

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

• /home/raphael/Programmieren/BPraktikum/SWE/src/writer/Writer.hh

# 7.8 io::NetCdfWriter Class Reference

Inheritance diagram for io::NetCdfWriter:



#### **Public Member Functions**

- NetCdfWriter (const std::string &i\_fileName, const Float2D &i\_b, const BoundarySize &i\_boundarySize, int i\_nX, int i\_nY, float i\_dX, float i\_dY, float ETime, bool newfile=false, float i\_originX=0., float i\_originY=0., unsigned int i\_flush=0)
- virtual ∼NetCdfWriter ()
- void writeTimeStep (const Float2D &i\_h, const Float2D &i\_hu, const Float2D &i\_hv, float i\_time)
- void writeBoundary (char \*up, char \*bottom, char \*left, char \*right)
- NetCdfWriter (char \*CPfile)
- NetCdfWriter (const std::string &i\_fileName, const Float2D &i\_b, const BoundarySize &i\_boundarySize, int i\_nX, int i\_nY, float i\_dX, float i\_dY, float ETime, float i\_originX=0., float i\_originY=0., unsigned int i\_flush=0)
- void writeTimeStep (const Float2D &i h, const Float2D &i hu, const Float2D &i hv, float i time)

# **Additional Inherited Members**

# 7.8.1 Constructor & Destructor Documentation

7.8.1.1 io::NetCdfWriter::NetCdfWriter ( const std::string &  $i\_baseName$ , const Float2D &  $i\_b$ , const BoundarySize &  $i\_boundarySize$ , int  $i\_nX$ , int  $i\_nY$ , float  $i\_dX$ , float  $i\_dY$ , float ETime, bool newfile = false, float  $i\_originX = 0$ ., float  $i\_originY = 0$ ., unsigned int  $i\_flush = 0$ )

Create a netCdf-file Any existing file will be replaced.

#### **Parameters**

i_baseName	base name of the netCDF-file to which the data will be written to.
i_nX	number of cells in the horizontal direction.
i_nY	number of cells in the vertical direction.
i_dX	cell size in x-direction.
i_dY	cell size in y-direction.
i_originX	
i_originY	
i_flush	If $>$ 0, flush data to disk every i_flush write operation
i_dynamic-	
Bathymetry	

7.8.1.2 io::NetCdfWriter::~NetCdfWriter( ) [virtual]

Destructor of a netCDF-writer.

7.8.1.3 io::NetCdfWriter::NetCdfWriter ( const std::string & *i\_baseName*, const Float2D & *i\_b*, const BoundarySize & *i\_boundarySize*, int *i\_nX*, int *i\_nY*, float *i\_dX*, float *i\_dY*, float *ETime*, float *i\_originX* = 0 . , float *i\_originY* = 0 . , unsigned int *i\_flush* = 0 )

Create a netCdf-file Any existing file will be replaced.

#### **Parameters**

i_baseName	base name of the netCDF-file to which the data will be written to.
i_nX	number of cells in the horizontal direction.
i_nY	number of cells in the vertical direction.
i_dX	cell size in x-direction.
i_dY	cell size in y-direction.
i_originX	
i_originY	
i_flush	If $>$ 0, flush data to disk every i_flush write operation
i_dynamic-	
Bathymetry	

# 7.8.2 Member Function Documentation

7.8.2.1 void io::NetCdfWriter::writeTimeStep ( const Float2D & i\_h, const Float2D & i\_hu, const Float2D & i\_hv, float i\_time ) [virtual]

Writes one time step

#### **Parameters**

ĺ	i h	water heights at a given time step.
	1_11	water neights at a given time step.
	i_hu	momentums in x-direction at a given time step.
	i_hv	momentums in y-direction at a given time step.
	i_time	simulation time of the time step.

Implements io::Writer.

7.8.2.2 void io::NetCdfWriter::writeTimeStep ( const Float2D & i\_h, const Float2D & i\_hu, const Float2D & i\_hv, float i\_time ) [virtual]

Writes the unknwons to a netCDF-file (-> constructor) with respect to the boundary sizes.

boundarySize[0] == left boundarySize[1] == right boundarySize[2] == bottom boundarySize[3] == top

### **Parameters**

i_h	water heights at a given time step.
i_hu	momentums in x-direction at a given time step.
<u>i_</u> hv	momentums in y-direction at a given time step.
i_boundarySize	size of the boundaries.
i_time	simulation time of the time step.

Implements io::Writer.

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

- $\bullet \ \ /home/raphael/Programmieren/BPraktikum/SWE/src/writer/NetCdfWriter.hh$
- /home/raphael/Programmieren/BPraktikum/SWE/src/writer/NetCdfWriterCP.hh
- /home/raphael/Programmieren/BPraktikum/SWE/src/writer/NetCdfWriter.cpp
- /home/raphael/Programmieren/BPraktikum/SWE/src/writer/NetCdfWriterCP.cpp

# 7.9 io::VtkWriter Class Reference

Inheritance diagram for io::VtkWriter:



# **Public Member Functions**

- VtkWriter (const std::string &i\_fileName, const Float2D &i\_b, const BoundarySize &i\_boundarySize, int i\_nX, int i\_nY, float i\_dX, float i\_dY, int i\_offsetX=0, int i\_offsetY=0)
- void writeTimeStep (const Float2D &i\_h, const Float2D &i\_hu, const Float2D &i\_hv, float i\_time)

# **Additional Inherited Members**

#### 7.9.1 Constructor & Destructor Documentation

7.9.1.1 io::VtkWriter::VtkWriter ( const std::string & *i\_baseName*, const Float2D & *i\_b*, const BoundarySize & *i\_boundarySize*, int *i\_nX*, int *i\_nY*, float *i\_dX*, float *i\_dY*, int *i\_offsetX* = 0, int *i\_offsetY* = 0)

Creates a vtk file for each time step. Any existing file will be replaced.

#### **Parameters**

i_baseName	base name of the netCDF-file to which the data will be written to.
i_nX	number of cells in the horizontal direction.
i_nY	number of cells in the vertical direction.
i_dX	cell size in x-direction.
i_dY	cell size in y-direction.
i_offsetX	x-offset of the block
i_offsetY	y-offset of the block
i_dynamic-	
Bathymetry	

Todo This version can only handle a boundary layer of size 1

#### 7.9.2 Member Function Documentation

7.9.2.1 void io::VtkWriter::writeTimeStep ( const Float2D & i\_h, const Float2D & i\_hu, const Float2D & i\_hv, float i\_time )
[virtual]

# Writes one time step

#### **Parameters**

i_h	water heights at a given time step.
i_hu	momentums in x-direction at a given time step.
<u>i_</u> hv	momentums in y-direction at a given time step.
i_time	simulation time of the time step.

Implements io::Writer.

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

- /home/raphael/Programmieren/BPraktikum/SWE/src/writer/VtkWriter.hh
- /home/raphael/Programmieren/BPraktikum/SWE/src/writer/VtkWriter.cpp

# 7.10 io::Writer Class Reference

Inheritance diagram for io::Writer:



#### **Public Member Functions**

- Writer (const std::string &i\_fileName, const Float2D &i\_b, const BoundarySize &i\_boundarySize, int i\_nX, int i\_nY)
- virtual void writeTimeStep (const Float2D &i\_h, const Float2D &i\_hu, const Float2D &i\_hv, float i\_time)=0

# **Protected Attributes**

· const std::string fileName

file name of the data file

const Float2D & b

(Reference) to bathymetry data

· const BoundarySize boundarySize

Boundary layer size.

const unsigned int nX

dimensions of the grid in x- and y-direction.

- const unsigned int nY
- size\_t timeStep

current time step

# 7.10.1 Constructor & Destructor Documentation

7.10.1.1 io::Writer::Writer ( const std::string & *i\_fileName*, const Float2D & *i\_b*, const BoundarySize & *i\_boundarySize*, int *i\_nX*, int *i\_nY* ) [inline]

# **Parameters**

i_boundarySize	size of the boundaries.

# 7.10.2 Member Function Documentation

7.10.2.1 virtual void io::Writer::writeTimeStep ( const Float2D & i\_h, const Float2D & i\_hu, const Float2D & i\_hu, const Float2D & i\_hu, float i\_time ) [pure virtual]

Writes one time step

# **Parameters**

=
---

i_hu	momentums in x-direction at a given time step.
i_hv	momentums in y-direction at a given time step.
i_time	simulation time of the time step.

Implemented in io::NetCdfWriter, io::NetCdfWriter, and io::VtkWriter.

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

• /home/raphael/Programmieren/BPraktikum/SWE/src/writer/Writer.hh

# 7.11 scenarios::ConstantFlow Class Reference

**Public Member Functions** 

- ConstantFlow (unsigned int size)
- float getHeight (unsigned int pos)
- float getMomentum (unsigned int pos)
- float getBathymetry (unsigned int pos)
- T getCellSize ()

# 7.11.1 Member Function Documentation

```
7.11.1.1 float scenarios::ConstantFlow::getBathymetry (unsigned int pos ) [inline]
```

Returns

floor elevation at pos

```
7.11.1.2 T scenarios::ConstantFlow::getCellSize( ) [inline]
```

Returns

Cell size of one cell (= domain size/number of cells)

```
\textbf{7.11.1.3} \quad \textbf{float scenarios::} \textbf{ConstantFlow::} \textbf{getHeight (unsigned int } \textit{pos } \textbf{)} \quad \texttt{[inline]}
```

Returns

Initial water height at pos

7.11.1.4 float scenarios::ConstantFlow::getMomentum ( unsigned int pos ) [inline]

Returns

Initial momentum at pos

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

/home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/scenarios/constant\_flow.h

#### scenarios::DamBreak Class Reference 7.12

# **Public Member Functions**

- · DamBreak (unsigned int size)
- float getHeight (unsigned int pos)
- float getMomentum (unsigned int pos)
- T getCellSize ()
- DamBreak (unsigned int size)
- float getHeight (unsigned int pos)
- float getMomentum (unsigned int pos)
- · float getBathymetry (unsigned int pos)
- T getCellSize ()

```
Member Function Documentation
7.12.1
7.12.1.1 float scenarios::DamBreak::getBathymetry ( unsigned int pos ) [inline]
Returns
     floor elevation at pos
7.12.1.2 T scenarios::DamBreak::getCellSize() [inline]
Returns
     Cell size of one cell (= domain size/number of cells)
7.12.1.3 T scenarios::DamBreak::getCellSize() [inline]
```

Returns

Cell size of one cell (= domain size/number of cells)

7.12.1.4 float scenarios::DamBreak::getHeight (unsigned int pos ) [inline]

Returns

Initial water height at pos

7.12.1.5 float scenarios::DamBreak::getHeight ( unsigned int pos ) [inline]

Returns

Initial water height at pos

7.12.1.6 float scenarios::DamBreak::getMomentum (unsigned int pos) [inline]

Returns

Initial momentum at pos

7.12.1.7 float scenarios::DamBreak::getMomentum (unsigned int pos ) [inline]

Returns

Initial momentum at pos

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

- /home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/scenarios/dambreak.h
- /home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/scenarios/dambreak\_bathy.h

# 7.13 scenarios::Eisbach Class Reference

**Public Member Functions** 

- Eisbach (unsigned int size)
- float getHeight (unsigned int pos)
- float getMomentum (unsigned int pos)
- float getBathymetry (unsigned int pos)
- T getCellSize ()

# 7.13.1 Member Function Documentation

```
7.13.1.1 float scenarios::Eisbach::getBathymetry ( unsigned int pos ) [inline]
```

Returns

floor elevation at pos

```
7.13.1.2 T scenarios::Eisbach::getCellSize() [inline]
```

Returns

Cell size of one cell (= domain size/number of cells)

```
7.13.1.3 float scenarios::Eisbach::getHeight ( unsigned int pos ) [inline]
```

Returns

Initial water height at pos

**7.13.1.4** float scenarios::Eisbach::getMomentum ( unsigned int pos ) [inline]

Returns

Initial momentum at pos

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

/home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/scenarios/eisbach.h

# 7.14 scenarios::RareRare Class Reference

# **Public Member Functions**

- RareRare (unsigned int size)
- unsigned int getHeight (unsigned int pos)
- signed int getMomentum (unsigned int pos)
- T getCellSize ()

# 7.14.1 Member Function Documentation

```
7.14.1.1 T scenarios::RareRare::getCellSize( ) [inline]
```

Returns

Cell size of one cell (= domain size/number of cells)

7.14.1.2 unsigned int scenarios::RareRare::getHeight (unsigned int pos ) [inline]

Returns

Initial water height at pos

7.14.1.3 signed int scenarios::RareRare::getMomentum (unsigned int pos) [inline]

Returns

Initial momentum at pos

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

• /home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/scenarios/rarerare.h

# 7.15 scenarios::ShockShock Class Reference

# **Public Member Functions**

- ShockShock (unsigned int size)
- unsigned int getHeight (unsigned int pos)
- int getMomentum (unsigned int pos)
- T getCellSize ()

### 7.15.1 Member Function Documentation

7.15.1.1 T scenarios::ShockShock::getCellSize() [inline]

Returns

Cell size of one cell (= domain size/number of cells)

```
7.15.1.2 unsigned int scenarios::ShockShock::getHeight ( unsigned int pos ) [inline]
Returns
Initial water height at pos
```

7.15.1.3 int scenarios::ShockShock::getMomentum ( unsigned int pos ) [inline]

Returns

Initial momentum at pos

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

/home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/scenarios/ShockShock.h

# 7.16 scenarios::Subcrit Class Reference

**Public Member Functions** 

- Subcrit (unsigned int size)
- float getHeight (unsigned int pos)
- float getMomentum (unsigned int pos)
- float getBathymetry (unsigned int pos)
- T getCellSize ()

# 7.16.1 Member Function Documentation

```
7.16.1.1 float scenarios::Subcrit::getBathymetry ( unsigned int pos ) [inline]
```

Returns

floor elevation at pos

```
7.16.1.2 T scenarios::Subcrit::getCellSize( ) [inline]
```

Returns

Cell size of one cell (= domain size/number of cells)

```
7.16.1.3 float scenarios::Subcrit::getHeight ( unsigned int pos ) [inline]
```

Returns

Initial water height at pos

7.16.1.4 float scenarios::Subcrit::getMomentum ( unsigned int pos ) [inline]

Returns

Initial momentum at pos

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

/home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/scenarios/Subcritical\_flow.h

# 7.17 scenarios::Supercrit Class Reference

# **Public Member Functions**

- Supercrit (unsigned int size)
- float getHeight (unsigned int pos)
- float getMomentum (unsigned int pos)
- · float getBathymetry (unsigned int pos)
- T getCellSize ()

# 7.17.1 Member Function Documentation

```
7.17.1.1 float scenarios::Supercrit::getBathymetry ( unsigned int pos ) [inline]
```

Returns

floor elevation at pos

```
7.17.1.2 T scenarios::Supercrit::getCellSize() [inline]
```

Returns

Cell size of one cell (= domain size/number of cells)

```
7.17.1.3 float scenarios::Supercrit::getHeight ( unsigned int pos ) [inline]
```

Returns

Initial water height at pos

7.17.1.4 float scenarios::Supercrit::getMomentum (unsigned int pos ) [inline]

Returns

Initial momentum at pos

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

/home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/scenarios/Supercritical\_flow.h

# 7.18 Shader Class Reference

# **Public Member Functions**

- Shader (char const \*vertexShaderFile, char const \*fragmentShaderFile)
- ∼Shader ()
- bool shadersLoaded ()
- · void enableShader ()
- · void disableShader ()
- GLint getUniformLocation (const char \*name)
- · void setUniform (GLint location, GLfloat value)

# 7.18.1 Constructor & Destructor Documentation

7.18.1.1 Shader::Shader ( char const \* vertexShaderFile, char const \* fragmentShaderFile )

Constructor. Check whether shaders are supported. If yes, load vertex and fragment shader from textfile into memory and compile

#### **Parameters**

vertexShaderFile	name of the text file containing the vertex shader code
fragmentShader-	name of the text file containing the fragment shader code
File	

7.18.1.2 Shader:: ∼Shader ( )

Destructor. Unload shaders and free resources.

# 7.18.2 Member Function Documentation

7.18.2.1 void Shader::disableShader ( )

Restores OpenGL default shaders

7.18.2.2 void Shader::enableShader ( )

Replaces OpenGL shaders by our custom shaders

7.18.2.3 GLint Shader::getUniformLocation ( const char \* name ) [inline]

Returns

Location of the uniform variable

7.18.2.4 void Shader::setUniform ( GLint location, GLfloat value ) [inline]

Set a uniform variable in the shader

7.18.2.5 bool Shader::shadersLoaded ( )

Returns, whether shaders could by loaded successfully

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

- /home/raphael/Programmieren/BPraktikum/SWE/src/opengl/shader.h
- /home/raphael/Programmieren/BPraktikum/SWE/src/opengl/shader.cpp

# 7.19 Simulation Class Reference

# **Public Member Functions**

- Simulation ()
- ∼Simulation ()
- void restart ()
- void loadNewScenario (SWE\_Scenario \*scene)
- void resize (float factor)
- void setBathBuffer (float \*output)
- void runCuda (struct cudaGraphicsResource \*\*vbo\_resource, struct cudaGraphicsResource \*\*vbo\_-normals)

- int getNx ()
- int getNy ()
- const Float2D & getBathymetry ()
- void getScalingApproximation (float &bScale, float &bOffset, float &wScale)

# 7.19.1 Constructor & Destructor Documentation

```
7.19.1.1 Simulation::Simulation ( )
```

Constructor. Initializes SWE BlockCUDA and creates a new instance of it.

7.19.1.2 Simulation::~Simulation()

Destructor.

# 7.19.2 Member Function Documentation

7.19.2.1 void Simulation::getScalingApproximation ( float & bScale, float & bOffset, float & wScale )

Computes a first approximation of the scaling values needed for visualization. Gets called before simulation starts and determines the average, mininimum and maximum values of the bathymetry and water surface data. Uses latter values to estimate the scaling factors.

7.19.2.2 void Simulation::restart ( )

Restarts the simulation. Restores the initial bondaries.

7.19.2.3 void Simulation::runCuda ( struct cudaGraphicsResource \*\* vbo\_resource, struct cudaGraphicsResource \*\* vbo\_normals )

This is the main simulation procedure. Simulates one timestep and computes normals afterwards.

#### **Parameters**

vbo_resource	cuda resource holding the vertex positions
vbo_normals	cuda resource holding the normals

7.19.2.4 void Simulation::setBathBuffer ( float \* bath )

Sets the bathymetry buffer. Buffer contains vertex position and vertex normal in sequence.

**Parameters** 

bath	float array in which computed values will be stored

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

- /home/raphael/Programmieren/BPraktikum/SWE/src/opengl/simulation.h
- · /home/raphael/Programmieren/BPraktikum/SWE/src/opengl/simulation.cu

# 7.20 solver::FWave < T > Class Template Reference

#include <FWave.hpp>

# **Public Member Functions**

void computeNetUpdates (const T &hLeft, const T &hRight, const T &huLeft, const T &huRight, const T &bathRight, T &hNetUpdatesLeft, T &hNetUpdatesRight, T &huNetUpdatesLeft, T &huNetUpdatesRight, T &maxEdgeSpeed)

# 7.20.1 Detailed Description

template < class T > class solver::FWave < T >

This class will approximately solve the initial value problem for the **shallow water equations** over time:

# 7.20.2 Member Function Documentation

7.20.2.1 template < class T > void FWave::computeNetUpdates ( const T & hLeft, const T & hRight, const T & huLeft, const T & huRight, const T & bathLeft, const T & bathRight, T & hNetUpdatesLeft, T & hNetUpdatesRight, T & huNetUpdatesLeft, T & huNetUpdatesRight, T & maxEdgeSpeed )

calculate the net-updates for a simulation of the flow of water

This implementation will calculate the net-updates for a simulation of flow of water using the height of the water column and its momentum as parameters.

#### **Parameters**

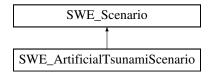
hLeft	water column height on the left side
hRight	water column height on the right side
huLeft	momentum of the water on the left side
huRight	momentum of the water on the right side
bathLeft	elevation of the ocean floor on the left side
bathRight	elevation of the ocean floor on the right side
hNetUpdatesLeft	reference to the variable that will store the update to the height of the water column on the
	left
hNetUpdates-	reference to the variable that will store the update to the height of the water column on the
Right	right
huNetUpdates-	reference to the variable that will store the update to the momentum of the water on the left
Left	
huNetUpdates-	reference to the variable that will store the update to the momentum of the water on the right
Right	
maxEdgeSpeed	reference to the variable that will store the maximum edge-speed

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

- /home/raphael/Programmieren/BPraktikum/SWE/submodules/f-wave-solver/src/FWave.hpp
- /home/raphael/Programmieren/BPraktikum/SWE/submodules/f-wave-solver/src/FWave.cpp

# 7.21 SWE\_ArtificialTsunamiScenario Class Reference

Inheritance diagram for SWE\_ArtificialTsunamiScenario:



# **Public Member Functions**

- float getWaterHeight (float x, float y)
- float **getBathymetry** (float x, float y)
- BoundaryType getBoundaryType (BoundaryEdge edge)
- float getBoundaryPos (BoundaryEdge i\_edge)
- float endSimulation ()

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

 $\bullet \ \ / home/raphael/Programmieren/BPraktikum/SWE/src/scenarios/SWE\_ArtificialTsunamiScenario.hh$ 

# 7.22 SWE\_AsagiGrid Class Reference

# **Public Member Functions**

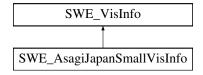
- · void open (const std::string &i filename)
- · void close ()
- · asagi::Grid & grid ()

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

/home/raphael/Programmieren/BPraktikum/SWE/src/scenarios/SWE\_AsagiScenario.hh

# 7.23 SWE\_AsagiJapanSmallVisInfo Class Reference

Inheritance diagram for SWE\_AsagiJapanSmallVisInfo:



# **Public Member Functions**

- · virtual float waterVerticalScaling ()
- · virtual float bathyVerticalScaling ()

# 7.23.1 Member Function Documentation

**7.23.1.1 virtual float SWE\_AsagiJapanSmallVisInfo::bathyVerticalScaling()** [inline], [virtual]

Returns

The vertical scaling factor for the bathymetry

Reimplemented from SWE\_VisInfo.

**7.23.1.2** virtual float SWE\_AsagiJapanSmallVisInfo::waterVerticalScaling( ) [inline], [virtual]

Returns

The vertical scaling factor of the water

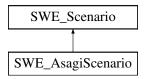
Reimplemented from SWE\_VisInfo.

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

/home/raphael/Programmieren/BPraktikum/SWE/src/scenarios/SWE AsagiScenario vis.hh

# 7.24 SWE\_AsagiScenario Class Reference

Inheritance diagram for SWE\_AsagiScenario:



# **Public Member Functions**

- SWE\_AsagiScenario (const std::string i\_bathymetryFile, const std::string i\_displacementFile, const float i\_duration, const float i\_simulationArea[4], const bool i\_dynamicDisplacement=false)
- · void deleteGrids ()
- float getWaterHeight (float i\_positionX, float i\_positionY)
- float getBathymetry (const float i\_positionX, const float i\_positionY)
- float getBathymetryAndDynamicDisplacement (const float i\_positionX, const float i\_positionY, const float i\_time)
- bool dynamicDisplacementAvailable (const float i\_time)
- float endSimulation ()
- BoundaryType getBoundaryType (BoundaryEdge i edge)
- float getBoundaryPos (BoundaryEdge i\_edge)

# 7.24.1 Constructor & Destructor Documentation

7.24.1.1 SWE\_AsagiScenario::SWE\_AsagiScenario ( const std::string *i\_bathymetryFile*, const std::string *i\_displacementFile*, const float *i\_duration*, const float *i\_simulationArea[4]*, const bool *i\_dynamicDisplacement* = false ) [inline]

Constructor of an Asagi Scenario, which initializes the corresponding Asagi grids.

**Parameters** 

i_originX	origin of the simulation area (x-direction)
i_originY	origin of the simulation area (y-direction)
i_bathymetryFile	path to the netCDF-bathymetry file
i_displacement-	path to the netCDF-displacement file
File	
i_duration	time the simulation runs (in seconds)

# 7.24.2 Member Function Documentation

7.24.2.1 bool SWE\_AsagiScenario::dynamicDisplacementAvailable ( const float i\_time ) [inline]

Check if there is an dynamic displacement is available for the corresponding time.

#### **Parameters**

i_time	current simulation time

#### Returns

true if there is data available, false else

**7.24.2.2** float SWE\_AsagiScenario::endSimulation() [inline], [virtual]

Get the number of seconds, the simulation should run.

#### Returns

number of seconds, the simulation should run

Reimplemented from SWE\_Scenario.

7.24.2.3 float SWE\_AsagiScenario::getBathymetry ( const float i\_positionX, const float i\_positionY ) [inline], [virtual]

Get the bathymetry including static displacement at a specific location

# **Parameters**

i_positionX	position relative to the origin of the displacement grid in x-direction
i_positionY	position relative to the origin of the displacement grid in y-direction

# Returns

bathymetry (after the initial displacement (static displacement)

Reimplemented from SWE Scenario.

7.24.2.4 float SWE\_AsagiScenario::getBathymetryAndDynamicDisplacement ( const float *i\_positionX*, const float *i\_positionX*, const float *i\_time* ) [inline]

Get the bathymetry including dynamic displacement at a specific location

### **Parameters**

	i_positionX	position relative to the origin of the displacement grid in x-direction
ĺ	i_positionY	position relative to the origin of the displacement grid in y-direction
ĺ	i_time	time relative to the origin of the dynamic displacement

# Returns

bathymetry (after the initial displacement (static displacement), after the specified amount of time (dynamic displacement))

**7.24.2.5** float SWE\_AsagiScenario::getBoundaryPos ( BoundaryEdge i\_edge ) [inline], [virtual]

Get the boundary positions

#### **Parameters**

i_edge	which edge

#### Returns

value in the corresponding dimension

Reimplemented from SWE\_Scenario.

**7.24.2.6 BoundaryType SWE\_AsagiScenario::getBoundaryType ( BoundaryEdge** *i\_edge* **)** [inline], [virtual]

Get the boundary types of the simulation

### **Parameters**

edge	specific edge

### Returns

type of the edge

Reimplemented from SWE\_Scenario.

7.24.2.7 float SWE\_AsagiScenario::getWaterHeight (float i\_positionX, float i\_positionY) [inline], [virtual]

Get the water height at a specific location (before the initial displacement).

# **Parameters**

i_positionX	position relative to the origin of the bathymetry grid in x-direction
i_positionY	position relative to the origin of the bathymetry grid in y-direction

#### Returns

water height (before the initial displacement)

Reimplemented from SWE\_Scenario.

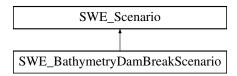
The documentation for this class was generated from the following files:

- /home/raphael/Programmieren/BPraktikum/SWE/src/scenarios/SWE\_AsagiScenario.hh
- /home/raphael/Programmieren/BPraktikum/SWE/src/scenarios/SWE\_AsagiScenario.cpp

# 7.25 SWE\_BathymetryDamBreakScenario Class Reference

#include <SWE\_simple\_scenarios.hh>

Inheritance diagram for SWE\_BathymetryDamBreakScenario:



#### **Public Member Functions**

- float getBathymetry (float x, float y)
- virtual float endSimulation ()
- virtual BoundaryType getBoundaryType (BoundaryEdge edge)
- float getBoundaryPos (BoundaryEdge i\_edge)
- float getWaterHeight (float i\_positionX, float i\_positionY)

# 7.25.1 Detailed Description

Scenario "Bathymetry Dam Break": uniform water depth, but elevated bathymetry in the centre of the domain

# 7.25.2 Member Function Documentation

**7.25.2.1** float SWE\_BathymetryDamBreakScenario::getBoundaryPos ( BoundaryEdge *i\_edge* ) [inline], [virtual]

Get the boundary positions

**Parameters** 

i_edge	which edge

### Returns

value in the corresponding dimension

Reimplemented from SWE\_Scenario.

**7.25.2.2** float SWE\_BathymetryDamBreakScenario::getWaterHeight ( float i\_positionX, float i\_positionY ) [inline], [virtual]

Get the water height at a specific location.

# **Parameters**

i_positionX	position relative to the origin of the bathymetry grid in x-direction
i_positionY	position relative to the origin of the bathymetry grid in y-direction

Returns

water height (before the initial displacement)

Reimplemented from SWE\_Scenario.

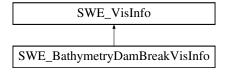
The documentation for this class was generated from the following file:

• /home/raphael/Programmieren/BPraktikum/SWE/src/scenarios/SWE\_simple\_scenarios.hh

# 7.26 SWE\_BathymetryDamBreakVisInfo Class Reference

```
#include <SWE_simple_scenarios_vis.hh>
```

Inheritance diagram for SWE\_BathymetryDamBreakVisInfo:



# **Public Member Functions**

· float bathyVerticalOffset ()

# 7.26.1 Detailed Description

VisInfo "Bathymetry Dam Break": uniform water depth, but elevated bathymetry in the center of the domain Set bathymetry offset hence it is visible in the screen

# 7.26.2 Member Function Documentation

7.26.2.1 float SWE\_BathymetryDamBreakVisInfo::bathyVerticalOffset() [inline], [virtual]

Returns

The vertical offset for the bathymetry. Should be 0 for "real" scenarios (scenarios with dry areas)

Reimplemented from SWE\_VisInfo.

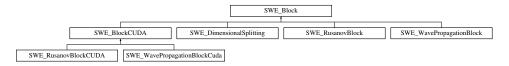
The documentation for this class was generated from the following file:

/home/raphael/Programmieren/BPraktikum/SWE/src/scenarios/SWE\_simple\_scenarios\_vis.hh

# 7.27 SWE\_Block Class Reference

```
#include <SWE_Block.hh>
```

Inheritance diagram for SWE\_Block:



# **Public Member Functions**

```
    void initScenario (float_offsetX, float_offsetY, SWE_Scenario &i_scenario, const bool i_multipleBlocks=false)
    initialise unknowns to a specific scenario:
```

void setWaterHeight (float(\*\_h)(float, float))

set the water height according to a given function

void setDischarge (float(\*\_u)(float, float), float(\*\_v)(float, float))

set the momentum/discharge according to the provided functions

void setBathymetry (float \_b)

set the bathymetry to a uniform value

void setBathymetry (float(\*\_b)(float, float))

set the bathymetry according to a given function

const Float2D & getWaterHeight ()

provides read access to the water height array

const Float2D & getDischarge hu ()

provides read access to the momentum/discharge array (x-component)

const Float2D & getDischarge hv ()

provides read access to the momentum/discharge array (y-component)

const Float2D & getBathymetry ()

provides read access to the bathymetry data

- void setBoundaryType (BoundaryEdge edge, BoundaryType boundtype, const SWE\_Block1D \*inflow=NULL)
   set type of boundary condition for the specified boundary
- virtual SWE\_Block1D \* registerCopyLayer (BoundaryEdge edge)

return a pointer to proxy class to access the copy layer

virtual SWE\_Block1D \* grabGhostLayer (BoundaryEdge edge)

"grab" the ghost layer in order to set these values externally

void setGhostLayer ()

set values in ghost layers

float getMaxTimestep ()

return maximum size of the time step to ensure stability of the method

- void computeMaxTimestep (const float i dryTol=0.1, const float i cflNumber=0.4)
- virtual void simulateTimestep (float dt)=0

execute a single time step of the simulation

- virtual float simulate (float tStart, float tEnd)=0
- virtual void computeNumericalFluxes ()=0

compute the numerical fluxes for each edge of the Cartesian grid

virtual void updateUnknowns (float dt)=0

compute the new values of the unknowns h, hu, and hv in all grid cells

• int getNx ()

returns nx, i.e. the grid size in x-direction

• int getNy ()

returns ny, i.e. the grid size in y-direction

# **Static Public Attributes**

• static const float g = 9.81fstatic variable that holds the gravity constant  $(g = 9.81 \text{ m/s}^{\land} 2)$ :

# **Protected Member Functions**

- SWE\_Block (int I\_nx, int I\_ny, float I\_dx, float I\_dy)
- virtual ∼SWE Block ()
- · void setBoundaryBathymetry ()
- virtual void synchAfterWrite ()
- virtual void synchWaterHeightAfterWrite ()
- virtual void synchDischargeAfterWrite ()
- virtual void synchBathymetryAfterWrite ()
- virtual void synchGhostLayerAfterWrite ()
- virtual void synchBeforeRead ()
- virtual void synchWaterHeightBeforeRead ()
- virtual void synchDischargeBeforeRead ()
- virtual void synchBathymetryBeforeRead ()
- virtual void synchCopyLayerBeforeRead ()
- virtual void setBoundaryConditions ()

set boundary conditions in ghost layers (set boundary conditions)

# **Protected Attributes**

int nx

size of Cartesian arrays in x-direction

int ny

size of Cartesian arrays in y-direction

float dx

mesh size of the Cartesian grid in x-direction

float dy

mesh size of the Cartesian grid in y-direction

Float2D h

array that holds the water height for each element

· Float2D hu

array that holds the x-component of the momentum for each element (water height h multiplied by velocity in x-direction)

Float2D hv

array that holds the y-component of the momentum for each element (water height h multiplied by velocity in y-direction)

Float2D b

array that holds the bathymetry data (sea floor elevation) for each element

• BoundaryType boundary [4]

type of boundary conditions at LEFT, RIGHT, TOP, and BOTTOM boundary

const SWE\_Block1D \* neighbour [4]

for CONNECT boundaries: pointer to connected neighbour block

float maxTimestep

maximum time step allowed to ensure stability of the method

· float offsetX

x-coordinate of the origin (left-bottom corner) of the Cartesian grid

· float offsetY

y-coordinate of the origin (left-bottom corner) of the Cartesian grid

# 7.27.1 Detailed Description

SWE\_Block is the main data structure to compute our shallow water model on a single Cartesian grid block: SWE\_Block is an abstract class (and interface) that should be extended by respective implementation classes.

**Cartesian Grid for Discretization:** 

SWE\_Blocks uses a regular Cartesian grid of size nx by ny, where each grid cell carries three unknowns:

- · the water level h
- the momentum components hu and hv (in x- and y- direction, resp.)
- · the bathymetry b

Each of the components is stored as a 2D array, implemented as a Float2D object, and are defined on grid indices [0,...,nx+1]\*[0,...,ny+1]. The computational domain is indexed with [1,...,nx]\*[1,...,ny].

The mesh sizes of the grid in x- and y-direction are stored in static variables dx and dy. The position of the Cartesian grid in space is stored via the coordinates of the left-bottom corner of the grid, in the variables offsetX and offsetY.

#### **Ghost layers:**

To implement the behaviour of the fluid at boundaries and for using multiple block in serial and parallel settings, SWE\_Block adds an additional layer of so-called ghost cells to the Cartesian grid, as illustrated in the following figure. Cells in the ghost layer have indices 0 or nx+1 / ny+1.

#### Memory Model:

The variables h, hu, hv for water height and momentum will typically be updated by classes derived from SWE\_Block. However, it is not assumed that such and updated will be performed in every time step. Instead, subclasses are welcome to update h, hu, and hv in a lazy fashion, and keep data in faster memory (incl. local memory of acceleration hardware, such as GPGPUs), instead.

It is assumed that the bathymetry data **b** is not changed during the algorithm (up to the exceptions mentioned in the following).

To force a synchronization of the respective data structures, the following methods are provided as part of SWE\_-Block:

- synchAfterWrite() to synchronize h, hu, hv, and b after an external update (reading a file, e.g.);
- synchWaterHeightAfterWrite(), synchDischargeAfterWrite(), synchBathymetryAfterWrite(): to synchronize only h or momentum (hu and hv) or bathymetry b;
- synchGhostLayerAfterWrite() to synchronize only the ghost layers
- synchBeforeRead() to synchronize h, hu, hv, and b before an output of the variables (writing a visualization file, e.g.)
- synchWaterHeightBeforeRead(), synchDischargeBeforeRead(), synchBathymetryBeforeRead(): as synch-BeforeRead(), but only for the specified variables
- synchCopyLayerBeforeRead(): synchronizes the copy layer only (i.e., a layer that is to be replicated in a neighbouring SWE\_Block.

# **Derived Classes**

As SWE\_Block just provides an abstract base class together with the most important data structures, the implementation of concrete models is the job of respective derived classes (see the class diagram at the top of this page). Similar, parallel implementations that are based on a specific parallel programming model (such as OpenMP) or parallel architecture (such as GPU/CUDA) should form subclasses of their own. Please refer to the documentation of these classes for more details on the model and on the parallelisation approach.

#### 7.27.2 Constructor & Destructor Documentation

```
7.27.2.1 SWE_Block::SWE_Block(int I_nx, int I_ny, float I_dx, float I_dy) [protected]
```

Constructor: allocate variables for simulation

unknowns h (water height), hu,hv (discharge in x- and y-direction), and b (bathymetry) are defined on grid indices [0,...,nx+1]\*[0,...,ny+1] -> computational domain is [1,...,nx]\*[1,...,ny] -> plus ghost cell layer

The constructor is protected: no instances of SWE\_Block can be generated.

```
7.27.2.2 SWE_Block::~SWE_Block() [protected], [virtual]
```

Destructor: de-allocate all variables

#### 7.27.3 Member Function Documentation

```
7.27.3.1 void SWE_Block::computeMaxTimestep ( const float i\_dryTol = 0.1, const float i\_cflNumber = 0.4)
```

Compute the largest allowed time step for the current grid block (reference implementation) depending on the current values of variables h, hu, and hv, and store this time step size in member variable maxTimestep.

#### **Parameters**

i_dryTol	dry tolerance (dry cells do not affect the time step).
i_cflNumber	CFL number of the used method.

#### 7.27.3.2 virtual void SWE\_Block::computeNumericalFluxes() [pure virtual]

compute the numerical fluxes for each edge of the Cartesian grid

The computation of fluxes strongly depends on the chosen numerical method. Hence, this purely virtual function has to be implemented in the respective derived classes.

Implemented in SWE\_WavePropagationBlock, SWE\_WavePropagationBlockCuda, SWE\_RusanovBlock, SWE\_RusanovBlockCuDA, and SWE\_DimensionalSplitting.

```
7.27.3.3 const Float2D & SWE_Block::getBathymetry ( )
```

provides read access to the bathymetry data return reference to bathymetry unknown b

```
7.27.3.4 const Float2D & SWE_Block::getDischarge_hu ( )
```

provides read access to the momentum/discharge array (x-component) return reference to discharge unknown hu

```
7.27.3.5 const Float2D & SWE_Block::getDischarge_hv ( )
```

provides read access to the momentum/discharge array (y-component)

7.27.3.6 float SWE\_Block::getMaxTimestep() [inline]

return reference to discharge unknown hv

return maximum size of the time step to ensure stability of the method

#### Returns

current value of the member variable maxTimestep

7.27.3.7 const Float2D & SWE\_Block::getWaterHeight ( )

provides read access to the water height array

Restores values for h, v, and u from file data

**Parameters** 

\_b | array holding b-values in sequence return reference to water height unknown h

7.27.3.8 SWE Block1D \* SWE\_Block::grabGhostLayer(BoundaryEdge edge) [virtual]

"grab" the ghost layer in order to set these values externally

"grab" the ghost layer at the specific boundary in order to set boundary values in this ghost layer externally. The boundary conditions at the respective ghost layer is set to PASSIVE, such that the grabbing program component is responsible to provide correct values in the ghost layer, for example by receiving data from a remote copy layer via MPI communication.

#### **Parameters**

appointed	odgo
speciliea	eage

#### Returns

a SWE\_Block1D object that contains row variables h, hu, and hv

Reimplemented in SWE BlockCUDA.

7.27.3.9 void SWE\_Block::initScenario ( float \_offsetX, float \_offsetY, SWE\_Scenario & i\_scenario, const bool i\_multipleBlocks = false )

initialise unknowns to a specific scenario:

Initializes the unknowns and bathymetry in all grid cells according to the given SWE\_Scenario.

In the case of multiple SWE\_Blocks at this point, it is not clear how the boundary conditions should be set. This is because an isolated SWE\_Block doesn't have any in information about the grid. Therefore the calling routine, which has the information about multiple blocks, has to take care about setting the right boundary conditions.

#### **Parameters**

i_scenario	scenario, which is used during the setup.
i_multipleBlocks	are the multiple SWE_blocks?

7.27.3.10 SWE\_Block1D \* SWE\_Block::registerCopyLayer( BoundaryEdge edge ) [virtual]

return a pointer to proxy class to access the copy layer

register the row or column layer next to a boundary as a "copy layer", from which values will be copied into the ghost layer or a neighbour;

# Returns

a SWE Block1D object that contains row variables h, hu, and hv

Reimplemented in SWE\_BlockCUDA.

7.27.3.11 void SWE\_Block::setBathymetry ( float \_b )

set the bathymetry to a uniform value

set Bathymetry b in all grid cells (incl. ghost/boundary layers) to a uniform value bathymetry source terms are re-computed

```
7.27.3.12 void SWE_Block::setBathymetry ( float(*)(float, float) _b )
```

set the bathymetry according to a given function

set Bathymetry b in all grid cells (incl. ghost/boundary layers) using the specified bathymetry function; bathymetry source terms are re-computed

```
7.27.3.13 void SWE_Block::setBoundaryBathymetry() [protected]
```

Sets the bathymetry on OUTFLOW or WALL boundaries. Should be called very time a boundary is changed to a OUTFLOW or WALL boundary **or** the bathymetry changes.

```
7.27.3.14 void SWE_Block::setBoundaryConditions() [protected], [virtual]
```

set boundary conditions in ghost layers (set boundary conditions)

set the values of all ghost cells depending on the specifed boundary conditions

- set boundary conditions for typs WALL and OUTFLOW
- · derived classes need to transfer ghost layers

Reimplemented in SWE\_BlockCUDA.

```
7.27.3.15 void SWE_Block::setBoundaryType ( BoundaryEdge edge, BoundaryType boundtype, const SWE_Block1D *i\_inflow = NULL )
```

set type of boundary condition for the specified boundary

Set the boundary type for specific block boundary.

#### **Parameters**

i_edge	location of the edge relative to the SWE_block.
i_boundaryType	type of the boundary condition.
i_inflow	pointer to an SWE_Block1D, which specifies the inflow (should be NULL for WALL or OUTF-
	LOW boundary)

7.27.3.16 void SWE\_Block::setDischarge ( float(\*)(float, float) \_u, float(\*)(float, float) \_v )

set the momentum/discharge according to the provided functions

set discharge in all interior grid cells (i.e. except ghost layer) to values specified by parameter functions Note: unknowns hu and hv represent momentum, while parameters u and v are velocities!

```
7.27.3.17 void SWE_Block::setGhostLayer ( )
```

set values in ghost layers

set the values of all ghost cells depending on the specifed boundary conditions; if the ghost layer replicates the variables of a remote SWE\_Block, the values are copied

7.27.3.18 void SWE\_Block::setWaterHeight ( float(\*)(float, float) \_h )

set the water height according to a given function

set water height h in all interior grid cells (i.e. except ghost layer) to values specified by parameter function \_h

```
7.27.3.19 virtual float SWE_Block::simulate (float tStart, float tEnd) [pure virtual]
```

perform the simulation starting with simulation time tStart, until simulation time tEnd is reached simulate implements the main simulation loop between two checkpoints; note that this function can typically only be used, if you only use a single SWE\_Block; in particular, simulate can not trigger calls to exchange values of copy and ghost layers between blocks!

#### **Parameters**

tStart	time where the simulation is started
tEnd	time of the next checkpoint

#### Returns

actual end time reached

Implemented in SWE\_WavePropagationBlock, SWE\_WavePropagationBlockCuda, SWE\_RusanovBlockCUDA, S-WE\_RusanovBlock, and SWE\_DimensionalSplitting.

```
7.27.3.20 void SWE_Block::synchAfterWrite() [protected], [virtual]
```

Update all temporary and non-local (for heterogeneous computing) variables after an external update of the main variables h, hu, hv, and b.

Reimplemented in SWE\_BlockCUDA.

```
7.27.3.21 void SWE_Block::synchBathymetryAfterWrite( ) [protected], [virtual]
```

Update temporary and non-local (for heterogeneous computing) variables after an external update of the bathymetry b

Reimplemented in SWE\_BlockCUDA.

```
7.27.3.22 void SWE_Block::synchBathymetryBeforeRead() [protected], [virtual]
```

Update temporary and non-local (for heterogeneous computing) variables before an external access to the bathymetry b

Reimplemented in SWE\_BlockCUDA.

```
7.27.3.23 void SWE_Block::synchBeforeRead( ) [protected], [virtual]
```

Update all temporary and non-local (for heterogeneous computing) variables before an external access to the main variables h, hu, hv, and b.

Reimplemented in SWE BlockCUDA.

```
7.27.3.24 void SWE_Block::synchCopyLayerBeforeRead( ) [protected], [virtual]
```

Update (for heterogeneous computing) variables in copy layers before an external access to the unknowns Reimplemented in SWE\_BlockCUDA.

```
7.27.3.25 void SWE_Block::synchDischargeAfterWrite( ) [protected], [virtual]
```

Update temporary and non-local (for heterogeneous computing) variables after an external update of the discharge variables hu and hv

Reimplemented in SWE\_BlockCUDA.

```
7.27.3.26 void SWE_Block::synchDischargeBeforeRead() [protected], [virtual]
```

Update temporary and non-local (for heterogeneous computing) variables before an external access to the discharge variables hu and hv

Reimplemented in SWE\_BlockCUDA.

```
7.27.3.27 void SWE_Block::synchGhostLayerAfterWrite( ) [protected], [virtual]
```

Update the ghost layers (only for CONNECT and PASSIVE boundary conditions) after an external update of the main variables h, hu, hv, and b in the ghost layer.

Reimplemented in SWE\_BlockCUDA.

```
7.27.3.28 void SWE_Block::synchWaterHeightAfterWrite( ) [protected], [virtual]
```

Update temporary and non-local (for heterogeneous computing) variables after an external update of the water height h

Reimplemented in SWE BlockCUDA.

```
7.27.3.29 void SWE_Block::synchWaterHeightBeforeRead() [protected], [virtual]
```

Update temporary and non-local (for heterogeneous computing) variables before an external access to the water height h

Reimplemented in SWE\_BlockCUDA.

```
7.27.3.30 virtual void SWE_Block::updateUnknowns (float dt ) [pure virtual]
```

compute the new values of the unknowns h, hu, and hv in all grid cells

based on the numerical fluxes (computed by computeNumericalFluxes) and the specified time step size dt, an Euler time step is executed. As the computational fluxes will depend on the numerical method, this purely virtual function has to be implemented separately for each specific numerical model (and parallelisation approach).

**Parameters** 

```
dt size of the time step
```

Implemented in SWE\_WavePropagationBlock, SWE\_WavePropagationBlockCuda, SWE\_RusanovBlock, SWE\_RusanovBlockCuDA, and SWE\_DimensionalSplitting.

#### 7.27.4 Member Data Documentation

```
7.27.4.1 float SWE_Block::maxTimestep [protected]
```

maximum time step allowed to ensure stability of the method

maxTimestep can be updated as part of the methods computeNumericalFluxes and updateUnknowns (depending on the numerical method)

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

- /home/raphael/Programmieren/BPraktikum/SWE/src/blocks/SWE Block.hh
- /home/raphael/Programmieren/BPraktikum/SWE/src/blocks/SWE\_Block.cpp

# 7.28 SWE\_Block1D Struct Reference

```
#include <SWE_Block.hh>
```

## **Public Member Functions**

- SWE\_Block1D (const Float1D &\_h, const Float1D &\_hu, const Float1D &\_hv)
- SWE Block1D (float \* h, float \* hu, float \* hv, int size, int stride=1)

## **Public Attributes**

- Float1D h
- Float1D hu
- Float1D hv

## 7.28.1 Detailed Description

SWE\_Block1D is a simple struct that can represent a single line or row of SWE\_Block unknowns (using the Float1D proxy class). It is intended to unify the implementation of inflow and periodic boundary conditions, as well as the ghost/copy-layer connection between several SWE\_Block grids.

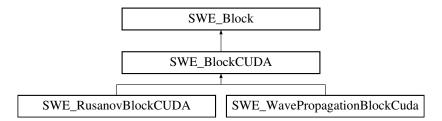
The documentation for this struct was generated from the following file:

• /home/raphael/Programmieren/BPraktikum/SWE/src/blocks/SWE\_Block.hh

# 7.29 SWE BlockCUDA Class Reference

```
#include <SWE BlockCUDA.hh>
```

Inheritance diagram for SWE BlockCUDA:



## **Public Member Functions**

- SWE\_BlockCUDA (int I\_nx, int I\_ny, float I\_dx, float I\_dy)
- virtual ~SWE BlockCUDA ()
- virtual SWE Block1D \* registerCopyLayer (BoundaryEdge edge)
- virtual SWE\_Block1D \* grabGhostLayer (BoundaryEdge edge)
- const float \* getCUDA waterHeight ()
- const float \* getCUDA\_bathymetry ()

## **Static Public Member Functions**

- static void printDeviceInformation ()
- static void init (int i\_cudaDevice=0)
- static void finalize ()

## **Protected Member Functions**

- virtual void synchAfterWrite ()
- · virtual void synchWaterHeightAfterWrite ()
- virtual void synchDischargeAfterWrite ()
- virtual void synchBathymetryAfterWrite ()
- virtual void synchGhostLayerAfterWrite ()
- virtual void synchBeforeRead ()
- virtual void synchWaterHeightBeforeRead ()
- virtual void synchDischargeBeforeRead ()
- · virtual void synchBathymetryBeforeRead ()
- virtual void synchCopyLayerBeforeRead ()
- virtual void setBoundaryConditions ()

## **Protected Attributes**

- float \* hd
- · float \* hud
- float \* hvd
- float \* **bd**

## **Additional Inherited Members**

## 7.29.1 Detailed Description

SWE\_BlockCUDA extends the base class SWE\_Block towards a base class for a CUDA implementation of the shallow water equations. It adds the respective variables in GPU memory, and provides methods for data transfer between main and GPU memory.

# 7.29.2 Constructor & Destructor Documentation

7.29.2.1 SWE\_BlockCUDA::SWE\_BlockCUDA ( int I\_nx, int I\_ny, float I\_dx, float I\_dy )

Constructor: allocate variables for simulation

unknowns h,hu,hv,b are defined on grid indices [0,...,nx+1]\*[0,...,ny+1] -> computational domain is [1,...,nx]\*[1,...,ny] -> plus ghost cell layer

flux terms are defined for edges with indices [0,...,nx]\*[1,...,ny] or [1,...,nx]\*0, ..., ny Flux term with index (i,j) is located on the edge between cells with index (i,j) and (i+1,j) or (i,j+1)

bathymetry source terms are defined for cells with indices [1,..,nx]\*[1,..,ny]

#### Parameters

	i cudaDevice	ID of the CUDA-device, which should be used.
--	--------------	--

```
7.29.2.2 SWE_BlockCUDA::~SWE_BlockCUDA( ) [virtual]
```

Destructor: de-allocate all variables

## 7.29.3 Member Function Documentation

```
7.29.3.1 void SWE_BlockCUDA::finalize( ) [static]
```

Cleans up the cuda device

```
7.29.3.2 const float* SWE_BlockCUDA::getCUDA_bathymetry() [inline]
```

Returns

pointer to the array #hb (bathymetry) in device memory

```
7.29.3.3 const float* SWE_BlockCUDA::getCUDA_waterHeight() [inline]
```

Returns

pointer to the array #hd (water height) in device memory

```
7.29.3.4 SWE Block1D * SWE_BlockCUDA::grabGhostLayer( BoundaryEdge edge) [virtual]
```

"grab" the ghost layer at the specific boundary in order to set boundary values in this ghost layer externally. The boundary conditions at the respective ghost layer is set to PASSIVE, such that the grabbing program component is responsible to provide correct values in the ghost layer, for example by receiving data from a remote copy layer via MPI communication.

**Parameters** 

```
specified edge
```

Returns

a SWE\_Block1D object that contains row variables h, hu, and hv

Reimplemented from SWE Block.

**7.29.3.5** void SWE\_BlockCUDA::init(int i\_cudaDevice = 0) [static]

Initializes the cuda device Has to be called once at the beginning.

**7.29.3.6 void SWE\_BlockCUDA::printDeviceInformation()** [static]

Print some available information about the CUDA devices. id of the CUDA device.

total number of CUDA devices on this host.

drive and runtime version

device properties

```
7.29.3.7 SWE_Block1D * SWE_BlockCUDA::registerCopyLayer( BoundaryEdge edge ) [virtual]
```

register the row or column layer next to a boundary as a "copy layer", from which values will be copied into the ghost layer or a neighbour;

Returns

```
a SWE Block1D object that contains row variables h, hu, and hv
```

Reimplemented from SWE Block.

```
7.29.3.8 void SWE BlockCUDA::setBoundaryConditions() [protected], [virtual]
```

set the values of all ghost cells depending on the specifed boundary conditions

Reimplemented from SWE Block.

```
7.29.3.9 void SWE_BlockCUDA::synchAfterWrite() [protected], [virtual]
```

Update all temporary and non-local (for heterogeneous computing) variables after an external update of the main variables h, hu, hv, and b.

Reimplemented from SWE Block.

```
7.29.3.10 void SWE_BlockCUDA::synchBathymetryAfterWrite() [protected], [virtual]
```

Update temporary and non-local (for heterogeneous computing) variables after an external update of the bathymetry b

Reimplemented from SWE Block.

```
7.29.3.11 void SWE_BlockCUDA::synchBathymetryBeforeRead() [protected], [virtual]
```

Update temporary and non-local (for heterogeneous computing) variables before an external access to the bathymetry b

Reimplemented from SWE Block.

```
7.29.3.12 void SWE_BlockCUDA::synchBeforeRead( ) [protected], [virtual]
```

Update the main variables h, hu, hv, and b before an external read access: copies current content of the respective device variables hd, hud, hvd, bd

Reimplemented from SWE\_Block.

```
7.29.3.13 void SWE_BlockCUDA::synchCopyLayerBeforeRead() [protected], [virtual]
```

Update (for heterogeneous computing) variables h, hu, hv in copy layers before an external access to the unknowns (only required for PASSIVE and CONNECT boundaries)

· copy (up-to-date) content from device memory into main memory

Reimplemented from SWE Block.

```
7.29.3.14 void SWE_BlockCUDA::synchDischargeAfterWrite() [protected], [virtual]
```

Update temporary and non-local (for heterogeneous computing) variables after an external update of the discharge variables hu and hv

Reimplemented from SWE Block.

```
7.29.3.15 void SWE_BlockCUDA::synchDischargeBeforeRead() [protected], [virtual]
```

Update temporary and non-local (for heterogeneous computing) variables before an external access to the discharge variables hu and hv

Reimplemented from SWE Block.

```
7.29.3.16 void SWE_BlockCUDA::synchGhostLayerAfterWrite() [protected], [virtual]
```

synchronise the ghost layer content of h, hu, and hv in main memory with device memory and auxiliary data structures, i.e. transfer memory from main/auxiliary memory into device memory

Reimplemented from SWE Block.

```
7.29.3.17 void SWE_BlockCUDA::synchWaterHeightAfterWrite() [protected], [virtual]
```

Update temporary and non-local (for heterogeneous computing) variables after an external update of the water height h

Reimplemented from SWE Block.

```
7.29.3.18 void SWE_BlockCUDA::synchWaterHeightBeforeRead( ) [protected], [virtual]
```

Update temporary and non-local (for heterogeneous computing) variables before an external access to the water height h

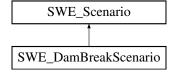
Reimplemented from SWE\_Block.

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

- /home/raphael/Programmieren/BPraktikum/SWE/src/blocks/cuda/SWE\_BlockCUDA.hh
- /home/raphael/Programmieren/BPraktikum/SWE/src/blocks/cuda/SWE\_BlockCUDA.cu

# 7.30 SWE DamBreakScenario Class Reference

Inheritance diagram for SWE\_DamBreakScenario:



**Public Member Functions** 

- float getBathymetry (float x, float y)
- float getWaterHeight (float x, float y)
- virtual float endSimulation ()

- virtual BoundaryType getBoundaryType (BoundaryEdge edge)
- float getBoundaryPos (BoundaryEdge i edge)

## 7.30.1 Member Function Documentation

7.30.1.1 float SWE\_DamBreakScenario::getBoundaryPos( BoundaryEdge i\_edge ) [inline], [virtual]

Get the boundary positions

**Parameters** 

```
i_edge which edge
```

#### Returns

value in the corresponding dimension

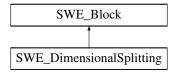
Reimplemented from SWE\_Scenario.

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

/home/raphael/Programmieren/BPraktikum/SWE/src/scenarios/SWE simple scenarios.hh

# 7.31 SWE\_DimensionalSplitting Class Reference

Inheritance diagram for SWE\_DimensionalSplitting:



# **Public Member Functions**

- SWE\_DimensionalSplitting (int I\_nx, int I\_ny, float I\_dx, float I\_dy)
- void simulateTimestep (float dt)

execute a single time step of the simulation

- float simulate (float tStart, float tEnd)
- · void computeNumericalFluxes ()
- void updateUnknowns (float dt)
- void runTimestep ()

# **Additional Inherited Members**

## 7.31.1 Member Function Documentation

**7.31.1.1 void SWE\_DimensionalSplitting::computeNumericalFluxes()** [virtual]

calculate the net-updates for the current state of the fluid, that can be applied later by updateUnknowns

Important if you change maxTimestep between this function an #updateUnkowns the accurancy of the calculation is not given any more.

Implements SWE\_Block.

7.31.1.2 void SWE\_DimensionalSplitting::runTimestep ( )

This funktion calculates and applays all changes for one Timestep

7.31.1.3 float SWE\_DimensionalSplitting::simulate (float tStart, float tEnd) [virtual]

This methode runs a simulation for the time intervall from tStart to tEnd Implements SWE\_Block.

**7.31.1.4** void SWE\_DimensionalSplitting::updateUnknowns (float dt) [virtual]

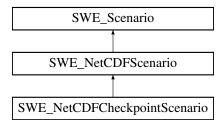
apply the net-updates calculated with computeNumericalFluxes Implements SWE\_Block.

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

- /home/raphael/Programmieren/BPraktikum/SWE/src/blocks/SWE\_DimensionalSplitting.hh
- /home/raphael/Programmieren/BPraktikum/SWE/src/blocks/SWE\_DimensionalSplitting.cpp

# 7.32 SWE\_NetCDFCheckpointScenario Class Reference

Inheritance diagram for SWE\_NetCDFCheckpointScenario:



## **Public Member Functions**

- SWE\_NetCDFCheckpointScenario ()
- int readNetCDF (const char \*filename, const char \*CPFile)
- float getWaterHeight (float x, float y)
- float getBathymetry (float x, float y)
- float getVeloc\_u (float x, float y)
- float **getVeloc\_v** (float x, float y)
- float getTime ()
- float getBoundaryPos (BoundaryEdge i\_edge)
- float endSimulation ()
- BoundaryType getBoundaryType (BoundaryEdge edge)

## 7.32.1 Constructor & Destructor Documentation

7.32.1.1 SWE NetCDFCheckpointScenario::SWE NetCDFCheckpointScenario() [inline]

load a scenario from a netCDF file

#### **Parameters**

file	the netCDF file to load

# 7.32.2 Member Function Documentation

7.32.2.1 int SWE\_NetCDFCheckpointScenario::readNetCDF ( const char \* filename, const char \* CPFile ) [inline]

readNetCDF will initialize the ids of the nc file and the ids of all the variables which are being used

#### **Parameters**

filename	the name of the nc-file to be opened
CPFile	filename of the checkpoint file

#### Returns

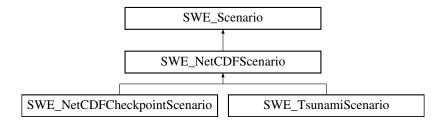
0 if successful, else the error value of the netcdf-library

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

/home/raphael/Programmieren/BPraktikum/SWE/src/scenarios/SWE\_NetCDFCheckpointScenario.hh

# 7.33 SWE\_NetCDFScenario Class Reference

Inheritance diagram for SWE NetCDFScenario:



### **Public Member Functions**

- virtual float **getWaterHeight** (float x, float y)
- virtual float **getVeloc\_u** (float x, float y)
- virtual float getVeloc\_v (float x, float y)
- virtual float getBathymetry (float x, float y)
- virtual int readNetCDF (char \*filename, char \*assistfile)
- virtual float waterHeightAtRest ()
- virtual float endSimulation ()
- virtual BoundaryType getBoundaryType (BoundaryEdge edge)
- virtual float getBoundaryPos (BoundaryEdge edge)

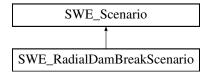
The documentation for this class was generated from the following file:

· /home/raphael/Programmieren/BPraktikum/SWE/src/scenarios/SWE\_NetCDFScenario.hh

# 7.34 SWE RadialDamBreakScenario Class Reference

#include <SWE\_simple\_scenarios.hh>

Inheritance diagram for SWE\_RadialDamBreakScenario:



# **Public Member Functions**

- float getBathymetry (float x, float y)
- float **getWaterHeight** (float x, float y)
- virtual float endSimulation ()
- virtual BoundaryType getBoundaryType (BoundaryEdge edge)
- float getBoundaryPos (BoundaryEdge i\_edge)

# 7.34.1 Detailed Description

Scenario "Radial Dam Break": elevated water in the center of the domain

#### 7.34.2 Member Function Documentation

**7.34.2.1 float SWE\_RadialDamBreakScenario::getBoundaryPos(BoundaryEdge** *i\_edge*) [inline], [virtual]

Get the boundary positions

Parameters

i_edge	which edge

## Returns

value in the corresponding dimension

Reimplemented from SWE\_Scenario.

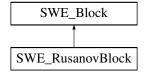
The documentation for this class was generated from the following file:

/home/raphael/Programmieren/BPraktikum/SWE/src/scenarios/SWE\_simple\_scenarios.hh

# 7.35 SWE\_RusanovBlock Class Reference

#include <SWE\_RusanovBlock.hh>

Inheritance diagram for SWE RusanovBlock:



## **Public Member Functions**

- SWE\_RusanovBlock (float \_offsetX=0, float \_offsetY=0)
- virtual ~SWE\_RusanovBlock ()
- virtual void simulateTimestep (float dt)

execute a single time step of the simulation

· virtual float simulate (float tStart, float tEnd)

compute simulate from specified start to end time

virtual void computeNumericalFluxes ()

compute flux terms on edges

virtual void updateUnknowns (float dt)

update unknowns according to fluxes (Euler time step)

#### **Protected Member Functions**

• virtual void computeBathymetrySources ()

compute source terms

- float computeLocalSV (int i, int j, char dir)
- virtual void computeMaxTimestep ()

# **Static Protected Member Functions**

• static float computeFlux (float fLoc, float fNeigh, float xiLoc, float xiNeigh, float llf)

# **Protected Attributes**

- Float2D Fh
- Float2D Fhu
- Float2D Fhv
- Float2D Gh
- Float2D Ghu
- Float2D Ghv
- Float2D Bx
- Float2D By

## **Friends**

ostream & operator<< (ostream &os, const SWE\_RusanovBlock &swe)</li>

## **Additional Inherited Members**

# 7.35.1 Detailed Description

SWE\_RusanovBlock is an implementation of the SWE\_Block abstract class. It uses a simple Rusanov flux (aka local Lax-Friedrich) in the model, with some simple modifications to obtain a well-balanced scheme.

#### 7.35.2 Constructor & Destructor Documentation

7.35.2.1 SWE\_RusanovBlock::SWE\_RusanovBlock ( float \_offsetX = 0, float \_offsetY = 0 )

Constructor: allocate variables for simulation

unknowns h,hu,hv,b are defined on grid indices [0,..,nx+1]\*[0,..,ny+1] -> computational domain is [1,..,nx]\*[1,..,ny] -> plus ghost cell layer

flux terms are defined for edges with indices [0,...,nx]\*[1,...,ny] or [1,...,nx]\*0, ..., ny Flux term with index (i,j) is located on the edge between cells with index (i,j) and (i+1,j) or (i,j+1)

bathymetry source terms are defined for cells with indices [1,..,nx]\*[1,..,ny]

@ param \_offsetX x coordinate of block origin @ param \_offsetY y coordinate of block origin

```
7.35.2.2 SWE_RusanovBlock: ~SWE_RusanovBlock( ) [virtual]
```

Destructor: de-allocate all variables

## 7.35.3 Member Function Documentation

```
7.35.3.1 void SWE_RusanovBlock::computeBathymetrySources() [protected], [virtual]
```

compute source terms

compute the bathymetry source terms in all cells

```
7.35.3.2 float SWE_RusanovBlock::computeFlux ( float fLow, float fHigh, float xiLow, float xiHigh, float llf ) [static], [protected]
```

compute the flux term on a given edge (acc. to local Lax-Friedrich method aka Rusanov flux): fLow and fHigh contain the values of the flux function in the two adjacent grid cells xiLow and xiHigh are the values of the unknowns in the two adjacent grid cells "Low" represents the cell with lower i/j index ("High" for larger index). Ilf should contain the local signal velocity (as compute by computeLocalSV) for llf=dx/dt (or dy/dt), we obtain the standard Lax Friedrich method

```
7.35.3.3 float SWE_RusanovBlock::computeLocalSV (int i, int j, char dir ) [protected]
```

computes the local signal velocity in x- or y-direction for two adjacent cells with indices (i,j) and (i+1,j) (if dir='x') or (i,j+1) (if dir='y'

```
7.35.3.4 void SWE_RusanovBlock::computeNumericalFluxes() [virtual]
```

compute flux terms on edges

compute the flux terms on all edges; before the computation, computeBathymetrySources is called Implements SWE\_Block.

```
7.35.3.5 float SWE_RusanovBlock::simulate ( float tStart, float tEnd ) [virtual]
```

compute simulate from specified start to end time

implements interface function simulate: perform forward-Euler time steps, starting with simulation time tStart,: until simulation time tEnd is reached; boundary conditions and bathymetry source terms are computed for each timestep as required - intended as main simulation loop between two checkpoints

Implements SWE\_Block.

**7.35.3.6** void SWE\_RusanovBlock::simulateTimestep ( float dt ) [virtual]

execute a single time step of the simulation

Depending on the current values of h, hu, hv (incl. ghost layers) update these unknowns in each grid cell (ghost layers and bathymetry are not updated). The Rusanov implementation of simulateTimestep subsequently calls the functions computeNumericalFluxes (to compute all fluxes on grid edges), and updateUnknowns (to update the variables according to flux values, typically according to an Euler time step).

**Parameters** 

```
dt size of the time step
```

Implements SWE Block.

7.35.3.7 void SWE\_RusanovBlock::updateUnknowns ( float dt ) [virtual]

update unknowns according to fluxes (Euler time step)

implements interface function updateUnknowns: based on the (Rusanov) fluxes computed on each edge (and stored in the variables Fh, Gh, etc.); compute the balance terms for each cell, and update the unknowns according to an Euler time step.

**Parameters** 

```
| dt | size of the time step.
```

Implements SWE\_Block.

# 7.35.4 Friends And Related Function Documentation

7.35.4.1 ostream& operator << ( ostream & os, const SWE\_RusanovBlock & swe ) [friend]

overload operator << such that data can be written via cout << -> needs to be declared as friend to be allowed to access private data

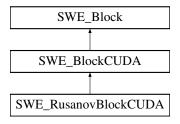
The documentation for this class was generated from the following files:

- /home/raphael/Programmieren/BPraktikum/SWE/src/blocks/rusanov/SWE RusanovBlock.hh
- /home/raphael/Programmieren/BPraktikum/SWE/src/blocks/rusanov/SWE\_RusanovBlock.cpp

# 7.36 SWE RusanovBlockCUDA Class Reference

#include <SWE\_RusanovBlockCUDA.hh>

Inheritance diagram for SWE\_RusanovBlockCUDA:



# **Public Member Functions**

- SWE RusanovBlockCUDA (float offsetX=0, float offsetY=0, const int i cudaDevice=0)
- virtual ~SWE\_RusanovBlockCUDA ()
- virtual void computeNumericalFluxes ()
- virtual void updateUnknowns (float dt)
- virtual void simulateTimestep (float dt)

execute a single time step of the simulation

· virtual float simulate (float tStart, float tEnd)

## **Friends**

ostream & operator<< (ostream &os, const SWE RusanovBlockCUDA &swe)</li>

#### **Additional Inherited Members**

# 7.36.1 Detailed Description

SWE\_RusanovBlockCUDA extends the base class SWE\_BlockCUDA, and provides a concrete CUDA implementation of a simple shallow water model based on Rusanov Flux computation on the edges and explicit time stepping.

#### 7.36.2 Constructor & Destructor Documentation

7.36.2.1 SWE\_RusanovBlockCUDA::SWE\_RusanovBlockCUDA ( float \_offsetX = 0, float \_offsetY = 0, const int i\_cudaDevice = 0 )

Constructor: allocate variables for simulation

unknowns h,hu,hv,b are defined on grid indices [0,...,nx+1]\*[0,...,ny+1] -> computational domain is [1,...,nx]\*[1,...,ny] -> plus ghost cell layer

flux terms are defined for edges with indices [0,...,nx]\*[1,...,ny] or [1,...,nx]\*[0,...,ny] Flux term with index (i,j) is located on the edge between cells with index (i,j) and (i+1,j) or (i,j+1)

bathymetry source terms are defined for cells with indices [1,..,nx]\*[1,..,ny]

```
7.36.2.2 SWE_RusanovBlockCUDA::~SWE_RusanovBlockCUDA( ) [virtual]
```

Destructor: de-allocate all variables

# 7.36.3 Member Function Documentation

```
7.36.3.1 void SWE_RusanovBlockCUDA::computeNumericalFluxes( ) [virtual]
```

compute the flux terms on all edges

Implements SWE Block.

```
7.36.3.2 __host__float SWE_RusanovBlockCUDA::simulate( float tStart, float tEnd ) [virtual]
```

perform forward-Euler time steps, starting with simulation time tStart,: until simulation time tEnd is reached; device-global variables hd, hud, hvd are updated; unknowns h, hu, hv in main memory are not updated. Ghost layers and bathymetry sources are updated between timesteps. intended as main simulation loop between two checkpoints

Implements SWE Block.

**7.36.3.3** void SWE\_RusanovBlockCUDA::simulateTimestep ( float dt ) [virtual]

execute a single time step of the simulation

Depending on the current values of h, hu, hv (incl. ghost layers) update these unknowns in each grid cell (ghost layers and bathymetry are not updated). The Rusanov CUDA-implementation of simulateTimestep subsequently calls the functions computeNumericalFluxes (to compute all fluxes on grid edges), and updateUnknowns (to update the variables according to flux values, typically according to an Euler time step).

**Parameters** 

```
dt size of the time step
```

Implements SWE\_Block.

```
7.36.3.4 __host__void SWE_RusanovBlockCUDA::updateUnknowns ( float dt ) [virtual]
```

implements interface function updateUnknowns: based on the (Rusanov) fluxes computed on each edge (and stored in the variables Fh, Gh, etc.); compute the balance terms for each cell, and update the unknowns according to an Euler time step. It will force an update of the copy layer in the main memory by calling synchCopyLayerBefore-Read(), and provide an compute the maximum allowed time step size by calling computeMaxTimestepCUDA().

**Parameters** 

```
dt size of the time step.
```

Implements SWE\_Block.

## 7.36.4 Friends And Related Function Documentation

```
7.36.4.1 ostream& operator<< ( ostream & os, const SWE_RusanovBlockCUDA & swe ) [friend]
```

overload operator << such that data can be written via cout << -> needs to be declared as friend to be allowed to access private data

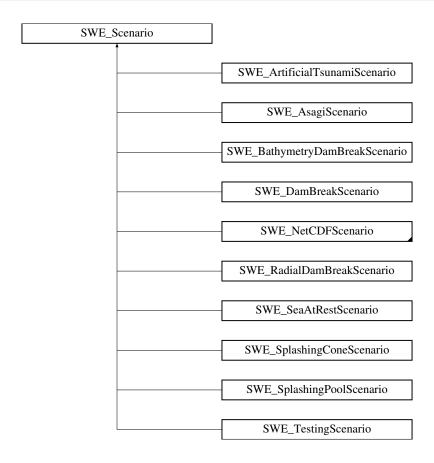
The documentation for this class was generated from the following files:

- /home/raphael/Programmieren/BPraktikum/SWE/src/blocks/rusanov/SWE\_RusanovBlockCUDA.hh
- /home/raphael/Programmieren/BPraktikum/SWE/src/blocks/rusanov/SWE RusanovBlockCUDA.cu

# 7.37 SWE\_Scenario Class Reference

#include <SWE\_Scenario.hh>

Inheritance diagram for SWE\_Scenario:



# **Public Member Functions**

- virtual float getWaterHeight (float x, float y)
- virtual float **getVeloc\_u** (float x, float y)
- virtual float getVeloc\_v (float x, float y)
- virtual float **getBathymetry** (float x, float y)
- virtual float waterHeightAtRest ()
- virtual float endSimulation ()
- virtual BoundaryType getBoundaryType (BoundaryEdge edge)
- virtual float **getBoundaryPos** (BoundaryEdge edge)

# 7.37.1 Detailed Description

SWE\_Scenario defines an interface to initialise the unknowns of a shallow water simulation - i.e. to initialise water height, velocities, and bathymatry according to certain scenarios. SWE\_Scenario can act as stand-alone scenario class, providing a very basic scenario (all functions are constant); however, the idea is to provide derived classes that implement the SWE\_Scenario interface for more interesting scenarios.

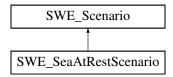
The documentation for this class was generated from the following file:

/home/raphael/Programmieren/BPraktikum/SWE/src/scenarios/SWE\_Scenario.hh

# 7.38 SWE SeaAtRestScenario Class Reference

#include <SWE\_simple\_scenarios.hh>

Inheritance diagram for SWE\_SeaAtRestScenario:



#### **Public Member Functions**

- float getWaterHeight (float x, float y)
- float getBathymetry (float x, float y)

# 7.38.1 Detailed Description

Scenario "Sea at Rest": flat water surface ("sea at rest"), but non-uniform bathymetry (id. to "Bathymetry Dam Break") test scenario for "sea at rest"-solution

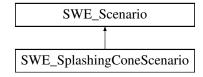
The documentation for this class was generated from the following file:

/home/raphael/Programmieren/BPraktikum/SWE/src/scenarios/SWE simple scenarios.hh

# 7.39 SWE\_SplashingConeScenario Class Reference

#include <SWE\_simple\_scenarios.hh>

Inheritance diagram for SWE\_SplashingConeScenario:



# **Public Member Functions**

- float getWaterHeight (float x, float y)
- float getBathymetry (float x, float y)
- float waterHeightAtRest ()
- float endSimulation ()
- virtual BoundaryType getBoundaryType (BoundaryEdge edge)

# 7.39.1 Detailed Description

Scenario "Splashing Cone": bathymetry forms a circular cone intial water surface designed to form "sea at rest" but: elevated water region in the centre (similar to radial dam break)

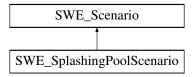
The documentation for this class was generated from the following file:

/home/raphael/Programmieren/BPraktikum/SWE/src/scenarios/SWE\_simple\_scenarios.hh

# 7.40 SWE\_SplashingPoolScenario Class Reference

#include <SWE\_simple\_scenarios.hh>

Inheritance diagram for SWE\_SplashingPoolScenario:



## **Public Member Functions**

- float getBathymetry (float x, float y)
- float getWaterHeight (float x, float y)
- virtual float endSimulation ()
- float getBoundaryPos (BoundaryEdge i\_edge)

# 7.40.1 Detailed Description

Scenario "Splashing Pool": intial water surface has a fixed slope (diagonal to x,y)

## 7.40.2 Member Function Documentation

7.40.2.1 float SWE\_SplashingPoolScenario::getBoundaryPos ( BoundaryEdge i\_edge ) [inline], [virtual]

Get the boundary positions

Parameters

i_edge	which edge

### Returns

value in the corresponding dimension

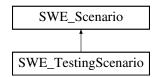
Reimplemented from SWE\_Scenario.

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

• /home/raphael/Programmieren/BPraktikum/SWE/src/scenarios/SWE\_simple\_scenarios.hh

# 7.41 SWE\_TestingScenario Class Reference

Inheritance diagram for SWE\_TestingScenario:



## **Public Member Functions**

- float getBathymetry (float x, float y)
- float getWaterHeight (float x, float y)
- virtual float endSimulation ()
- virtual BoundaryType getBoundaryType (BoundaryEdge edge)
- float getBoundaryPos (BoundaryEdge i\_edge)

## 7.41.1 Member Function Documentation

7.41.1.1 float SWE\_TestingScenario::getBoundaryPos ( BoundaryEdge i\_edge ) [inline], [virtual]

Get the boundary positions

**Parameters** 

i_edge	which edge

## Returns

value in the corresponding dimension

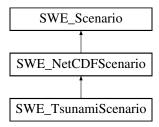
Reimplemented from SWE\_Scenario.

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

/home/raphael/Programmieren/BPraktikum/SWE/src/testing/testing\_scenario.hh

# 7.42 SWE\_TsunamiScenario Class Reference

Inheritance diagram for SWE TsunamiScenario:



# **Public Member Functions**

- SWE\_TsunamiScenario ()
- float getWaterHeight (float x, float y)
- float getBathymetry (float x, float y)
- float getVeloc\_u (float x, float y)
- float **getVeloc\_v** (float x, float y)
- float getBoundaryPos (BoundaryEdge i\_edge)
- BoundaryType getBoundaryType (BoundaryEdge edge)
- int readNetCDF (const char \*filename, const char \*d\_filename)

# 7.42.1 Constructor & Destructor Documentation

**7.42.1.1 SWE\_TsunamiScenario::SWE\_TsunamiScenario( )** [inline]

load a scenario from a netCDF file

#### **Parameters**

file	the netCDF file to load

#### 7.42.2 Member Function Documentation

7.42.2.1 int SWE\_TsunamiScenario::readNetCDF ( const char \* filename, const char \* d\_filename ) [inline]

readNetCDF will initialize the ids of the nc file and the ids of all the variables which are being used Parameters

filename	the name of the nc-file to be opened

#### Returns

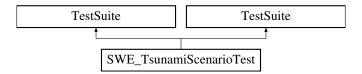
0 if successful, else the error value of the netcdf-library

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

/home/raphael/Programmieren/BPraktikum/SWE/src/scenarios/SWE TsunamiScenario.hh

# 7.43 SWE\_TsunamiScenarioTest Class Reference

Inheritance diagram for SWE\_TsunamiScenarioTest:



## **Public Member Functions**

- · void testgetBoundaryPos (void)
- void testgetBathymetry (void)
- void testcornners (void)
- void testpossibleScenario (void)
- · void testgetOriginalBathymetry (void)
- void testgetWaterHeight (void)
- void testgetVelco (void)
- void testtoGridCoordinates (void)
- void testgetBoundaryPos ()

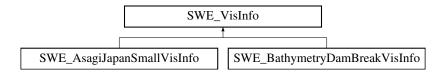
The documentation for this class was generated from the following files:

- /home/raphael/Programmieren/BPraktikum/SWE/src/scenarios/SWE\_TsunamiScenarioTest.hh
- /home/raphael/Programmieren/BPraktikum/SWE/src/scenarios/SWE\_TsunamiScenarioTest.t.h

# 7.44 SWE VisInfo Class Reference

```
#include <SWE_VisInfo.hh>
```

Inheritance diagram for SWE\_VisInfo:



## **Public Member Functions**

- virtual ∼SWE\_VisInfo ()
- virtual float waterVerticalScaling ()
- virtual float bathyVerticalOffset ()
- · virtual float bathyVerticalScaling ()

# 7.44.1 Detailed Description

SWE\_VisInfo defines an interface that can be used for online visualization of a shallow water simulation. In particular, it provides information required for proper scaling of the involved variables.

For water height: displayedWaterHeight = waterVerticalScaling() \* simulatedWaterHeight

For bathymetry: displayedBatyhmetry = bathyVerticalScaling() \* realBathymetry

bathyVerticalOffset()

The default water height should be 0. In this case a bathymetry value smaller than 0 means water and a value greater than 0 is land. Therefore bathyVerticalOffset should 0 for all real scenarios.

If you do not not provide an SWE\_VisInfo for scenario, (water|bathy)VerticalScaling will be guessed form the value initial values. bathyVerticalOffset is always 0 in this case.

## 7.44.2 Constructor & Destructor Documentation

```
7.44.2.1 virtual SWE_VisInfo::~SWE_VisInfo() [inline], [virtual]
```

Empty virtual destructor

#### 7.44.3 Member Function Documentation

```
7.44.3.1 virtual float SWE_VisInfo::bathyVerticalOffset() [inline], [virtual]
```

Returns

The vertical offset for the bathymetry. Should be 0 for "real" scenarios (scenarios with dry areas)

Reimplemented in SWE BathymetryDamBreakVisInfo.

```
7.44.3.2 virtual float SWE_VisInfo::bathyVerticalScaling( ) [inline], [virtual]
```

Returns

The vertical scaling factor for the bathymetry

Reimplemented in SWE\_AsagiJapanSmallVisInfo.

7.44.3.3 virtual float SWE\_VisInfo::waterVerticalScaling() [inline], [virtual]

Returns

The vertical scaling factor of the water

Reimplemented in SWE\_AsagiJapanSmallVisInfo.

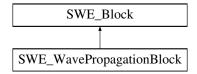
The documentation for this class was generated from the following file:

• /home/raphael/Programmieren/BPraktikum/SWE/src/scenarios/SWE VisInfo.hh

# 7.45 SWE\_WavePropagationBlock Class Reference

#include <SWE\_WavePropagationBlock.hh>

Inheritance diagram for SWE WavePropagationBlock:



# **Public Member Functions**

- SWE\_WavePropagationBlock (int I\_nx, int I\_ny, float I\_dx, float I\_dy)
- virtual void simulateTimestep (float dt)
- · void computeNumericalFluxes ()
- void updateUnknowns (float dt)
- float simulate (float i\_tStart, float i\_tEnd)
- virtual ~SWE\_WavePropagationBlock ()

#### **Additional Inherited Members**

# 7.45.1 Detailed Description

SWE\_WavePropagationBlock is an implementation of the SWE\_Block abstract class. It uses a wave propagation solver which is defined with the pre-compiler flag WAVE PROPAGATION SOLVER (see above).

Possible wave propagation solvers are: F-Wave, Apprximate Augmented Riemann, Hybrid (f-wave + augmented). (details can be found in the corresponding source files)

#### 7.45.2 Constructor & Destructor Documentation

7.45.2.1 SWE\_WavePropagationBlock::SWE\_WavePropagationBlock ( int I\_nx, int I\_ny, float I\_dx, float I\_dy )

Constructor of a SWE\_WavePropagationBlock.

Allocates the variables for the simulation: unknowns h,hu,hv,b are defined on grid indices [0,..,nx+1]\*[0,..,ny+1] (-> Abstract class SWE\_Block) -> computational domain is [1,..,nx]\*[1,..,ny] -> plus ghost cell layer

```
net-updates are defined for edges with indices [0,..,nx]*[0,..,ny-1] or [0,..,nx-1]*0,...,ny
```

A left/right net update with index (i-1,j-1) is located on the edge between cells with index (i-1,j) and (i,j):

A below/above net update with index (i-1, j-1) is located on the edge between cells with index (i, j-1) and (i,j):

7.45.2.2 virtual SWE\_WavePropagationBlock::~SWE\_WavePropagationBlock( ) [inline], [virtual]

Destructor of a SWE WavePropagationBlock.

In the case of a hybrid solver (NDEBUG not defined) information about the used solvers will be printed.

# 7.45.3 Member Function Documentation

```
7.45.3.1 void SWE_WavePropagationBlock::computeNumericalFluxes() [virtual]
```

Compute net updates for the block. The member variable maxTimestep will be updated with the maximum allowed time step size

Implements SWE\_Block.

7.45.3.2 float SWE\_WavePropagationBlock::simulate ( float i\_tStart, float i\_tEnd ) [virtual]

Runs the simulation until i\_tEnd is reached.

#### **Parameters**

i_tStart	time when the simulation starts
i_tEnd	time when the simulation should end

## Returns

time we reached after the last update step, in general a bit later than i\_tEnd

Implements SWE Block.

**7.45.3.3 void SWE\_WavePropagationBlock::simulateTimestep (float** *dt***)** [virtual]

Update the bathymetry values with the displacement corresponding to the current time step.

# **Parameters**

i_asagiScenario	the corresponding ASAGI-scenario Executes a single timestep.
	compute net updates for every edge
	update cell values with the net updates
dt	time step width of the update

Implements SWE Block.

7.45.3.4 void SWE\_WavePropagationBlock::updateUnknowns (float dt) [virtual]

Updates the unknowns with the already computed net-updates.

## **Parameters**

dt	time step width used in the update.

Implements SWE\_Block.

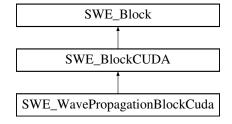
The documentation for this class was generated from the following files:

- /home/raphael/Programmieren/BPraktikum/SWE/src/blocks/SWE WavePropagationBlock.hh
- /home/raphael/Programmieren/BPraktikum/SWE/src/blocks/SWE\_WavePropagationBlock.cpp

# 7.46 SWE\_WavePropagationBlockCuda Class Reference

#include <SWE\_WavePropagationBlockCuda.hh>

Inheritance diagram for SWE\_WavePropagationBlockCuda:



## **Public Member Functions**

- SWE WavePropagationBlockCuda (int I nx, int I ny, float I dx, float I dy)
- ~SWE\_WavePropagationBlockCuda ()
- void simulateTimestep (float i dT)
- float simulate (float, float)
- void computeNumericalFluxes ()
- void updateUnknowns (const float i\_deltaT)

### **Additional Inherited Members**

## 7.46.1 Detailed Description

SWE\_WavePropagationBlockCuda is an implementation of the SWE\_BlockCuda abstract class. It uses a wave propagation solver which is defined with the pre-compiler flag WAVE\_PROPAGATION\_SOLVER (see above).

Possible wave propagation solvers are: F-Wave, <strike>Approximate Augmented Riemann, Hybrid (f-wave + augmented).</strike> (details can be found in the corresponding source files)

## 7.46.2 Constructor & Destructor Documentation

7.46.2.1 SWE\_WavePropagationBlockCuda::SWE\_WavePropagationBlockCuda (int I\_nx, int I\_ny, float I\_dx, float I\_dy)

Constructor of a SWE\_WavePropagationBlockCuda.

Allocates the variables for the simulation: Please note: The definition of indices changed in contrast to the CPU-Implementation.

unknowns hd,hud,hvd,bd stored on the CUDA device are defined for grid indices [0,...,nx+1]\*[0,...,ny+1] (-> Abstract class SWE\_BlockCUDA) -> computational domain is [1,...,nx]\*[1,...,ny] -> plus ghost cell layer

net-updates are defined for edges with indices [0,...,nx]\*[0,...,ny] for horizontal and vertical edges for simplicity (one layer is not necessary).

A left/right net update with index (i-1,j) is located on the edge between cells with index (i-1,j) and (i,j):

A below/above net update with index (i, j-1) is located on the edge between cells with index (i, j-1) and (i,j):

```
*******

* (i, j) * *

* (i, j) * *

* NetUpdatesBelow(i, j-1)

**********

or

* * * NetUpdatesAbove(i, j-1)

* (i, j-1) * *
```

\* \* \*\*\*\*\*\*\*

#### **Parameters**

i_offsetX	spatial offset of the block in x-direction.
i_offsetY	spatial offset of the offset in y-direction.
i_cudaDevice	ID of the CUDA-device, which should be used.

#### 7.46.2.2 SWE\_WavePropagationBlockCuda::~SWE\_WavePropagationBlockCuda ( )

Destructor of a SWE\_WavePropagationBlockCuda.

Frees all of the memory, which was allocated within the constructor. Resets the CUDA device: Useful if error occured and printf is used on the device (buffer).

#### 7.46.3 Member Function Documentation

7.46.3.1 void SWE\_WavePropagationBlockCuda::computeNumericalFluxes( ) [virtual]

Compute the numerical fluxes (net-update formulation here) on all edges.

The maximum wave speed is computed within the net-updates kernel for each CUDA-block. To finalize the method the Thrust-library is called, which does the reduction over all blockwise maxima. In the wave speed reduction step the actual cell width in x- and y-direction is not taken into account.

TODO: A splitting or direct computation of the time step width might increase the total time step size. Example: dx = 11, dy = 6; max wave speed in x-direction: 10 max wave speed in y-direction: 5.5 max wave speed in both direction: 10

=> maximum time step (current implementation): min(11/10, 6/10) = 0.6 => maximum time step (splitting the dimensions): <math>min(11/10, 6/5.5) = 1.09.. **Row-major vs column-major** 

C/C++ arrays are row-major whereas warps are constructed in column-major order from threads/blocks. To get coalesced memory access in CUDA, we can use a 2-dimensional CUDA structure but we have to switch x and y inside a block.

This means, we have to switch threadIdx.x <-> threadIdx.y as well as blockDim.x <-> blockDim.y. Important: blockDim has to be switched for the kernel call as well!

definition of one CUDA-block. Typical size are 8\*8 or 16\*16

Definition of the "main" CUDA-grid. This grid covers only edges 0..#(edges in x-direction)-2 and 0..#(edges in y-direction)-2.

An example with a computational domain of size nx = 24, ny = 16 with a 1 cell ghost layer would result in a grid with (nx+2)\*(ny+2) = (26\*18) cells and (nx+1)\*(ny+1) = (25\*17) edges.

The CUDA-blocks (here 8\*8) mentioned above would cover all edges except the ones lying between the computational domain and the right/top ghost layer:

```
top ghost laver,
                                                                cell ids
                                                                 = (*, ny+1)
                        8 * 8
                                 8 * 8
                                            8 * 8
                      block
                                 block
                                           block
                      ******
                       8 * 8
                                 8 * 8
                                        *
                                            8 * 8
                      block
                                block
                                           block
bottom
               * *
ghost
layer,
```

Implements SWE Block.

```
7.46.3.2 __host__float SWE_WavePropagationBlockCuda::simulate( float tStart, float tEnd ) [virtual]
```

perform forward-Euler time steps, starting with simulation time tStart,: until simulation time tEnd is reached; device-global variables hd, hud, hvd are updated; unknowns h, hu, hv in main memory are not updated. Ghost layers and bathymetry sources are updated between timesteps. intended as main simulation loop between two checkpoints Implements SWE Block.

```
7.46.3.3 __host__void SWE_WavePropagationBlockCuda::simulateTimestep ( float i_dT ) [virtual]
```

Compute a single global time step of a given time step width. Remark: The user has to take care about the time step width. No additional check is done. The time step width typically available after the computation of the numerical fluxes (hidden in this method).

First the net-updates are computed. Then the cells are updated with the net-updates and the given time step width.

#### **Parameters**

```
i_{-}dT time step width in seconds.
```

Implements SWE\_Block.

7.46.3.4 void SWE\_WavePropagationBlockCuda::updateUnknowns ( const float i\_deltaT ) [virtual]

Update the cells with a given global time step.

**Parameters** 

```
i_deltaT | time step size.
```

definition of one CUDA-block. Typical size are 8\*8 or 16\*16

definition of the CUDA-grid.

Implements SWE\_Block.

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

- /home/raphael/Programmieren/BPraktikum/SWE/src/blocks/cuda/SWE\_WavePropagationBlockCuda.hh
- /home/raphael/Programmieren/BPraktikum/SWE/src/blocks/cuda/SWE\_WavePropagationBlockCuda.cu

# 7.47 Text Class Reference

**Public Member Functions** 

- void addText (const char \*text)
- void startTextMode ()
- bool showNextText (SDL\_Rect &location)
- void endTextMode ()

## 7.47.1 Member Function Documentation

## 7.47.1.1 bool Text::showNextText ( SDL\_Rect & location ) [inline]

#### Returns

True there are more textures

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

- /home/raphael/Programmieren/BPraktikum/SWE/src/opengl/text.h
- · /home/raphael/Programmieren/BPraktikum/SWE/src/opengl/text.cpp

# 7.48 tools::Args Class Reference

```
#include <args.h>
```

## **Public Member Functions**

- Args (int argc, char \*\*argv)
- unsigned int size ()
- unsigned int timeSteps ()

# 7.48.1 Detailed Description

Parse command line arguments

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

/home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/tools/args.h

# 7.49 tools::Logger Class Reference

# **Public Types**

• enum Level { INFO, WARNING, ERROR }

## **Public Member Functions**

- void setOutputStream (std::ostream &output)
- void log (std::string &message, Level level=INFO)
- void log (const char \*message, Level level=INFO)
- void info (std::string &message)
- void info (const char \*message)
- std::ostream & info ()
- void warning (std::string &message)
- void warning (const char \*message)
- std::ostream & warning ()
- void error (std::string &message)
- void error (const char \*message)
- template<typename T >
   Logger & operator<<< (T value)</p>

- Logger & operator<< (std::ostream &(\*func)(std::ostream &))</li>
- virtual ~Logger ()
- void printWelcomeMessage ()
- void printFinishMessage ()
- std::ostream & cout ()
- void setProcessRank (const int i processRank)
- void printString (const std::string i string)
- void printNumberOfProcesses (const int i\_numberOfProcesses, const std::string i\_processesName="MPI processes")
- void printNumberOfCells (const int i\_nX, const int i\_nY, const std::string i\_cellMessage="cells")
- void printNumberOfCellsPerProcess (const int i\_nX, const int i\_nY)
- void printCellSize (const float i dX, const float i dY, const std::string i unit="m")
- void printNumberOfBlocks (const int i\_nX, const int i\_nY)
- void printStartMessage (const std::string i\_startMessage="Everything is set up, starting the simulation.")
- void printSimulationTime (const float i\_time, const std::string i\_simulationTimeMessage="Simulation at time")
- void printOutputFileCreation (const std::string i\_fileName, const int i\_blockX, const int i\_blockY, const std::string i\_fileType="netCDF")
- void printOutputTime (const float i\_time, const std::string i\_outputTimeMessage="Writing output file at time")
- void printStatisticsMessage (const std::string i\_statisticsMessage="Simulation finished. Printing statistics for each process.")
- void printSolverStatistics (const long i\_firstSolverCounter, const long i\_secondSolverCounter, const int i\_blockY=0, const int i\_blockY=0, const std::string i\_firstSolverName="f-Wave solver", const std::string i\_secondSolverName="Augemented Riemann solver")
- void updateCpuTime ()
- void updateCpuCommunicationTime ()
- void resetCpuClockToCurrentTime ()
- void resetCpuCommunicationClockToCurrentTime ()
- void initWallClockTime (const double i wallClockTime)
- void printWallClockTime (const double i\_wallClockTime, const std::string i\_wallClockTimeMessage="wall clock time")
- void printCpuTime (const std::string i\_cpuTimeMessage="CPU time")
- void printCpuCommunicationTime (const std::string i\_cpuCommunicationTimeMessage="CPU + communication time")
- void printIterationsDone (unsigned int i\_iterations, std::string i\_iterationMessage="iterations done")

## **Static Public Attributes**

· static Logger logger

## 7.49.1 Constructor & Destructor Documentation

The Constructor. Prints the welcome message (process rank 0 only).

#### **Parameters**

i_processRank	rank of the constructing process.
i_programName	definition of the program name.
i_welcome-	definition of the welcome message.
Message	
i_startMessage	definition of the start message.
i_simulation-	definition of the simulation time message.
TimeMessage	
i_executionTime-	definition of the execution time message.
Message	
i_cpuTime-	definition of the CPU time message.
Message	
i_finishMessage	definition of the finish message.
i_midDelimiter	definition of the mid-size delimiter.
i_largeDelimiter	definition of the large delimiter.
i_indentation	definition of the indentation (used in all messages, except welcome, start and finish).

7.49.1.2 virtual tools::Logger::~Logger() [inline], [virtual]

The Destructor. Prints the finish message (process rank 0 only).

## 7.49.2 Member Function Documentation

7.49.2.1 std::ostream& tools::Logger::cout() [inline]

Default output stream of the logger.

# Returns

extended (time + indentation) std::cout stream.

7.49.2.2 void tools::Logger::initWallClockTime ( const double i\_wallClockTime ) [inline]

Initialize the wall clock time.

**Parameters** 

i_wallClockTime	value the wall block time will be set to.

7.49.2.3 template < typename T > Logger & tools::Logger::operator << ( T value ) [inline]

Can be used to print arbitrary info messages. Does not append std::endl.

7.49.2.4 Logger& tools::Logger::operator<<< ( std::ostream &(\*)(std::ostream &) func ) [inline]

Allow to print std::endl

7.49.2.5 void tools::Logger::printCellSize ( const float  $i_dX$ , const float  $i_dY$ , const std::string  $i_unit = "m"$  ) [inline]

Print the size of a cell

#### **Parameters**

i_dX	size in x-direction.
i_dY	size in y-direction.
i_unit	measurement unit.

7.49.2.6 void tools::Logger::printCpuCommunicationTime ( const std::string i\_cpuCommunicationTimeMessage = "CPU + communication time" ) [inline]

Print elapsed CPU + communication time.

#### **Parameters**

i_cpu-	CPU + communication time message.
Communication-	
TimeMessage	

7.49.2.7 void tools::Logger::printCpuTime ( const std::string i\_cpuTimeMessage = "CPU time" ) [inline]

Print elapsed CPU time.

#### **Parameters**

i_cpuTime-	cpu time message.
Message	

7.49.2.8 void tools::Logger::printFinishMessage() [inline]

Print the finish message.

7.49.2.9 void tools::Logger::printlterationsDone ( unsigned int *i\_iterations*, std::string *i\_iterationMessage* = "iterations done") [inline]

Print number of iterations done

## **Parameters**

i_iterations	Number of iterations done
i_interation-	Iterations done message
Message	

7.49.2.10 void tools::Logger::printNumberOfBlocks ( const int *i\_nX*, const int *i\_nY* ) [inline]

Print the number of defined blocks. (process rank 0 only)

### **Parameters**

i_nX	number of blocks in x-direction.
i_nY	number of blocks in y-direction.

7.49.2.11 void tools::Logger::printNumberOfCells ( const int i\_nX, const int i\_nY, const std::string i\_cellMessage = "cells" ) [inline]

Print the number of cells. (process rank 0 only)

#### **Parameters**

i_nX	number of cells in x-direction.
i_nY	number of cells in y-direction.
i_cellMessage	cell message.

7.49.2.12 void tools::Logger::printNumberOfCellsPerProcess ( const int i\_nX, const int i\_nY) [inline]

Print the number of cells per Process.

## **Parameters**

i_nX	number of cells in x-direction.
i_nY	number of cells in y-direction.

7.49.2.13 void tools::Logger::printNumberOfProcesses ( const int i\_numberOfProcesses, const std::string i\_processesName = "MPI processes" ) [inline]

Print the number of processes. (process rank 0 only)

#### **Parameters**

i_numberOf-	number of processes.
Processes	
i_processes-	name of the processes.
Name	

7.49.2.14 void tools::Logger::printOutputFileCreation ( const std::string i\_fileName, const int i\_blockX, const int i\_blockX, const int i\_blockY, const std::string i\_fileType = "netCDF" ) [inline]

Print the creation of an output file.

## **Parameters**

i_fileName	name of the file.
i_blockX	block position in x-direction.
i_blockY	block position in y-direction.
i_fileType	type of the output file.

7.49.2.15 void tools::Logger::printOutputTime ( const float i\_time, const std::string i\_outputTimeMessage = "Writing output file at time") [inline]

Print the current output time.

# **Parameters**

_		
	i_time	time in seconds.
Ī	i_outputTime-	output message.
	Message	

7.49.2.16 void tools::Logger::printSimulationTime ( const float *i\_time*, const std::string *i\_simulationTimeMessage* = "Simulation at time") [inline]

Print current simulation time. (process rank 0 only)

#### **Parameters**

i_time	time in seconds.
--------	------------------

7.49.2.17 void tools::Logger::printSolverStatistics ( const long i\_firstSolverCounter, const long i\_secondSolverCounter, const int i\_blockX = 0, const int i\_blockY = 0, const std::string i\_firstSolverName = "f-Wave solver", const std::string i\_secondSolverName = "Augemented Riemann solver") [inline]

#### Print solver statistics

#### **Parameters**

i_firstSolver-	times the first solver was used.
Counter	
i_secondSolver-	times the second solver was used.
Counter	
i_blockX	position of the block in x-direction
i_blockY	position of the block in y-direction
i_firstSolver-	name of the first solver.
Name	
i_secondSolver-	name of the second solver.
Name	

```
7.49.2.18 void tools::Logger::printStartMessage ( const std::string i_startMessage =
```

```
"Everything is set up, starting the simulation." ) [inline]
```

Print the start message. (process rank 0 only)

7.49.2.19 void tools::Logger::printStatisticsMessage ( const std::string *i\_statisticsMessage* =

```
"Simulation finished. Printing statistics for each process." ) [inline]
```

Print the statics message.

# **Parameters**

i statistics-	statistics message.
	Stationio mossage.
Message	
Message	

7.49.2.20 void tools::Logger::printString ( const std::string i\_string ) [inline]

Print an arbitrary string.

# **Parameters**

i_string	some string.

7.49.2.21 void tools::Logger::printWallClockTime ( const double i\_wallClockTime, const std::string i\_wallClockTimeMessage = "wall clock time" ) [inline]

Print the elapsed wall clock time.

#### **Parameters**

i_wallClockTime	wall clock time message.
-----------------	--------------------------

7.49.2.22 void tools::Logger::printWelcomeMessage( ) [inline]

Print the welcome message.

7.49.2.23 void tools::Logger::setProcessRank ( const int i\_processRank ) [inline]

Set the process rank.

#### **Parameters**

i processRank	process rank.
i_processi iarik	process rank.

7.49.2.24 void tools::Logger::updateCpuCommunicationTime( ) [inline]

Update the CPU-Communication time.

**7.49.2.25** void tools::Logger::updateCpuTime() [inline]

Update the CPU time.

## 7.49.3 Member Data Documentation

**7.49.3.1 static Logger tools::Logger::logger** [static]

The logger all classes shoud use

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

- /home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/tools/logger.h
- · /home/raphael/Programmieren/BPraktikum/SWE/src/tools/Logger.hh
- /home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/tools/logger.cpp
- /home/raphael/Programmieren/BPraktikum/SWE/src/tools/Logger.cpp

# 7.50 tools::ProgressBar Class Reference

**Public Member Functions** 

- ProgressBar (float totalWork=1., int rank=0)
- void update (float done)
- void clear ()

# 7.50.1 Member Function Documentation

7.50.1.1 void tools::ProgressBar::update ( float done ) [inline]

7.51 VBO Class Reference 87

#### **Parameters**

done	The amount of work already done

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

· /home/raphael/Programmieren/BPraktikum/SWE/src/tools/ProgressBar.hh

# 7.51 VBO Class Reference

#### **Public Member Functions**

- void init ()
- · GLuint getName ()
- void setBufferData (GLsizei size, const void \*data, GLenum target=GL\_ARRAY\_BUFFER, GLenum usage=GL\_STATIC\_DRAW)
- void bindBuffer (GLenum target=GL\_ARRAY\_BUFFER)
- · void finialize ()

# 7.51.1 Member Function Documentation

```
7.51.1.1 void VBO::finialize() [inline]Frees all associated memory7.51.1.2 GLuint VBO::getName() [inline]
```

Returns

The OpenGL name of the buffer

```
7.51.1.3 void VBO::init ( )
```

Initializes the object

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

- /home/raphael/Programmieren/BPraktikum/SWE/src/opengl/vbo.h
- /home/raphael/Programmieren/BPraktikum/SWE/src/opengl/vbo.cpp

# 7.52 Visualization Class Reference

# **Public Member Functions**

- Visualization (int windowWidth, int windowHeight, const char \*window\_title)
- ∼Visualization ()
- void init (Simulation &sim, SWE\_VisInfo \*visInfo=0L)
- void cleanUp ()
- cudaGraphicsResource \*\* getCudaNormalsPtr ()
- cudaGraphicsResource \*\* getCudaWaterSurfacePtr ()
- void renderDisplay ()
- void modifyWaterScaling (float factor)

88 Class Documentation

- void setRenderingMode (RenderMode mode)
- void toggleRenderingMode ()
- int resizeWindow (int newWidth, int newHeight)

#### **Static Public Member Functions**

• static bool isExtensionSupported (const char \*szTargetExtension)

# **Public Attributes**

• Camera \* camera

#### 7.52.1 Constructor & Destructor Documentation

7.52.1.1 Visualization::Visualization ( int windowWidth, int windowHeight, const char \* window\_title )

Constructor. All dimensions are node-based, this means a grid consisting of 2x2 cells would have 3x3 nodes.

#### **Parameters**

window_title title of the window created
_grid_x_size number of nodes of the grid (in x-direction)
_grid_y_size number of nodes of the grid (in y-direction)

#### 7.52.1.2 Visualization::~Visualization ( )

Destructor (see note below)

# 7.52.2 Member Function Documentation

7.52.2.1 void Visualization::cleanUp ( )

Frees all memory we used for geometry data Needs to be called before destructor gets called in order to work correctly

7.52.2.2 cudaGraphicsResource \*\* Visualization::getCudaNormalsPtr ( )

Returns a pointer to the cuda memory object holding the vertex normals

7.52.2.3 cudaGraphicsResource \*\* Visualization::getCudaWaterSurfacePtr ( )

Returns a pointer to the cuda memory object holding the vertex positions

7.52.2.4 void Visualization::init ( Simulation & sim, SWE\_VisInfo \* visInfo = 0L )

Allocates memory for vertices and other geometry data.

**Parameters** 

sim	instance of the simulation class

7.52.2.5 bool Visualization::isExtensionSupported ( const char \* szTargetExtension ) [static]

Returns, whether a special extension is supported by the current graphics card

#### **Parameters**

szTarget-	string describing the extension to look for
Extention	

7.52.2.6 void Visualization::renderDisplay ( )

Main rendering function. Draws the scene and updates screen

7.52.2.7 int Visualization::resizeWindow (int newWidth, int newHeight)

Gets called when window gets resized

#### **Parameters**

newWidth	new window width in pixels
newHeight	height in pixels

7.52.2.8 void Visualization::setRenderingMode ( RenderMode mode )

Sets current rendering mode

#### Parameters

mode	rendering mode

7.52.2.9 void Visualization::toggleRenderingMode ( )

Switches between 3 different rendering modes:

- · Shaded: Use OpenGL shading
- Wireframe: Only render edges of each triangle
- · Watershader: Use custom GLSL shader for water surface

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

- /home/raphael/Programmieren/BPraktikum/SWE/src/opengl/visualization.h
- /home/raphael/Programmieren/BPraktikum/SWE/src/opengl/visualization.cpp

# 7.53 WavePropagation Class Reference

#include <WavePropagation.h>

90 Class Documentation

#### **Public Member Functions**

- WavePropagation (T \*h, T \*hu, T \*b, unsigned int size, T cellSize)
- T computeNumericalFluxes ()
- void updateUnknowns (T dt)
- void setOutflowBoundaryConditions ()

# 7.53.1 Detailed Description

Allocated variables: unknowns h,hu are defined on grid indices [0,...,n+1] (done by the caller) -> computational domain is [1,...,nx] -> plus ghost cell layer

net-updates are defined for edges with indices [0,..,n]

A left/right net update with index (i-1) is located on the edge between cells with index (i-1) and (i):

```
* (i-1) * (i) *
```

# 7.53.2 Constructor & Destructor Documentation

7.53.2.1 WavePropagation::WavePropagation ( T\*h, T\*hu, T\*hu, T\*b, unsigned int size, T cellSize ) [inline]

#### **Parameters**

b	elevation of the ocean floor
size	Domain size (= number of cells) without ghost cells
cellSize	Size of one cell

#### 7.53.3 Member Function Documentation

7.53.3.1 T WavePropagation::computeNumericalFluxes ( )

Computes the net-updates from the unknowns

# Returns

The maximum possible time step

7.53.3.2 void WavePropagation::setOutflowBoundaryConditions ( )

Updates h and hu according to the outflow condition to both boundaries

7.53.3.3 void WavePropagation::updateUnknowns ( T dt )

Update the unknowns with the already computed net-updates

**Parameters** 

dt | Time step size

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

- /home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/WavePropagation.h
- /home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/WavePropagation.cpp

# 7.54 writer::ConsoleWriter Class Reference

```
#include <ConsoleWriter.h>
```

#### **Public Member Functions**

- ConsoleWriter (std::ostream &ostream=std::cout)
- void write (const T \*h, const T \*hu, unsigned int size)

# 7.54.1 Detailed Description

A simple writer class, that writes h and hu to stdout (or another ostream)

# 7.54.2 Member Function Documentation

```
7.54.2.1 void writer::ConsoleWriter::write ( const T * h, const T * hu, unsigned int size ) [inline]
```

Writes all values (without boundary values) to the ostream

Parameters

size Number of cells (without boundary values)

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

/home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/writer/ConsoleWriter.h

# 7.55 writer::VtkWriter Class Reference

```
#include <VtkWriter.h>
```

### **Public Member Functions**

- VtkWriter (const std::string &basename="swe1d", const T cellSize=1)
- void write (const T time, const T \*h, const T \*hu, const T \*b, unsigned int size)

92 Class Documentation

# 7.55.1 Detailed Description

A writer class that generates vtk files

# 7.55.2 Member Function Documentation

7.55.2.1 void writer::VtkWriter::write ( const T time, const T \* h, const T \* hu, const T \* b, unsigned int size ) [inline]

Writes all values to vtk file

**Parameters** 

size Number of cells (without boundary values)

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

• /home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/writer/VtkWriter.h

# **Chapter 8**

# **File Documentation**

# 8.1 mainpage.txt File Reference

# 8.1.1 Detailed Description

This file is part of SWE.

Author

```
Michael Bader (bader AT in.tum.de, http://www5.in.tum.de/wiki/index.php/Univ.--Prof._Dr._Michael_Bader)
```

#### 8.1.2 LICENSE

SWE is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE. If not, see http://www.-gnu.org/licenses/.

#### 8.1.3 DESCRIPTION

Main section of the doxygen documentation.

# 8.2 /home/raphael/Programmieren/BPraktikum/SWE/src/blocks/cuda/SWE\_BlockCUD-A.cu File Reference

```
#include "SWE_BlockCUDA.hh"
#include "SWE_BlockCUDA_kernels.hh"
#include "tools/help.hh"
#include "tools/Logger.hh"
#include <cassert>
#include <cstdlib>
#include <cmath>
```

#### **Functions**

- void checkCUDAError (const char \*msg)
- void tryCUDA (cudaError\_t err, const char \*msg)

# 8.2.1 Detailed Description

This file is part of SWE.

#### **Author**

```
Michael Bader, Kaveh Rahnema, Tobias Schnabel
Sebastian Rettenberger (rettenbs AT in.tum.de, http://www5.in.tum.de/wiki/index.php/-Sebastian_Rettenberger,_M.Sc.)
```

#### 8.2.2 LICENSE

SWE is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE. If not, see <a href="http://www.-gnu.org/licenses/">http://www.-gnu.org/licenses/</a>.

### 8.2.3 DESCRIPTION

TODO

# 8.3 /home/raphael/Programmieren/BPraktikum/SWE/src/blocks/cuda/SWE\_BlockCUD-A.hh File Reference

```
#include "blocks/SWE_Block.hh"
#include "tools/help.hh"
#include <iostream>
#include <fstream>
#include <cuda_runtime.h>
```

#### Classes

class SWE\_BlockCUDA

#### **Functions**

- void checkCUDAError (const char \*msg)
- void tryCUDA (cudaError\_t err, const char \*msg)
- \_\_device\_\_ int getCellCoord (int x, int y, int ny)
- \_\_device\_\_ int getEdgeCoord (int x, int y, int ny)
- \_\_device\_\_ int getBathyCoord (int x, int y, int ny)

#### **Variables**

• const int TILE SIZE =16

### 8.3.1 Detailed Description

This file is part of SWE.

#### Author

```
Michael Bader, Kaveh Rahnema, Tobias Schnabel
Sebastian Rettenberger (rettenbs AT in.tum.de, http://www5.in.tum.de/wiki/index.php/-Sebastian_Rettenberger,_M.Sc.)
```

#### 8.3.2 LICENSE

SWE is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE. If not, see <a href="http://www.-gnu.org/licenses/">http://www.-gnu.org/licenses/</a>.

#### 8.3.3 DESCRIPTION

**TODO** 

### 8.3.4 Function Documentation

```
8.3.4.1 __device__ int getBathyCoord ( int x, int y, int ny ) [inline]
```

Return index of a specific element in the arrays of bathymetry source terms

#### **Parameters**

i,j	x- and y-coordinate of grid cell
ny	grid size in y-direction (without ghost layers)

```
8.3.4.2 __device__ int getCellCoord ( int x, int y, int ny ) [inline]
```

Return index of hd[i][j] in linearised array

#### **Parameters**

i,j	x- and y-coordinate of grid cell
ny	grid size in y-direction (without ghost layers)

```
8.3.4.3 __device__ int getEdgeCoord ( int x, int y, int ny ) [inline]
```

Return index of edge-data Fhd[i][j] or Ghd[i][j] in linearised array

#### **Parameters**

i,j	x- and y-coordinate of grid cell
ny	grid size in y-direction (without ghost layers)

# 8.4 /home/raphael/Programmieren/BPraktikum/SWE/src/blocks/cuda/SWE\_BlockCUDA\_-kernels.cu File Reference

```
#include "SWE_BlockCUDA.hh"
#include "SWE_BlockCUDA_kernels.hh"
```

#### **Functions**

- $\bullet \ \underline{\hspace{0.3cm}} global\underline{\hspace{0.3cm}} void \ kernelHdBufferEdges \ (float *hd, int nx, int ny)$
- \_\_global\_\_ void kernelLeftBoundary (float \*hd, float \*hud, float \*hvd, int nx, int ny, BoundaryType bound)
- \_\_global\_\_ void kernelRightBoundary (float \*hd, float \*hud, float \*hvd, int nx, int ny, BoundaryType bound)
- \_\_global\_\_ void kernelBottomBoundary (float \*hd, float \*hud, float \*hvd, int nx, int ny, BoundaryType bound)
- \_\_global\_\_ void kernelTopBoundary (float \*hd, float \*hud, float \*hvd, int nx, int ny, BoundaryType bound)
- \_\_global\_\_ void kernelBottomGhostBoundary (float \*hd, float \*hud, float \*hvd, float \*bottomGhostLayer, int nx, int ny)
- \_\_global\_\_ void kernelTopGhostBoundary (float \*hd, float \*hud, float \*hvd, float \*topGhostLayer, int nx, int ny)
- \_\_global\_\_ void kernelBottomCopyLayer (float \*hd, float \*hvd, float \*hvd, float \*bottomCopyLayer, int nx, int ny)
- \_\_global\_\_ void kernelTopCopyLayer (float \*hd, float \*hd, float \*hvd, float \*topCopyLayer, int nx, int ny)

#### 8.4.1 Detailed Description

This file is part of SWE.

#### **Author**

```
Michael Bader (bader AT in.tum.de, http://www5.in.tum.de/wiki/index.php/Univ.--Prof._Dr._Michael_Bader)
```

#### 8.4.2 LICENSE

SWE is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE. If not, see <a href="http://www.-gnu.org/licenses/">http://www.-gnu.org/licenses/</a>.

#### 8.4.3 DESCRIPTION

**TODO** 

8.4.4 Function Documentation

8.4.4.1 \_\_global\_\_ void kernelBottomBoundary ( float \* hd, float \* hud, float \* hvd, int nx, int ny, BoundaryType bound )

CUDA kernel to set bottom boundary layer for conditions WALL & OUTFLOW blockldx.x and threadldx.x loop over the boundary elements SWE Block size ny is assumed to be a multiple of the TILE SIZE

8.4.4.2 global void kernelBottomCopyLayer (float \* hd, float \* hvd, float \* hvd, float \* bottomCopyLayer, int nx, int ny)

CUDA kernel to update bottom copy layer according (for boundary conditions PASSIVE and CONNECT) blockldx.x and threadldx.x loop over the boundary elements. Note that diagonal elements are currently not copied! SWE\_Block size ny is assumed to be a multiple of the TILE\_SIZE

8.4.4.3 \_\_global\_\_ void kernelBottomGhostBoundary ( float \* hd, float \* hvd, float \* hvd, float \* bottomGhostLayer, int nx, int ny )

CUDA kernel to set bottom boundary layer according to the external ghost layer status (conditions PASSIVE and CONNECT) blockldx.x and threadldx.x loop over the boundary elements. Note that diagonal elements are currently not copied! SWE\_Block size ny is assumed to be a multiple of the TILE\_SIZE

8.4.4.4 \_\_global\_\_ void kernelHdBufferEdges ( float \* hd, int nx, int ny )

Sets corner values of hd (only needed for visualization)

**Parameters** 

hd h-values on device

8.4.4.5 \_\_global\_\_ void kernelLeftBoundary ( float \* hd, float \* hud, float \* hvd, int nx, int ny, BoundaryType bound )

CUDA kernel to set left boundary layer for conditions WALL & OUTFLOW blockldx.y and threadldx.y loop over the boundary elements SWE Block size ny is assumed to be a multiple of the TILE SIZE

8.4.4.6 \_\_global\_\_ void kernelRightBoundary ( float \* hd, float \* hvd, float \* hvd, int nx, int ny, BoundaryType bound )

CUDA kernel to set right boundary layer for conditions WALL & OUTFLOW blockldx.y and threadldx.y loop over the boundary elements SWE\_Block size ny is assumed to be a multiple of the TILE\_SIZE

8.4.4.7 \_\_global\_\_ void kernelTopBoundary ( float \* hd, float \* hud, float \* hvd, int nx, int ny, BoundaryType bound )

CUDA kernel to set bottom boundary layer for conditions WALL & OUTFLOW blockldx.x and threadldx.x loop over the boundary elements

8.4.4.8 global void kernelTopCopyLayer (float \* hd, float \* hud, float \* hvd, float \* topCopyLayer, int nx, int ny)

CUDA kernel to set top boundary layer according to the external ghost layer status (conditions PASSIVE and CONNECT) blockldx.x and threadldx.x loop over the boundary elements Note that diagonal elements are currently not copied! SWE Block size ny is assumed to be a multiple of the TILE SIZE

```
8.4.4.9 <u>global</u> void kernelTopGhostBoundary ( float * hd, float * hud, float * hvd, float * topGhostLayer, int nx, int ny )
```

CUDA kernel to set top boundary layer according to the external ghost layer status (conditions PASSIVE and CO-NNECT) blockldx.x and threadldx.x loop over the boundary elements Note that diagonal elements are currently not copied! SWE Block size ny is assumed to be a multiple of the TILE SIZE

# 8.5 /home/raphael/Programmieren/BPraktikum/SWE/src/blocks/cuda/SWE\_BlockCUDA\_-kernels.hh File Reference

#### **Functions**

- global void kernelHdBufferEdges (float \*hd, int nx, int ny)
- \_\_global\_\_ void kernelMaximum (float \*maxhd, float \*maxvd, int start, int size)
- \_\_global\_\_ void kernelLeftBoundary (float \*hd, float \*hud, float \*hvd, int nx, int ny, BoundaryType bound)
- global void kernelRightBoundary (float \*hd, float \*hvd, int nx, int ny, BoundaryType bound)
- global void kernelBottomBoundary (float \*hd, float \*hud, float \*hvd, int nx, int ny, BoundaryType bound)
- \_\_global\_\_ void kernelTopBoundary (float \*hd, float \*hud, float \*hvd, int nx, int ny, BoundaryType bound)
- \_\_global\_\_ void kernelBottomGhostBoundary (float \*hd, float \*hud, float \*hvd, float \*bottomGhostLayer, int nx, int ny)
- \_\_global\_\_ void kernelTopGhostBoundary (float \*hd, float \*hud, float \*hvd, float \*topGhostLayer, int nx, int ny)
- \_\_global\_\_ void kernelBottomCopyLayer (float \*hd, float \*hvd, float \*hvd, float \*bottomCopyLayer, int nx, int ny)
- \_\_global\_\_ void kernelTopCopyLayer (float \*hd, float \*hud, float \*hvd, float \*topCopyLayer, int nx, int ny)

# 8.5.1 Detailed Description

This file is part of SWE.

#### Author

```
Michael Bader (bader AT in.tum.de, http://www5.in.tum.de/wiki/index.php/Univ.--Prof._Dr._Michael_Bader)
```

### 8.5.2 LICENSE

SWE is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE. If not, see <a href="http://www.-gnu.org/licenses/">http://www.-gnu.org/licenses/</a>.

#### 8.5.3 DESCRIPTION

TODO

8.5.4 Function Documentation

```
8.5.4.1 __global__ void kernelBottomBoundary ( float * hd, float * hud, float * hvd, int nx, int ny, BoundaryType bound )
```

CUDA kernel to set bottom boundary layer for conditions WALL & OUTFLOW blockldx.x and threadldx.x loop over the boundary elements SWE Block size ny is assumed to be a multiple of the TILE SIZE

```
8.5.4.2 global void kernelBottomCopyLayer (float * hd, float * hvd, float * hvd, float * bottomCopyLayer, int nx, int ny)
```

CUDA kernel to update bottom copy layer according (for boundary conditions PASSIVE and CONNECT) blockldx.x and threadldx.x loop over the boundary elements. Note that diagonal elements are currently not copied! SWE\_Block size ny is assumed to be a multiple of the TILE\_SIZE

```
8.5.4.3 __global__ void kernelBottomGhostBoundary ( float * hd, float * hvd, float * hvd, float * bottomGhostLayer, int nx, int ny )
```

CUDA kernel to set bottom boundary layer according to the external ghost layer status (conditions PASSIVE and CONNECT) blockldx.x and threadldx.x loop over the boundary elements. Note that diagonal elements are currently not copied! SWE\_Block size ny is assumed to be a multiple of the TILE\_SIZE

```
8.5.4.4 global void kernelHdBufferEdges (float * hd, int nx, int ny)
```

Sets corner values of hd (only needed for visualization)

**Parameters** 

```
hd h-values on device
```

```
8.5.4.5 __global__ void kernelLeftBoundary ( float * hd, float * hud, float * hvd, int nx, int ny, BoundaryType bound )
```

CUDA kernel to set left boundary layer for conditions WALL & OUTFLOW blockldx.y and threadldx.y loop over the boundary elements SWE Block size ny is assumed to be a multiple of the TILE SIZE

```
8.5.4.6 __global__ void kernelMaximum ( float * maxhd, float * maxvd, int start, int size )
```

CUDA kernel for maximum reduction required to compute maximum water height and velocities to determine allow time step

```
8.5.4.7 __global__ void kernelRightBoundary ( float * hd, float * hud, float * hvd, int nx, int ny, BoundaryType bound )
```

CUDA kernel to set right boundary layer for conditions WALL & OUTFLOW blockldx.y and threadldx.y loop over the boundary elements SWE\_Block size ny is assumed to be a multiple of the TILE\_SIZE

```
8.5.4.8 __global__ void kernelTopBoundary ( float * hd, float * hud, float * hvd, int nx, int ny, BoundaryType bound )
```

CUDA kernel to set bottom boundary layer for conditions WALL & OUTFLOW blockldx.x and threadldx.x loop over the boundary elements

```
8.5.4.9 __global__ void kernelTopCopyLayer ( float * hd, float * hud, float * hvd, float * topCopyLayer, int nx, int ny )
```

CUDA kernel to set top boundary layer according to the external ghost layer status (conditions PASSIVE and CO-NNECT) blockldx.x and threadldx.x loop over the boundary elements Note that diagonal elements are currently not

copied! SWE\_Block size ny is assumed to be a multiple of the TILE\_SIZE

```
8.5.4.10 __global__ void kernelTopGhostBoundary ( float * hd, float * hvd, float * hvd, float * topGhostLayer, int nx, int ny )
```

CUDA kernel to set top boundary layer according to the external ghost layer status (conditions PASSIVE and CONNECT) blockldx.x and threadldx.x loop over the boundary elements Note that diagonal elements are currently not copied! SWE\_Block size ny is assumed to be a multiple of the TILE\_SIZE

# 8.6 /home/raphael/Programmieren/BPraktikum/SWE/src/blocks/cuda/SWE\_WavePropagation-BlockCuda.cu File Reference

```
#include "SWE_WavePropagationBlockCuda.hh"
#include "SWE_BlockCUDA.hh"
#include "SWE_WavePropagationBlockCuda_kernels.hh"
#include "tools/Logger.hh"

#include <cassert>
#include <cuda.h>
#include <cuda_runtime_api.h>
#include <thrust/device_vector.h>
```

### 8.6.1 Detailed Description

This file is part of SWE.

# Author

```
Alexander Breuer (breuera AT in.tum.de, http://www5.in.tum.de/wiki/index.php/Dipl.--Math._Alexander_Breuer)

Sebastian Rettenberger (rettenbs AT in.tum.de, http://www5.in.tum.de/wiki/index.php/-Sebastian_Rettenberger,_M.Sc.)
```

### 8.6.2 LICENSE

SWE is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE. If not, see <a href="http://www.-gnu.org/licenses/">http://www.-gnu.org/licenses/</a>.

#### 8.6.3 DESCRIPTION

SWE Block in CUDA, which uses solvers in the wave propagation formulation.

# 8.7 /home/raphael/Programmieren/BPraktikum/SWE/src/blocks/cuda/SWE\_WavePropagation-BlockCuda.hh File Reference

```
#include <cassert>
#include "SWE_BlockCUDA.hh"
```

#### Classes

· class SWE WavePropagationBlockCuda

# 8.7.1 Detailed Description

This file is part of SWE.

#### Author

```
Alexander Breuer (breuera AT in.tum.de, http://www5.in.tum.de/wiki/index.php/Dipl.--Math._Alexander_Breuer)

Sebastian Rettenberger (rettenbs AT in.tum.de, http://www5.in.tum.de/wiki/index.php/-Sebastian_Rettenberger,_M.Sc.)
```

#### 8.7.2 LICENSE

SWE is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE. If not, see <a href="http://www.-gnu.org/licenses/">http://www.-gnu.org/licenses/</a>.

### 8.7.3 DESCRIPTION

SWE\_Block in CUDA, which uses solvers in the wave propagation formulation.

# 8.8 /home/raphael/Programmieren/BPraktikum/SWE/src/blocks/cuda/SWE\_WavePropagation-BlockCuda\_kernels.cu File Reference

```
#include "SWE_BlockCUDA.hh"
#include "SWE_WavePropagationBlockCuda_kernels.hh"
#include <cmath>
#include <cstdio>
#include "solvers/FWaveCuda.h"
```

#### **Functions**

• \_\_global\_\_ void computeNetUpdatesKernel (const float \*i\_h, const float \*i\_hu, const float \*i\_hv, const float \*i\_b, float \*o\_hNetUpdatesLeftD, float \*o\_hNetUpdatesRightD, float \*o\_huNetUpdatesLeftD, float \*o\_huNetUpdatesLeftD

UpdatesRightD, float  $*o_hNetUpdatesBelowD$ , float  $*o_hNetUpdatesBelowD$ , float  $*o_hvNetUpdatesBelowD$ , float  $*o_hvNetUpdatesAboveD$ , float  $*o_maximumWaveSpeeds$ , const int  $i_nX$ , const int  $i_nY$ , const

- \_\_global\_\_ void updateUnknownsKernel (const float \*i\_hNetUpdatesLeftD, const float \*i\_hNetUpdatesRightD, const float \*i\_huNetUpdatesLeftD, const float \*i\_huNetUpdatesRightD, const float \*i\_hNetUpdatesBelowD, const float \*i\_hNetUpdatesAboveD, const float \*i\_hvNetUpdatesBelowD, const float \*i\_hvNetUpdatesAboveD, float \*io\_h, float \*io\_hu, float \*io\_hv, const float i\_updateWidthX, const float i\_updateWidthY, const int i\_nX, const int i\_nY)
- \_\_device\_\_ int computeOneDPositionKernel (const int i\_i, const int i\_j, const int i\_ny)

#### 8.8.1 Detailed Description

This file is part of SWE.

**Author** 

```
Alexander Breuer (breuera AT in.tum.de, http://www5.in.tum.de/wiki/index.php/Dipl.--Math._Alexander_Breuer)

Sebastian Rettenberger (rettenbs AT in.tum.de, http://www5.in.tum.de/wiki/index.php/-Sebastian_Rettenberger,_M.Sc.)
```

#### 8.8.2 LICENSE

SWE is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE. If not, see <a href="http://www.-gnu.org/licenses/">http://www.-gnu.org/licenses/</a>.

#### 8.8.3 DESCRIPTION

CUDA Kernels for a SWE\_Block, which uses solvers in the wave propagation formulation.

#### 8.8.4 Function Documentation

8.8.4.1 \_\_global\_\_ void computeNetUpdatesKernel ( const float \* i\_h, const float \* i\_hu, const float \* i\_hu, const float \* i\_h, float \* o\_hNetUpdatesLeftD, float \* o\_hNetUpdatesRightD, float \* o\_huNetUpdatesLeftD, float \* o\_huNetUpdatesRightD, float \* o\_hNetUpdatesBelowD, float \* o\_hNetUpdatesAboveD, float \* o\_hvNetUpdatesAboveD, float \* o\_hvNetUpdatesAboveD, float \* o\_maximumWaveSpeeds, const int i\_nX, const int i\_nY, const int i\_offsetX, const int i\_offsetY, const int i\_blockOffSetY )

The compute net-updates kernel calls the solver for a defined CUDA-Block and does a reduction over the computed wave speeds within this block.

Remark: In overall we have nx+1 / ny+1 edges. Therefore the edges "simulation domain"/"top ghost layer" and "simulation domain"/"right ghost layer" will not be computed in a typical call of the function: computeNetUpdates-Kernel<<<dimGrid,dimBlock>>>(hd, hud, hvd, bd, hNetUpdatesLeftD, hNetUpdatesRightD, huNetUpdatesBelowD, hNetUpdatesAboveD, hvNetUpdatesBelowD, hvNetUpdatesAboveD, l\_maximumWaveSpeedsD, i\_nx, i\_ny); To reduce the effect of branch-mispredictions the kernel provides optional offsets, which can be used to compute the missing edges.

 $SWE\_Wave Propagation Block Cuda:: compute Numerical Fluxes () explains the coalesced memory access.$ 

Reference 103
Parameters

i_h	water heights (CUDA-array).
i_hu	momentums in x-direction (CUDA-array).
i_hv	momentums in y-direction (CUDA-array).
i_b	bathymetry values (CUDA-array).
o_hNetUpdates-	left going net-updates for the water height (CUDA-array).
LeftD	
o_hNetUpdates-	right going net-updates for the water height (CUDA-array).
RightD	
o_huNet-	left going net-updates for the momentum in x-direction (CUDA-array).
UpdatesLeftD	
o_huNet-	right going net-updates for the momentum in x-direction (CUDA-array).
UpdatesRightD	
o_hNetUpdates-	downwards going net-updates for the water height (CUDA-array).
BelowD	
o_hNetUpdates-	upwards going net-updates for the water height (CUDA-array).
AboveD	
o_hvNet-	downwards going net-updates for the momentum in y-direction (CUDA-array).
UpdatesBelowD	
o_hvNet-	upwards going net-updates for the momentum in y-direction (CUDA-array).
UpdatesAboveD	
o_maximum-	maximum wave speed which occurred within the CUDA-block is written here (CUDA-array).
WaveSpeeds	
i_nx	number of cells within the simulation domain in x-direction (excludes ghost layers).
i_ny	number of cells within the simulation domain in y-direction (excludes ghost layers).
i_offsetX	cell/edge offset in x-direction.
i_offsetY	cell/edge offset in y-direction.

array maximum wave speed within this CUDA-block

thread local index in the shared maximum wave speed array

index (I\_cellIndexI,I\_cellIndexJ) of the cell lying on the right side of the edge/above the edge where the thread works on.

array which holds the thread local net-updates.

location of the thread local cells in the global CUDA-arrays.

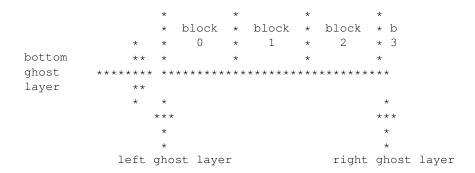
reduction partner for a thread

Position of the maximum wave speed in the global device array.

In the 'main' part (e.g.  $gridDim = nx/TILE\_SIZEm ny/TILE\_SIZE$ ) the position is simply given by the blockld in x-and y-direction with a stride of gridDim.x + 1. The +1 results from the speeds in the 'boundary' case, see below.

In the 'boundary' case, where the edges lie between the computational domain and the right/top ghost layer, this is more complicated. In this case block offsets in x- and y-direction are used. The offsets define how many blocks in the resp. direction have to be added to get a valid result. Computational domain - right ghost layer: In this case the dimension of the grid in x-direction is 1. Computational domain - top ghost layer: In this case the dimension of the grid in y-direction is 1.

Same Example as in SWE\_WavePropagationBlockCuda::computeNumericalFluxes(), assume the CUDA-grid/blocks has the following layout:



This results in a 'main' part containing of (3\*2) blocks and two 'boundary' parts containing of (1\*2) blocks and (3\*1) blocks.

The maximum wave speed array on the device represents therefore logically a (4 \* 3)-1 2D-array (-1: no block on the top right). The 'main' part writes into cells 0, 1, 2, 4, 5 and 6. The 'computational domain - right ghost layer' part writes into 3 and 7 with offset in x-direction = 3 The 'computational domain - top ghost layer' part writes into 8, 9, 10 with offset in y-direction = 2

**8.8.4.2** \_\_device\_\_ int computeOneDPositionKernel ( const int *i\_i*, const int *i\_j*, const int *i\_ny* ) [inline]

Compute the position of 2D coordinates in a 1D array. array[i][j] -> i \* ny + j

#### **Parameters**

<u>i_i</u>	row index.
i_j	column index.
i_ny	#(cells in y-direction).

#### Returns

1D index.

8.8.4.3 \_\_global\_\_ void updateUnknownsKernel ( const float \* i\_hNetUpdatesLeftD, const float \* i\_hNetUpdatesRightD, const float \* i\_hNetUpdatesLeftD, const float \* i\_hNetUpdatesBelowD, const float \* i\_hNetUpdatesAboveD, const float \* i\_hNetUpdatesAboveD, const float \* i\_hNetUpdatesAboveD, float \* io\_h, float \* io\_hu, float \* io\_hv, const float i\_updateWidthX, const float i\_updateWidthY, const int i\_nX, const i\_nX, cons

The "update unknowns"-kernel updates the unknowns in the cells with precomputed net-updates.

SWE WavePropagationBlockCuda::computeNumericalFluxes() explains the coalesced memory access.

### **Parameters**

i_hNetUpdates-	left going net-updates for the water height (CUDA-array).
LeftD	
i_hNetUpdates-	right going net-updates for the water height (CUDA-array).
RightD	
i_huNetUpdates-	left going net-updates for the momentum in x-direction (CUDA-array).
LeftD	
i_huNetUpdates-	right going net-updates for the momentum in x-direction (CUDA-array).
RightD	
i_hNetUpdates-	downwards going net-updates for the water height (CUDA-array).
BelowD	

# 8.9 /home/raphael/Programmieren/BPraktikum/SWE/src/blocks/cuda/SWE\_WavePropagationBlockCuda\_kernels.hh File

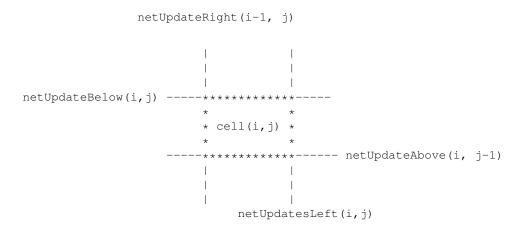
Reference	
i_hNetUpdates-	upwards going net-updates for the water height (CUDA-array).
AboveD	
i_hvNetUpdates-	downwards going net-updates for the momentum in y-direction (CUDA-array).
BelowD	
i_hvNetUpdates-	upwards going net-updates for the momentum in y-direction (CUDA-array).
AboveD	
io_h	water heights (CUDA-array).
io_hu	momentums in x-direction (CUDA-array).
io_hv	momentums in y-direction (CUDA-array).
i_updateWidthX	update width in x-direction.
i_updateWidthY	update width in y-direction.
i_nx	number of cells within the simulation domain in x-direction (excludes ghost layers).
i_ny	number of cells within the simulation domain in y-direction (excludes ghost layers).

cell indices (I cellIndexI,I cellIndexJ) of the cell which the thread updates.

location of the thread local cell in the global CUDA-arrays.

positions of the net-updates in the global CUDA-arrays.

Compute the positions of the net updates relative to a given cell



# 8.9 /home/raphael/Programmieren/BPraktikum/SWE/src/blocks/cuda/SWE\_WavePropagation-BlockCuda kernels.hh File Reference

### **Functions**

- \_\_global\_\_ void computeNetUpdatesKernel (const float \*i\_h, const float \*i\_hu, const float \*i\_hv, const float \*i\_b, float \*o\_hNetUpdatesLeftD, float \*o\_hNetUpdatesRightD, float \*o\_huNetUpdatesLeftD, float \*o\_huNetUpdatesRightD, float \*o\_hvNetUpdatesBelowD, float \*o\_hvNetUpdatesBelowD, float \*o\_hvNetUpdatesAboveD, float \*o\_hvNetUpdatesAboveD, float \*o\_maximumWaveSpeeds, const int i\_nx, const int i\_ny, const int i\_offsetX=0, const int i\_offsetY=0)
- \_\_global\_\_ void updateUnknownsKernel (const float \*i\_hNetUpdatesLeftD, const float \*i\_hNetUpdatesRightD, const float \*i\_huNetUpdatesBelowD, const float \*i\_hNetUpdatesBelowD, const float \*i\_hNetUpdatesAboveD, const float \*i\_hNetUpdatesAboveD, const float \*i\_hNetUpdatesAboveD, float \*io\_h, float \*io\_hu, float \*io\_hv, const float i\_updateWidthX, const float i\_updateWidthY, const int i nx, const int i ny)
- \_\_device\_\_ int computeOneDPositionKernel (const int i\_i, const int i\_j, const int i\_nx)

# 8.9.1 Detailed Description

This file is part of SWE.

#### **Author**

Alexander Breuer (breuera AT in.tum.de, http://www5.in.tum.de/wiki/index.php/Dipl.--Math.\_Alexander\_Breuer)

#### 8.9.2 LICENSE

SWE is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE. If not, see http://www.-gnu.org/licenses/.

#### 8.9.3 DESCRIPTION

CUDA Kernels for a SWE Block, which uses solvers in the wave propagation formulation.

#### 8.9.4 Function Documentation

8.9.4.1 \_\_global\_\_ void computeNetUpdatesKernel ( const float \* i\_h, const float \* i\_hu, const float \* i\_hu, const float \* i\_h, float \* o\_hNetUpdatesLeftD, float \* o\_hNetUpdatesRightD, float \* o\_huNetUpdatesLeftD, float \* o\_huNetUpdatesRightD, float \* o\_hNetUpdatesBelowD, float \* o\_hNetUpdatesAboveD, float \* o\_hvNetUpdatesAboveD, float \* o\_hvNetUpdatesAboveD, float \* o\_maximumWaveSpeeds, const int i\_nX, const int i\_nY, const int i\_offsetX, const int i\_offsetY, const int i\_blockOffSetY )

The compute net-updates kernel calls the solver for a defined CUDA-Block and does a reduction over the computed wave speeds within this block.

Remark: In overall we have nx+1 / ny+1 edges. Therefore the edges "simulation domain"/"top ghost layer" and "simulation domain"/"right ghost layer" will not be computed in a typical call of the function: computeNetUpdates-Kernel<<<dimGrid,dimBlock>>>(hd, hud, hvd, bd, hNetUpdatesLeftD, hNetUpdatesRightD, huNetUpdatesBelowD, hNetUpdatesAboveD, hvNetUpdatesBelowD, hvNetUpdatesAboveD, l\_maximumWaveSpeedsD, i\_nx, i\_ny); To reduce the effect of branch-mispredictions the kernel provides optional offsets, which can be used to compute the missing edges.

SWE\_WavePropagationBlockCuda::computeNumericalFluxes() explains the coalesced memory access.

#### **Parameters**

i_h	water heights (CUDA-array).
i_hu	momentums in x-direction (CUDA-array).
<u>i_</u> hv	momentums in y-direction (CUDA-array).
i_b	bathymetry values (CUDA-array).
o_hNetUpdates-	left going net-updates for the water height (CUDA-array).
LeftD	
o_hNetUpdates-	right going net-updates for the water height (CUDA-array).
RightD	
o_huNet-	left going net-updates for the momentum in x-direction (CUDA-array).
UpdatesLeftD	

# 8.9 /home/raphael/Programmieren/BPraktikum/SWE/src/blocks/cuda/SWE\_WavePropagationBlockCuda\_kernels.hh File

Reference	
o_huNet-	right going net-updates for the momentum in x-direction (CUDA-array).
UpdatesRightD	
o_hNetUpdates-	downwards going net-updates for the water height (CUDA-array).
BelowD	
o_hNetUpdates-	upwards going net-updates for the water height (CUDA-array).
AboveD	
o_hvNet-	downwards going net-updates for the momentum in y-direction (CUDA-array).
UpdatesBelowD	
o_hvNet-	upwards going net-updates for the momentum in y-direction (CUDA-array).
UpdatesAboveD	
o_maximum-	maximum wave speed which occurred within the CUDA-block is written here (CUDA-array).
WaveSpeeds	
i_nx	number of cells within the simulation domain in x-direction (excludes ghost layers).
i_ny	number of cells within the simulation domain in y-direction (excludes ghost layers).
i_offsetX	cell/edge offset in x-direction.
i_offsetY	cell/edge offset in y-direction.

array maximum wave speed within this CUDA-block

thread local index in the shared maximum wave speed array

index (I\_cellIndexI,I\_cellIndexJ) of the cell lying on the right side of the edge/above the edge where the thread works on.

array which holds the thread local net-updates.

location of the thread local cells in the global CUDA-arrays.

reduction partner for a thread

Position of the maximum wave speed in the global device array.

In the 'main' part (e.g.  $gridDim = nx/TILE\_SIZEm ny/TILE\_SIZE$ ) the position is simply given by the blockld in x- and y-direction with a stride of gridDim.x + 1. The +1 results from the speeds in the 'boundary' case, see below.

In the 'boundary' case, where the edges lie between the computational domain and the right/top ghost layer, this is more complicated. In this case block offsets in x- and y-direction are used. The offsets define how many blocks in the resp. direction have to be added to get a valid result. Computational domain - right ghost layer: In this case the dimension of the grid in x-direction is 1. Computational domain - top ghost layer: In this case the dimension of the grid in y-direction is 1.

Same Example as in SWE\_WavePropagationBlockCuda::computeNumericalFluxes(), assume the CUDA-grid/blocks has the following layout:

```
top ghost layer,

* block 8 * block 9 * block 10 * ********

* block 8 * block 9 * block 10 * *******

* block * block * block * b

* 4 * 5 * 6 * 7

* * * * * * * *

* block * block * block * b

* * block * block * block * b

* * * * * * * *

* block * block * block * block * b

* * * * * * *

* block * block * block * block * b

* * * * * * *

* block * block * block * block * b

* * * * * * *

* block * * block * block * block * b

* * * * * * *

* block * * block * block
```

This results in a 'main' part containing of (3\*2) blocks and two 'boundary' parts containing of (1\*2) blocks and (3\*1) blocks.

The maximum wave speed array on the device represents therefore logically a (4 \* 3)-1 2D-array (-1: no block on the top right). The 'main' part writes into cells 0, 1, 2, 4, 5 and 6. The 'computational domain - right ghost layer' part writes into 3 and 7 with offset in x-direction = 3 The 'computational domain - top ghost layer' part writes into 8, 9, 10 with offset in y-direction = 2

**8.9.4.2** \_\_device\_\_ int computeOneDPositionKernel ( const int *i\_i*, const int *i\_j*, const int *i\_ny* ) [inline]

Compute the position of 2D coordinates in a 1D array. array[i][j] -> i \* ny + j

#### **Parameters**

<u>i_</u> i	row index.
i_j	column index.
i_ny	#(cells in y-direction).

#### Returns

1D index.

8.9.4.3 \_\_global\_\_ void updateUnknownsKernel ( const float \* i\_hNetUpdatesLeftD, const float \* i\_hNetUpdatesRightD, const float \* i\_hNetUpdatesBelowD, const float \* i\_hNetUpdatesBelowD, const float \* i\_hNetUpdatesAboveD, const float \* i\_hNetUpdatesAboveD, float \* io\_h, float \* io\_hu, float \* io\_hu, const float i\_updateWidthX, const float i\_updateWidthY, const int i\_nX, const int i\_nY)

The "update unknowns"-kernel updates the unknowns in the cells with precomputed net-updates.

SWE\_WavePropagationBlockCuda::computeNumericalFluxes() explains the coalesced memory access.

#### **Parameters**

i_hNetUpdates- LeftD	left going net-updates for the water height (CUDA-array).
i_hNetUpdates-	right going net-updates for the water height (CUDA-array).
RightD	
i_huNetUpdates-	left going net-updates for the momentum in x-direction (CUDA-array).
LeftD	
i_huNetUpdates-	right going net-updates for the momentum in x-direction (CUDA-array).
RightD	
i_hNetUpdates-	downwards going net-updates for the water height (CUDA-array).
BelowD	
i_hNetUpdates-	upwards going net-updates for the water height (CUDA-array).
AboveD	
i_hvNetUpdates-	downwards going net-updates for the momentum in y-direction (CUDA-array).
BelowD	
i_hvNetUpdates-	upwards going net-updates for the momentum in y-direction (CUDA-array).
AboveD	
io_h	water heights (CUDA-array).
io_hu	momentums in x-direction (CUDA-array).
io_hv	momentums in y-direction (CUDA-array).
i_updateWidthX	update width in x-direction.

•	$\Delta$	

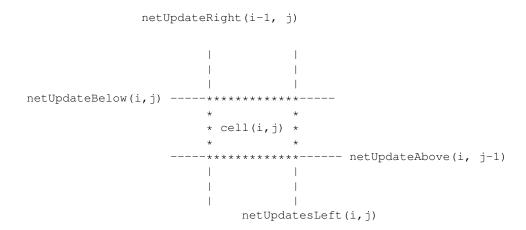
i_updateWidthY	update width in y-direction.
i_nx	number of cells within the simulation domain in x-direction (excludes ghost layers).
i_ny	number of cells within the simulation domain in y-direction (excludes ghost layers).

cell indices (I\_cellIndexI,I\_cellIndexJ) of the cell which the thread updates.

location of the thread local cell in the global CUDA-arrays.

positions of the net-updates in the global CUDA-arrays.

Compute the positions of the net updates relative to a given cell



# 8.10 /home/raphael/Programmieren/BPraktikum/SWE/src/blocks/rusanov/SWE\_Rusanov-Block.cpp File Reference

```
#include "SWE_RusanovBlock.hh"
#include <math.h>
```

#### **Functions**

ostream & operator<< (ostream &os, const SWE\_RusanovBlock &swe)</li>

# 8.10.1 Detailed Description

This file is part of SWE.

**Author** 

Michael Bader, Kaveh Rahnema, Tobias Schnabel

### 8.10.2 LICENSE

SWE is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE. If not, see <a href="http://www.-gnu.org/licenses/">http://www.-gnu.org/licenses/</a>.

#### 8.10.3 DESCRIPTION

**TODO** 

#### 8.10.4 Function Documentation

```
8.10.4.1 ostream & os, const SWE_RusanovBlock & swe )
```

overload operator << such that data can be written via cout << -> needs to be declared as friend to be allowed to access private data

# 8.11 /home/raphael/Programmieren/BPraktikum/SWE/src/blocks/rusanov/SWE\_Rusanov-Block.hh File Reference

```
#include <iostream>
#include <stdio.h>
#include <fstream>
#include "tools/help.hh"
#include "SWE_Block.hh"
```

#### Classes

· class SWE RusanovBlock

#### **Functions**

• ostream & operator << (ostream &os, const SWE\_RusanovBlock &swe)

# 8.11.1 Detailed Description

This file is part of SWE.

**Author** 

Michael Bader, Kaveh Rahnema, Tobias Schnabel

# 8.11.2 LICENSE

SWE is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE. If not, see http://www.-gnu.org/licenses/.

# 8.11.3 DESCRIPTION

**TODO** 

#### 8.11.4 Function Documentation

```
8.11.4.1 ostream& operator << ( ostream & os, const SWE_RusanovBlock & swe )
```

overload operator << such that data can be written via cout << -> needs to be declared as friend to be allowed to access private data

# 8.12 /home/raphael/Programmieren/BPraktikum/SWE/src/blocks/rusanov/SWE\_Rusanov-BlockCUDA.cu File Reference

```
#include <math.h>
#include "tools/help.hh"
#include "SWE_BlockCUDA.hh"
#include "SWE_RusanovBlockCUDA.hh"
#include "SWE_RusanovBlockCUDA_kernels.hh"
```

#### **Functions**

ostream & operator<< (ostream &os, const SWE\_RusanovBlockCUDA &swe)</li>

# 8.12.1 Detailed Description

This file is part of SWE.

**Author** 

Michael Bader, Kaveh Rahnema, Tobias Schnabel

### **8.12.2 LICENSE**

SWE is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE. If not, see <a href="http://www.-gnu.org/licenses/">http://www.-gnu.org/licenses/</a>.

#### 8.12.3 DESCRIPTION

TODO

# 8.12.4 Function Documentation

```
8.12.4.1 ostream& operator << ( ostream & os, const SWE_RusanovBlockCUDA & swe )
```

overload operator << such that data can be written via cout << -> needs to be declared as friend to be allowed to access private data

# 8.13 /home/raphael/Programmieren/BPraktikum/SWE/src/blocks/rusanov/SWE\_Rusanov-BlockCUDA.hh File Reference

```
#include <iostream>
#include <stdio.h>
#include <fstream>
#include <cuda_runtime.h>
#include "tools/help.hh"
#include "SWE_Block.hh"
#include "SWE_BlockCUDA.hh"
```

#### Classes

• class SWE\_RusanovBlockCUDA

#### **Functions**

ostream & operator<< (ostream &os, const SWE\_RusanovBlockCUDA &swe)</li>

# 8.13.1 Detailed Description

This file is part of SWE.

Author

Michael Bader, Kaveh Rahnema, Tobias Schnabel

### 8.13.2 LICENSE

SWE is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE. If not, see <a href="http://www.-gnu.org/licenses/">http://www.-gnu.org/licenses/</a>.

# 8.13.3 DESCRIPTION

**TODO** 

### 8.13.4 Function Documentation

8.13.4.1 ostream& operator<< ( ostream & os, const SWE\_RusanovBlockCUDA & swe )

overload operator << such that data can be written via cout << -> needs to be declared as friend to be allowed to access private data

/home/raphael/Programmieren/BPraktikum/SWE/src/blocks/rusanov/SWE\_RusanovBlockCUDA\_kernels.cu File Reference 113

# 8.14 /home/raphael/Programmieren/BPraktikum/SWE/src/blocks/rusanov/SWE\_Rusanov-BlockCUDA kernels.cu File Reference

```
#include "SWE_BlockCUDA.hh"
#include "SWE_RusanovBlockCUDA_kernels.hh"
```

#### **Functions**

- \_\_device\_\_ float computeFlux (float fLow, float fHigh, float xiLow, float xiHigh, float llf)
- \_\_global\_\_ void kernelComputeFluxesF (float \*hd, float \*hud, float \*hvd, float \*Fhd, float \*Fhud, float \*Fhvd, int ny, float g, float llf, int istart)
- \_\_global\_\_ void kernelComputeFluxesG (float \*hd, float \*hud, float \*hvd, float \*Ghd, float \*Ghud, float \*Ghvd, int ny, float g, float llf, int jstart)
- \_\_global\_\_ void kernelComputeBathymetrySources (float \*hd, float \*bd, float \*Bxd, float \*Byd, int ny, float g)
- \_\_global\_\_ void kernelEulerTimestep (float \*hd, float \*hud, float \*hvd, float \*Fhd, float \*Fhud, float \*Fhvd, float \*Ghd, float \*Ghud, float \*Ghvd, float \*Bxd, float \*Byd, float \*maxhd, float \*maxvd, int nx, int ny, float dt, float dxi, float dyi)
- \_\_global\_\_ void kernelMaximum (float \*maxhd, float \*maxvd, int start, int size)

#### 8.14.1 Detailed Description

This file is part of SWE.

**Author** 

```
Michael Bader (bader AT in.tum.de, http://www5.in.tum.de/wiki/index.php/Univ.--Prof. Dr. Michael Bader)
```

#### **8.14.2 LICENSE**

SWE is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE. If not, see <a href="http://www.-gnu.org/licenses/">http://www.-gnu.org/licenses/</a>.

# 8.14.3 DESCRIPTION

TODO

#### 8.14.4 Function Documentation

```
8.14.4.1 __global__ void kernelComputeBathymetrySources ( float * hd, float * bd, float * Bxd, float * Byd, int ny, float g )
```

computes the bathymetry source terms for the hu and hv equation for a given cell in the resp. array elements Bxd and Byd

8.14.4.2 \_\_global\_\_ void kernelComputeFluxesF ( float \* hd, float \* hud, float \* hvd, float \* Fhd, float \* Fhud, f

computes the flux vector components Fhd, Fhud and Fhvd for a single edge by calling the function computeFlux

8.14.4.3 \_\_global\_\_ void kernelComputeFluxesG ( float \* hd, float \* hud, float \* hvd, float \* Ghd, float \* Ghud, f

computes the flux vector components Ghd, Ghud and Ghvd for a single edge by calling the function computeFlux

8.14.4.4 \_\_global\_\_ void kernelEulerTimestep ( float \* hd, float \* hud, float \* hud, float \* Fhud, f

CUDA kernel for Euler time step

```
8.14.4.5 __global__ void kernelMaximum ( float * maxhd, float * maxvd, int start, int size )
```

CUDA kernel for maximum reduction required to compute maximum water height and velocities to determine allow time step

# 8.15 /home/raphael/Programmieren/BPraktikum/SWE/src/blocks/rusanov/SWE\_Rusanov-BlockCUDA kernels.hh File Reference

#### **Functions**

- \_\_global\_\_ void kernelComputeFluxesF (float \*hd, float \*hud, float \*hvd, float \*Fhd, float \*Fhud, float \*Fhvd, int ny, float g, float llf, int istart)
- \_\_global\_\_ void kernelComputeFluxesG (float \*hd, float \*hud, float \*hvd, float \*Ghd, float \*Ghud, float \*Ghvd, int ny, float g, float llf, int jstart)
- \_\_global\_\_ void kernelComputeBathymetrySources (float \*hd, float \*bd, float \*Bxd, float \*Byd, int ny, float g)
- \_\_global\_\_ void kernelEulerTimestep (float \*hd, float \*hud, float \*hvd, float \*Fhd, float \*Fhud, float \*Fhvd, float \*Ghd, float \*Ghud, float \*Bxd, float \*Byd, float \*maxhd, float \*maxvd, int nx, int ny, float dt, float dxi, float dyi)
- global void kernelMaximum (float \*maxhd, float \*maxvd, int start, int size)

#### 8.15.1 Detailed Description

This file is part of SWE.

**Author** 

Michael Bader, Kaveh Rahnema, Tobias Schnabel

#### 8.15.2 LICENSE

SWE is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE. If not, see http://www.-gnu.org/licenses/.

#### 8.15.3 DESCRIPTION

TODO

#### 8.15.4 Function Documentation

```
8.15.4.1 __global__ void kernelComputeBathymetrySources ( float * hd, float * bd, float * Bxd, float * Byd, int ny, float g )
```

computes the bathymetry source terms for the hu and hv equation for a given cell in the resp. array elements Bxd and Byd

```
8.15.4.2 __global__ void kernelComputeFluxesF ( float * hd, float * hud, float * hvd, float * Fhd, float * Fhud, float * Fhvd, int ny, float g, float llf, int istart )
```

computes the flux vector components Fhd, Fhud and Fhvd for a single edge by calling the function computeFlux

```
8.15.4.3 __global__ void kernelComputeFluxesG ( float * hd, float * hud, float * hvd, float * Ghd, float * Ghud, f
```

computes the flux vector components Ghd, Ghud and Ghvd for a single edge by calling the function computeFlux

```
8.15.4.4 __global__ void kernelEulerTimestep ( float * hd, float * hud, float * hvd, float * Fhd, float * Fhud, float * Fhvd, float * Ghd, float * Ghud, float * Ghvd, float * Bxd, float * Byd, float * maxhd, float * maxvd, int nx, int ny, float dt, float dxi, float dyi )
```

CUDA kernel for Euler time step

```
8.15.4.5 __global__ void kernelMaximum ( float * maxhd, float * maxvd, int start, int size )
```

CUDA kernel for maximum reduction required to compute maximum water height and velocities to determine allow time step

# 8.16 /home/raphael/Programmieren/BPraktikum/SWE/src/blocks/SWE\_Block.cpp File Reference

```
#include "SWE_Block.hh"
#include "tools/help.hh"
#include <cmath>
#include <iostream>
#include <cassert>
#include <limits>
```

#### 8.16.1 Detailed Description

This file is part of SWE.

#### Author

```
Michael Bader, Kaveh Rahnema, Tobias Schnabel
Sebastian Rettenberger (rettenbs AT in.tum.de, http://www5.in.tum.de/wiki/index.php/-Sebastian_Rettenberger,_M.Sc.)
```

#### 8.16.2 **LICENSE**

SWE is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE. If not, see <a href="http://www.-gnu.org/licenses/">http://www.-gnu.org/licenses/</a>.

#### 8.16.3 DESCRIPTION

TODO

# 8.17 /home/raphael/Programmieren/BPraktikum/SWE/src/blocks/SWE\_Block.hh File Reference

```
#include "tools/help.hh"
#include "scenarios/SWE_Scenario.hh"
#include <iostream>
#include <fstream>
```

#### **Classes**

- · class SWE Block
- struct SWE\_Block1D

#### **Variables**

• const int BLOCKS =4

# 8.17.1 Detailed Description

This file is part of SWE.

#### **Author**

```
Michael Bader, Kaveh Rahnema, Tobias Schnabel
Sebastian Rettenberger (rettenbs AT in.tum.de, http://www5.in.tum.de/wiki/index.php/-Sebastian_Rettenberger,_M.Sc.)
```

#### 8.17.2 LICENSE

SWE is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE. If not, see <a href="http://www.-gnu.org/licenses/">http://www.-gnu.org/licenses/</a>.

#### 8.17.3 DESCRIPTION

**TODO** 

# 8.18 /home/raphael/Programmieren/BPraktikum/SWE/src/blocks/SWE\_WavePropagation-Block.cpp File Reference

```
#include "SWE_WavePropagationBlock.hh"
#include <cassert>
#include <string>
#include <limits>
```

# 8.18.1 Detailed Description

This file is part of SWE.

#### Author

```
Alexander Breuer (breuera AT in.tum.de, http://www5.in.tum.de/wiki/index.php/Dipl.--Math._Alexander_Breuer)

Sebastian Rettenberger (rettenbs AT in.tum.de, http://www5.in.tum.de/wiki/index.php/-Sebastian_Rettenberger,_M.Sc.)
```

#### 8.18.2 LICENSE

SWE is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE. If not, see <a href="http://www.-gnu.org/licenses/">http://www.-gnu.org/licenses/</a>.

#### 8.18.3 DESCRIPTION

SWE\_Block, which uses solvers in the wave propagation formulation.

# 8.19 /home/raphael/Programmieren/BPraktikum/SWE/src/blocks/SWE\_WavePropagation-Block.hh File Reference

```
#include "blocks/SWE_Block.hh"
#include "tools/help.hh"
#include <string>
#include "solvers/Hybrid.hpp"
```

#### Classes

class SWE\_WavePropagationBlock

#### 8.19.1 Detailed Description

This file is part of SWE.

#### **Author**

```
Alexander Breuer (breuera AT in.tum.de, http://www5.in.tum.de/wiki/index.php/Dipl.--Math._Alexander_Breuer)

Sebastian Rettenberger (rettenbs AT in.tum.de, http://www5.in.tum.de/wiki/index.php/-Sebastian_Rettenberger,_M.Sc.)
```

# 8.19.2 LICENSE

SWE is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE. If not, see <a href="http://www.-gnu.org/licenses/">http://www.-gnu.org/licenses/</a>.

# 8.19.3 DESCRIPTION

SWE\_Block, which uses solvers in the wave propagation formulation.

# 8.20 /home/raphael/Programmieren/BPraktikum/SWE/src/examples/swe\_Dimensional-Splitting.cpp File Reference

```
#include <cassert>
#include <cstdlib>
#include <string>
#include "blocks/SWE_DimensionalSplitting.hh"
#include "writer/VtkWriter.hh"
#include "writer/NetCdfWriter.hh"
#include "scenarios/SWE_simple_scenarios.hh"
#include "scenarios/SWE_TsunamiScenario.hh"
#include "tools/help.hh"
#include "tools/Logger.hh"
#include "tools/ProgressBar.hh"
```

#### **Functions**

• int main (int argc, char \*\*argv)

### 8.20.1 Detailed Description

This file is part of SWE.

#### **Author**

```
Alexander Breuer (breuera AT in.tum.de, http://www5.in.tum.de/wiki/index.php/Dipl.--Math._Alexander_Breuer) Michael Bader (bader AT in.tum.de, http://www5.in.tum.-de/wiki/index.php/Univ.-Prof._Dr._Michael_Bader)
```

#### **8.20.2 LICENSE**

SWE is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE. If not, see <a href="http://www.-gnu.org/licenses/">http://www.-gnu.org/licenses/</a>.

#### 8.20.3 DESCRIPTION

Basic setting of SWE, which uses a wave propagation solver and an artificial or ASAGI scenario on a single block.

# 8.20.4 Function Documentation

```
8.20.4.1 int main ( int argc, char ** argv )
```

Main program for the simulation on a single SWE\_WavePropagationBlock. Initialization. number of grid cells in x- and y-direction.

I\_baseName of the plots.

number of checkpoints for visualization (at each checkpoint in time, an output file is written).

size of a single cell in x- and y-direction

origin of the simulation domain in x- and y-direction

time when the simulation ends.

checkpoints when output files are written.

Simulation.

simulation time.

Finalize.

# 8.21 /home/raphael/Programmieren/BPraktikum/SWE/src/examples/swe\_mpi.cpp File Reference

```
#include <algorithm>
#include <cassert>
#include <cmath>
#include <cstdlib>
#include <mpi.h>
#include <string>
#include <vector>
#include "blocks/SWE_WavePropagationBlock.hh"
#include "blocks/SWE_simple_scenarios.hh"
#include "tools/help.hh"
#include "tools/Logger.hh"
#include "tools/ProgressBar.hh"
```

### **Macros**

#define ARG(arg\_name) getArgByName(vargs, arg\_name, argv)

# **Functions**

- int computeNumberOfBlockRows (int i\_numberOfProcesses)
- void exchangeLeftRightGhostLayers (const int i\_leftNeighborRank, SWE\_Block1D \*o\_leftInflow, SWE\_Block1D \*o\_rightInflow, SWE\_Block1D \*i\_rightNeighborRank, SWE\_Block1D \*o\_rightInflow, SWE
- void exchangeBottomTopGhostLayers (const int i\_bottomNeighborRank, SWE\_Block1D \*o\_bottomNeighborInflow, SWE\_Block1D \*i\_bottomNeighborOutflow, const int i\_topNeighborRank, SWE\_Block1D \*o\_top-NeighborInflow, SWE\_Block1D \*i\_topNeighborOutflow, const MPI\_Datatype i\_mpiRow)
- int main (int argc, char \*\*argv)

#### 8.21.1 Detailed Description

This file is part of SWE.

Author

```
Michael Bader (bader AT in.tum.de, http://www5.in.tum.de/wiki/index.php/Univ.--Prof._Dr._Michael_Bader)
Alexander Breuer (breuera AT in.tum.de, http://www5.in.tum.de/wiki/index.php/Dipl.--Math._Alexander_Breuer)
Sebastian Rettenberger (rettenbs AT in.tum.de, http://www5.in.tum.de/wiki/index.php/-Sebastian_Rettenberger,_M.Sc.)
```

#### 8.21.2 LICENSE

SWE is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE. If not, see <a href="http://www.-gnu.org/licenses/">http://www.-gnu.org/licenses/</a>.

#### 8.21.3 DESCRIPTION

Setting of SWE, which uses a wave propagation solver and an artificial or ASAGI scenario on multiple blocks.

#### 8.21.4 Function Documentation

8.21.4.1 int computeNumberOfBlockRows ( int i\_numberOfProcesses )

Compute the number of block rows from the total number of processes.

The number of rows is determined as the square root of the number of processes, if this is a square number; otherwise, we use the largest number that is smaller than the square root and still a divisor of the number of processes.

**Parameters** 

numProcs	number of process.

#### Returns

number of block rows

8.21.4.2 void exchangeBottomTopGhostLayers ( const int i\_bottomNeighborRank, SWE\_Block1D \* o\_bottomNeighborInflow, SWE\_Block1D \* i\_bottomNeighborOutflow, const int i\_topNeighborRank, SWE\_Block1D \* o\_topNeighborInflow, SWE\_Block1D \* i\_topNeighborOutflow, const MPI\_Datatype i\_mpiRow )

Exchanges the bottom and top ghost layers with MPI's SendReceive.

# **Parameters**

i_bottom-	MPI rank of the bottom neighbor.
NeighborRank	

o_bottom-	ghost layer, where the bottom neighbor writes into.
NeighborInflow	
i_bottom-	host layer, where the bottom neighbor reads from.
NeighborOutflow	
i_topNeighbor-	MPI rank of the top neighbor.
Rank	
o_topNeighbor-	ghost layer, where the top neighbor writes into.
Inflow	
i_topNeighbor-	ghost layer, where the top neighbor reads from.
Outflow	
i_mpiRow	MPI data type for the horizontal ghost layers.

8.21.4.3 void exchangeLeftRightGhostLayers ( const int i\_leftNeighborRank, SWE\_Block1D \* o\_leftInflow, SWE\_Block1D \* i\_leftOutflow, const int i\_rightNeighborRank, SWE\_Block1D \* o\_rightInflow, SWE\_Block1D \* i\_rightOutflow, MPI\_Datatype i\_mpiCol )

Exchanges the left and right ghost layers with MPI's SendReceive.

#### **Parameters**

i_leftNeighbor-	MPI rank of the left neighbor.
Rank	
o_leftInflow	ghost layer, where the left neighbor writes into.
i_leftOutflow	layer where the left neighbor reads from.
i_rightNeighbor-	MPI rank of the right neighbor.
Rank	
o_rightInflow	ghost layer, where the right neighbor writes into.
i_rightOutflow	layer, where the right neighbor reads form.
i_mpiCol	MPI data type for the vertical gost layers.

#### 8.21.4.4 int main ( int argc, char \*\* argv )

Main program for the simulation on a single SWE\_WavePropagationBlock. Initialization.

MPI Rank of a process.

number of MPI processes.

total number of grid cell in x- and y-direction.

 $I\_baseName\ of\ the\ plots.$ 

number of SWE\_Blocks in x- and y-direction.

local position of each MPI process in x- and y-direction.

number of checkpoints for visualization (at each checkpoint in time, an output file is written).

number of grid cells in x- and y-direction per process.

size of a single cell in x- and y-direction

origin of the simulation domain in x- and y-direction

time when the simulation ends.

checkpoints when output files are written.

MPI row-vector: I\_nXLocal+2 blocks, 1 element per block, stride of I\_nYLocal+2

MPI row-vector: 1 block, I\_nYLocal+2 elements per block, stride of 1

MPI ranks of the neighbors

#### Simulation.

simulation time.

maximum allowed time step width within a block.

maximum allowed time steps of all blocks

Finalize.

# 8.22 /home/raphael/Programmieren/BPraktikum/SWE/src/examples/swe\_simple.cpp File Reference

```
#include <cassert>
#include <cstdlib>
#include <string>
#include <iostream>
#include <fstream>
#include "blocks/SWE_WavePropagationBlock.hh"
#include "writer/VtkWriter.hh"
#include "scenarios/SWE_simple_scenarios.hh"
#include "tools/help.hh"
#include "tools/Logger.hh"
#include "tools/ProgressBar.hh"
```

#### **Functions**

• int main (int argc, char \*\*argv)

## 8.22.1 Detailed Description

This file is part of SWE.

Author

```
Alexander Breuer (breuera AT in.tum.de, http://www5.in.tum.de/wiki/index.php/Dipl.--Math._Alexander_Breuer) Michael Bader (bader AT in.tum.de, http://www5.in.tum.-de/wiki/index.php/Univ.-Prof._Dr._Michael_Bader)
```

## **8.22.2 LICENSE**

SWE is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE. If not, see <a href="http://www.-gnu.org/licenses/">http://www.-gnu.org/licenses/</a>.

#### 8.22.3 DESCRIPTION

Basic setting of SWE, which uses a wave propagation solver and an artificial or ASAGI scenario on a single block.

#### 8.22.4 Function Documentation

```
8.22.4.1 int main ( int argc, char ** argv )
```

Main program for the simulation on a single SWE\_WavePropagationBlock. Initialization.

number of grid cells in x- and y-direction.

I baseName of the plots.

true if checkpoint file exists

number of checkpoints for visualization (at each checkpoint in time, an output file is written).

size of a single cell in x- and y-direction

origin of the simulation domain in x- and y-direction

time when the simulation ends.

checkpoints when output files are written.

Simulation.

simulation time.

number of checkpoints that are already passed

maximum allowed time step width.

Finalize.

## 8.23 /home/raphael/Programmieren/BPraktikum/SWE/src/opengl/vbo.cpp File Reference

```
#include "vbo.h"
#include "visualization.h"
```

## 8.23.1 Detailed Description

This file is part of SWE.

Author

```
Sebastian Rettenberger (rettenbs AT in.tum.de, http://www5.in.tum.de/wiki/index.php/-Sebastian_Rettenberger,_M.Sc.)
```

#### **8.23.2 LICENSE**

SWE is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE. If not, see <a href="http://www.-gnu.org/licenses/">http://www.-gnu.org/licenses/</a>.

## 8.24 /home/raphael/Programmieren/BPraktikum/SWE/src/opengl/vbo.h File Reference

```
#include "tools/Logger.hh"
```

#### Classes

• class VBO

#### 8.24.1 Detailed Description

This file is part of SWE.

#### **Author**

Sebastian Rettenberger (rettenbs AT in.tum.de, http://www5.in.tum.de/wiki/index.php/-Sebastian\_Rettenberger,\_M.Sc.)

## **8.24.2 LICENSE**

SWE is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE. If not, see <a href="http://www.-gnu.org/licenses/">http://www.-gnu.org/licenses/</a>.

### 8.24.3 DESCRIPTION

Handles a VertexBufferObject.

## 8.25 /home/raphael/Programmieren/BPraktikum/SWE/src/scenarios/SWE\_Artificial-TsunamiScenario.hh File Reference

#include <cmath>

## Classes

· class SWE ArtificialTsunamiScenario

### **Macros**

• #define PI 3.1415926535897932384626433832795

## 8.25.1 Detailed Description

This file is part of SWE.

**Author** 

Raphael Dümig

#### 8.25.2 LICENSE

SWE is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE. If not, see <a href="http://www.-gnu.org/licenses/">http://www.-gnu.org/licenses/</a>.

#### 8.25.3 DESCRIPTION

TODO

# 8.26 /home/raphael/Programmieren/BPraktikum/SWE/src/scenarios/SWE\_AsagiScenario.cpp File Reference

```
#include "SWE_AsagiScenario.hh"
```

### 8.26.1 Detailed Description

This file is part of SWE.

Author

Sebastian Rettenberger (rettenbs AT in.tum.de, http://www5.in.tum.de/wiki/index.php/-Sebastian\_Rettenberger,\_M.Sc.)

#### 8.26.2 LICENSE

SWE is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE. If not, see <a href="http://www.-gnu.org/licenses/">http://www.-gnu.org/licenses/</a>.

# 8.27 /home/raphael/Programmieren/BPraktikum/SWE/src/scenarios/SWE\_AsagiScenario.hh

#include <cassert>

```
#include <cstring>
#include <string>
#include <iostream>
#include <map>
#include <asagi.h>
#include "SWE_Scenario.hh"
```

#### **Classes**

- · class SWE AsagiGrid
- class SWE\_AsagiScenario

## 8.27.1 Detailed Description

This file is part of SWE.

#### **Author**

```
Alexander Breuer (breuera AT in.tum.de, http://www5.in.tum.de/wiki/index.php/Dipl.--Math._Alexander_Breuer)

Sebastian Rettenberger (rettenbs AT in.tum.de, http://www5.in.tum.de/wiki/index.php/-Sebastian_Rettenberger,_M.Sc.)
```

## 8.27.2 LICENSE

SWE is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE. If not, see <a href="http://www.-gnu.org/licenses/">http://www.-gnu.org/licenses/</a>.

## 8.27.3 DESCRIPTION

Access to bathymetry and displacement files with ASAGI.

## 8.28 /home/raphael/Programmieren/BPraktikum/SWE/src/scenarios/SWE\_AsagiScenario-\_vis.hh File Reference

```
#include "SWE_VisInfo.hh"
```

### Classes

class SWE\_AsagiJapanSmallVisInfo

#### 8.28.1 Detailed Description

This file is part of SWE.

**Author** 

```
Sebastian Rettenberger (rettenbs AT in.tum.de, http://www5.in.tum.de/wiki/index.php/-Sebastian_Rettenberger,_M.Sc.)
```

#### 8.28.2 LICENSE

SWE is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE. If not, see http://www.-qnu.org/licenses/.

#### 8.28.3 DESCRIPTION

Rescale water height in small Japan scenario

## 8.29 /home/raphael/Programmieren/BPraktikum/SWE/src/scenarios/SWE\_NetCDFCheckpoint-Scenario.hh File Reference

```
#include "SWE_NetCDFScenario.hh"
#include <netcdf.h>
#include <iostream>
#include <cstdlib>
```

## Classes

• class SWE\_NetCDFCheckpointScenario

#### 8.29.1 Detailed Description

This file is part of SWE.

Author

Thomas Blocher, Raphael Dümig

#### 8.29.2 LICENSE

SWE is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE. If not, see http://www.-gnu.org/licenses/.

#### 8.29.3 DESCRIPTION

TODO

# 8.30 /home/raphael/Programmieren/BPraktikum/SWE/src/scenarios/SWE\_Scenario.hh File Reference

#### Classes

· class SWE Scenario

### **Typedefs**

- typedef enum BoundaryType BoundaryType
- typedef enum BoundaryEdge BoundaryEdge

#### **Enumerations**

enum BoundaryType {
 OUTFLOW, WALL, INFLOW, CONNECT,
 PASSIVE }

enum BoundaryEdge { BND LEFT, BND RIGHT, BND BOTTOM, BND TOP }

## 8.30.1 Detailed Description

This file is part of SWE.

**Author** 

Michael Bader, Kaveh Rahnema, Tobias Schnabel

## 8.30.2 LICENSE

SWE is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE. If not, see <a href="http://www.-gnu.org/licenses/">http://www.-gnu.org/licenses/</a>.

#### 8.30.3 DESCRIPTION

**TODO** 

## 8.30.4 Typedef Documentation

8.30.4.1 typedef enum BoundaryEdge BoundaryEdge

enum type: numbering of the boundary edges

8.30.4.2 typedef enum BoundaryType BoundaryType

enum type: available types of boundary conditions

8.30.5 Enumeration Type Documentation

8.30.5.1 enum BoundaryEdge

enum type: numbering of the boundary edges

8.30.5.2 enum BoundaryType

enum type: available types of boundary conditions

# 8.31 /home/raphael/Programmieren/BPraktikum/SWE/src/scenarios/SWE\_simple\_scenarios.hh

```
#include <cmath>
#include "SWE_Scenario.hh"
```

## Classes

- class SWE\_RadialDamBreakScenario
- · class SWE\_BathymetryDamBreakScenario
- class SWE\_SeaAtRestScenario
- class SWE\_SplashingPoolScenario
- class SWE\_SplashingConeScenario
- class SWE\_DamBreakScenario

#### 8.31.1 Detailed Description

This file is part of SWE.

Author

```
Michael Bader, Kaveh Rahnema, Tobias Schnabel
Sebastian Rettenberger (rettenbs AT in.tum.de, http://www5.in.tum.de/wiki/index.php/-Sebastian_Rettenberger,_M.Sc.)
```

#### 8.31.2 LICENSE

SWE is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE. If not, see <a href="http://www.-gnu.org/licenses/">http://www.-gnu.org/licenses/</a>.

#### 8.31.3 DESCRIPTION

TODO

## 8.32 /home/raphael/Programmieren/BPraktikum/SWE/src/scenarios/SWE\_Tsunami-Scenario.hh File Reference

```
#include "SWE_NetCDFScenario.hh"
#include <netcdf.h>
#include <iostream>
#include <cstdlib>
#include <cassert>
```

#### Classes

· class SWE\_TsunamiScenario

#### 8.32.1 Detailed Description

This file is part of SWE.

Author

Thomas Blocher, Raphael Dümig

#### **8.32.2 LICENSE**

SWE is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE. If not, see <a href="http://www.-gnu.org/licenses/">http://www.-gnu.org/licenses/</a>.

### 8.32.3 DESCRIPTION

TODO

# 8.33 /home/raphael/Programmieren/BPraktikum/SWE/src/scenarios/SWE\_VisInfo.hh File Reference

```
#include "SWE_Scenario.hh"
```

### Classes

· class SWE VisInfo

## 8.33.1 Detailed Description

This file is part of SWE.

#### Author

Michael Bader
Kaveh Rahnema
Tobias Schnabel
Sebastian Rettenberger (rettenbs AT in.tum.de, http://www5.in.tum.de/wiki/index.php/-Sebastian\_Rettenberger,\_M.Sc.)

#### 8.33.2 LICENSE

SWE is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE. If not, see <a href="http://www.-gnu.org/licenses/">http://www.-gnu.org/licenses/</a>.

## 8.33.3 DESCRIPTION

**TODO** 

# 8.34 /home/raphael/Programmieren/BPraktikum/SWE/src/testing/testing\_scenario.hh

```
#include "scenarios/SWE_Scenario.hh"
```

## Classes

• class SWE\_TestingScenario

## 8.34.1 Detailed Description

This file is part of SWE.

**Author** 

Raphael Dümig duemig@in.tum.de

#### 8.34.2 LICENSE

SWE is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE. If not, see <a href="http://www.-gnu.org/licenses/">http://www.-gnu.org/licenses/</a>.

#### 8.34.3 DESCRIPTION

**TODO** 

## 8.35 /home/raphael/Programmieren/BPraktikum/SWE/src/tools/help.hh File Reference

```
#include <cstring>
#include <iostream>
#include <fstream>
#include <sstream>
```

#### **Classes**

- class Float1D
- · class Float2D

## **Functions**

- std::string generateFileName (std::string baseName, int timeStep)
- std::string generateFileName (std::string i\_baseName, int i\_blockPositionX, int i\_blockPositionY, std::string i\_fileExtension=".nc")
- std::string generateFileName (std::string baseName, int timeStep, int block\_X, int block\_Y, std::string i\_file-Extension=".vts")
- std::string generateBaseFileName (std::string &i\_baseName, int i\_blockPositionX, int i\_blockPositionY)
- std::string generateContainerFileName (std::string baseName, int timeStep)

## 8.35.1 Detailed Description

This file is part of SWE.

### Author

Michael Bader, Kaveh Rahnema Sebastian Rettenberger

## 8.35.2 LICENSE

SWE is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE. If not, see <a href="http://www.-gnu.org/licenses/">http://www.-gnu.org/licenses/</a>.

#### 8.35.3 DESCRIPTION

**TODO** 

## 8.35.4 Function Documentation

8.35.4.1 std::string generateBaseFileName ( std::string & *i\_baseName*, int *i\_blockPositionX*, int *i\_blockPositionY* ) [inline]

Generates an output file name for a multiple SWE\_Block version based on the ordering of the blocks.

#### **Parameters**

i_baseName	base name of the output.
i_blockPositionX	position of the SWE_Block in x-direction.
i_blockPositionY	position of the SWE_Block in y-direction.

### Returns

the output filename without timestep information and file extension

```
8.35.4.2 std::string generateContainerFileName ( std::string baseName, int timeStep ) [inline]
```

generate output filename for the ParaView-Container-File (to visualize multiple SWE\_Blocks per checkpoint)

```
8.35.4.3 std::string generateFileName ( std::string baseName, int timeStep ) [inline]
```

generate output filenames for the single-SWE\_Block version (for serial and OpenMP-parallelised versions that use only a single SWE\_Block - one output file is generated per checkpoint)

### **Deprecated**

8.35.4.4 std::string generateFileName ( std::string i\_baseName, int i\_blockPositionX, int i\_blockPositionY, std::string i\_fileExtension = " . nc" ) [inline]

Generates an output file name for a multiple SWE Block version based on the ordering of the blocks.

**Parameters** 

i_baseName	base name of the output.
i_blockPositionX	position of the SWE_Block in x-direction.
i_blockPositionY	position of the SWE_Block in y-direction.
i_fileExtension	file extension of the output file.

Returns

## **Deprecated**

```
8.35.4.5 std::string generateFileName ( std::string baseName, int timeStep, int block_X, int block_Y, std::string i_fileExtension = ".vts") [inline]
```

generate output filename for the multiple-SWE\_Block version (for serial and parallel (OpenMP and MPI) versions that use multiple SWE\_Blocks - for each block, one output file is generated per checkpoint)

### **Deprecated**

# 8.36 /home/raphael/Programmieren/BPraktikum/SWE/src/tools/Logger.cpp File Reference

```
#include "Logger.hh"
```

## 8.36.1 Detailed Description

This file is part of SWE.

Author

```
Sebastian Rettenberger (rettenbs AT in.tum.de, http://www5.in.tum.de/wiki/index.php/-Sebastian_Rettenberger,_M.Sc.)
```

## 8.36.2 LICENSE

SWE is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE. If not, see <a href="http://www.-gnu.org/licenses/">http://www.-gnu.org/licenses/</a>.

## 8.37 /home/raphael/Programmieren/BPraktikum/SWE/src/tools/Logger.hh File Reference

```
#include <string>
#include <iostream>
#include <ctime>
```

#### Classes

· class tools::Logger

#### 8.37.1 Detailed Description

This file is part of SWE.

#### **Author**

```
Alexander Breuer (breuera AT in.tum.de, http://www5.in.tum.de/wiki/index.php/Dipl.--Math._Alexander_Breuer)

Sebastian Rettenberger (rettenbs AT in.tum.de, http://www5.in.tum.de/wiki/index.php/-Sebastian_Rettenberger,_M.Sc.)
```

#### 8.37.2 LICENSE

SWE is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE. If not, see <a href="http://www.-gnu.org/licenses/">http://www.-gnu.org/licenses/</a>.

## 8.37.3 DESCRIPTION

Collection of basic logging routines.

# 8.38 /home/raphael/Programmieren/BPraktikum/SWE/src/tools/ProgressBar.hh File Reference

```
#include <cassert>
#include <cmath>
#include <ctime>
#include <algorithm>
#include <iostream>
#include <limits>
#include <unistd.h>
#include <sys/ioctl.h>
```

## Classes

· class tools::ProgressBar

## 8.38.1 Detailed Description

This file is part of SWE.

Author

Sebastian Rettenberger (rettenbs AT in.tum.de, http://www5.in.tum.de/wiki/index.php/-Sebastian\_Rettenberger,\_M.Sc.)

#### 8.38.2 LICENSE

SWE is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE. If not, see http://www.-qnu.org/licenses/.

## 8.38.3 DESCRIPTION

A simple progress bar using stdout

# 8.39 /home/raphael/Programmieren/BPraktikum/SWE/src/writer/NetCdfWriter.cpp File Reference

```
#include "NetCdfWriter.hh"
#include <string>
#include <vector>
#include <iostream>
#include <cassert>
```

#### 8.39.1 Detailed Description

This file is part of SWE.

Author

```
Alexander Breuer (breuera AT in.tum.de, http://www5.in.tum.de/wiki/index.php/Dipl.--Math._Alexander_Breuer)

Sebastian Rettenberger (rettenbs AT in.tum.de, http://www5.in.tum.de/wiki/index.php/-Sebastian_Rettenberger,_M.Sc.)
```

#### 8.39.2 LICENSE

SWE is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE. If not, see <a href="http://www.-gnu.org/licenses/">http://www.-gnu.org/licenses/</a>.

#### 8.39.3 DESCRIPTION

A writer for the netCDF-format: http://www.unidata.ucar.edu/software/netcdf/

## 8.40 /home/raphael/Programmieren/BPraktikum/SWE/src/writer/NetCdfWriter.hh File Reference

```
#include <cstring>
#include <string>
#include <vector>
#include <netcdf.h>
#include "writer/Writer.hh"
```

#### Classes

· class io::NetCdfWriter

## 8.40.1 Detailed Description

This file is part of SWE.

#### **Author**

```
Alexander Breuer (breuera AT in.tum.de, http://www5.in.tum.de/wiki/index.php/Dipl.--Math._Alexander_Breuer)

Sebastian Rettenberger (rettenbs AT in.tum.de, http://www5.in.tum.de/wiki/index.php/-Sebastian_Rettenberger,_M.Sc.)
```

## 8.40.2 LICENSE

SWE is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE. If not, see <a href="http://www.-gnu.org/licenses/">http://www.-gnu.org/licenses/</a>.

## 8.40.3 DESCRIPTION

A writer for the netCDF-format: http://www.unidata.ucar.edu/software/netcdf/

## 8.41 /home/raphael/Programmieren/BPraktikum/SWE/src/writer/NetCdfWriterCP.cpp File Reference

```
#include "NetCdfWriterCP.hh"
#include <string>
#include <vector>
#include <iostream>
#include <cassert>
```

## 8.41.1 Detailed Description

This file is part of SWE.

#### Author

```
Alexander Breuer (breuera AT in.tum.de, http://www5.in.tum.de/wiki/index.php/Dipl.--Math._Alexander_Breuer)

Sebastian Rettenberger (rettenbs AT in.tum.de, http://www5.in.tum.de/wiki/index.php/-Sebastian_Rettenberger,_M.Sc.)
```

#### 8.41.2 LICENSE

SWE is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE. If not, see http://www.-gnu.org/licenses/.

## 8.41.3 DESCRIPTION

A writer for the netCDF-format: http://www.unidata.ucar.edu/software/netcdf/

## 8.42 /home/raphael/Programmieren/BPraktikum/SWE/src/writer/NetCdfWriterCP.hh File Reference

```
#include <cstring>
#include <string>
#include <vector>
#include <netcdf.h>
#include "writer/Writer.hh"
```

#### **Classes**

· class io::NetCdfWriter

#### 8.42.1 Detailed Description

This file is part of SWE.

**Author** 

```
Alexander Breuer (breuera AT in.tum.de, http://www5.in.tum.de/wiki/index.php/Dipl.--Math._Alexander_Breuer)

Sebastian Rettenberger (rettenbs AT in.tum.de, http://www5.in.tum.de/wiki/index.php/-Sebastian_Rettenberger,_M.Sc.)
```

#### **8.42.2 LICENSE**

SWE is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE. If not, see <a href="http://www.-gnu.org/licenses/">http://www.-gnu.org/licenses/</a>.

#### 8.42.3 DESCRIPTION

A writer for the netCDF-format: http://www.unidata.ucar.edu/software/netcdf/

# 8.43 /home/raphael/Programmieren/BPraktikum/SWE/src/writer/VtkWriter.cpp File Reference

```
#include <cassert>
#include <fstream>
#include "VtkWriter.hh"
```

#### 8.43.1 Detailed Description

This file is part of SWE.

Author

```
Sebastian Rettenberger (rettenbs AT in.tum.de, http://www5.in.tum.de/wiki/index.php/-Sebastian_Rettenberger,_M.Sc.)
```

### 8.43.2 LICENSE

SWE is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE. If not, see http://www.-gnu.org/licenses/.

## 8.43.3 DESCRIPTION

# 8.44 /home/raphael/Programmieren/BPraktikum/SWE/src/writer/VtkWriter.hh File Reference

```
#include <sstream>
#include "writer/Writer.hh"
```

### Classes

· class io::VtkWriter

## 8.44.1 Detailed Description

This file is part of SWE.

**Author** 

Sebastian Rettenberger (rettenbs AT in.tum.de, http://www5.in.tum.de/wiki/index.php/-Sebastian\_Rettenberger,\_M.Sc.)

#### 8.44.2 LICENSE

SWE is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE. If not, see <a href="http://www.-gnu.org/licenses/">http://www.-gnu.org/licenses/</a>.

#### 8.44.3 DESCRIPTION

## 8.45 /home/raphael/Programmieren/BPraktikum/SWE/src/writer/Writer.hh File Reference

```
#include "tools/help.hh"
```

#### Classes

- struct io::BoundarySize
- · class io::Writer

#### 8.45.1 Detailed Description

This file is part of SWE.

Author

Sebastian Rettenberger (rettenbs AT in.tum.de, http://www5.in.tum.de/wiki/index.php/-Sebastian\_Rettenberger,\_M.Sc.)

## 8.45.2 LICENSE

SWE is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE. If not, see <a href="http://www.-gnu.org/licenses/">http://www.-gnu.org/licenses/</a>.

### 8.45.3 DESCRIPTION

## 8.46 /home/raphael/Programmieren/BPraktikum/SWE/submodules/f-wave-solver/src/F-Wave.cpp File Reference

```
#include "FWave.hpp"
#include <cmath>
#include <cassert>
```

## 8.46.1 Detailed Description

Implementation of an f-wave solver

**Author** 

Raphael Dümig

## 8.47 /home/raphael/Programmieren/BPraktikum/SWE/submodules/f-wave-solver/src/F-Wave.hpp File Reference

```
#include "FWave.cpp"
```

#### Classes

class solver::FWave< T >

## **Macros**

• #define **G** 9.81

## 8.47.1 Detailed Description

f-wave solver

**Author** 

Raphael Dümig

## 8.48 /home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/main.cpp File Reference

```
#include "types.h"
#include "WavePropagation.h"
#include "scenarios/eisbach.h"
#include "writer/VtkWriter.h"
#include "tools/args.h"
#include <cstring>
```

## **Functions**

• int main (int argc, char \*\*argv)

## 8.48.1 Detailed Description

This file is part of SWE1D

SWE1D is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE1D is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE1D. If not, see http-://www.gnu.org/licenses/.

Diese Datei ist Teil von SWE1D.

SWE1D ist Freie Software: Sie koennen es unter den Bedingungen der GNU General Public License, wie von der Free Software Foundation, Version 3 der Lizenz oder (nach Ihrer Option) jeder spaeteren veroeffentlichten Version, weiterverbreiten und/oder modifizieren.

SWE1D wird in der Hoffnung, dass es nuetzlich sein wird, aber OHNE JEDE GEWAEHELEISTUNG, bereitgestellt; sogar ohne die implizite Gewaehrleistung der MARKTFAEHIGKEIT oder EIGNUNG FUER EINEN BESTIMMTEN ZWECK. Siehe die GNU General Public License fuer weitere Details.

Sie sollten eine Kopie der GNU General Public License zusammen mit diesem Programm erhalten haben. Wenn nicht, siehe http://www.gnu.org/licenses/.

#### Copyright

2013 Technische Universitaet Muenchen

#### **Author**

Sebastian Rettenberger rettenbs@in.tum.de

# 8.49 /home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/scenarios/constant\_flow.h File Reference

```
#include "types.h"
```

#### Classes

· class scenarios::ConstantFlow

## 8.49.1 Detailed Description

This file is part of SWE1D

SWE1D is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE1D is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE1D. If not, see http-://www.gnu.org/licenses/.

Diese Datei ist Teil von SWE1D.

SWE1D ist Freie Software: Sie koennen es unter den Bedingungen der GNU General Public License, wie von der Free Software Foundation, Version 3 der Lizenz oder (nach Ihrer Option) jeder spaeteren veroeffentlichten Version, weiterverbreiten und/oder modifizieren.

SWE1D wird in der Hoffnung, dass es nuetzlich sein wird, aber OHNE JEDE GEWAEHELEISTUNG, bereitgestellt; sogar ohne die implizite Gewaehrleistung der MARKTFAEHIGKEIT oder EIGNUNG FUER EINEN BESTIMMTEN ZWECK. Siehe die GNU General Public License fuer weitere Details.

Sie sollten eine Kopie der GNU General Public License zusammen mit diesem Programm erhalten haben. Wenn nicht, siehe http://www.gnu.org/licenses/.

## Copyright

2013 Technische Universitaet Muenchen

### **Author**

```
Sebastian Rettenberger rettenbs@in.tum.de Raphael Dümig duemig@in.tum.de
```

# 8.50 /home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/scenarios/dambreak.h File Reference

```
#include "types.h"
```

### Classes

· class scenarios::DamBreak

## 8.50.1 Detailed Description

This file is part of SWE1D

SWE1D is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

## /home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/scenarios/dambreak\_bathy.h File Reference

SWE1D is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE1D. If not, see http-://www.gnu.org/licenses/.

Diese Datei ist Teil von SWE1D.

SWE1D ist Freie Software: Sie koennen es unter den Bedingungen der GNU General Public License, wie von der Free Software Foundation, Version 3 der Lizenz oder (nach Ihrer Option) jeder spaeteren veroeffentlichten Version, weiterverbreiten und/oder modifizieren.

SWE1D wird in der Hoffnung, dass es nuetzlich sein wird, aber OHNE JEDE GEWAEHELEISTUNG, bereitgestellt; sogar ohne die implizite Gewaehrleistung der MARKTFAEHIGKEIT oder EIGNUNG FUER EINEN BESTIMMTEN ZWECK. Siehe die GNU General Public License fuer weitere Details.

Sie sollten eine Kopie der GNU General Public License zusammen mit diesem Programm erhalten haben. Wenn nicht, siehe http://www.gnu.org/licenses/.

#### Copyright

2013 Technische Universitaet Muenchen

#### **Author**

```
Sebastian Rettenberger rettenbs@in.tum.de
Raphael Dümig duemig@in.tum.de
```

# 8.51 /home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/scenarios/dambreak-bathy.h File Reference

```
#include "types.h"
```

#### **Classes**

· class scenarios::DamBreak

#### 8.51.1 Detailed Description

This file is part of SWE1D

SWE1D is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE1D is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE1D. If not, see http-://www.gnu.org/licenses/.

Diese Datei ist Teil von SWE1D.

SWE1D ist Freie Software: Sie koennen es unter den Bedingungen der GNU General Public License, wie von der Free Software Foundation, Version 3 der Lizenz oder (nach Ihrer Option) jeder spaeteren veroeffentlichten Version, weiterverbreiten und/oder modifizieren.

SWE1D wird in der Hoffnung, dass es nuetzlich sein wird, aber OHNE JEDE GEWAEHELEISTUNG, bereitgestellt; sogar ohne die implizite Gewaehrleistung der MARKTFAEHIGKEIT oder EIGNUNG FUER EINEN BESTIMMTEN ZWECK. Siehe die GNU General Public License fuer weitere Details.

Sie sollten eine Kopie der GNU General Public License zusammen mit diesem Programm erhalten haben. Wenn nicht, siehe http://www.gnu.org/licenses/.

Copyright

2013 Technische Universitaet Muenchen

**Author** 

```
Sebastian Rettenberger rettenbs@in.tum.de Raphael Dümig duemig@in.tum.de
```

# 8.52 /home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/scenarios/eisbach.h

```
#include "types.h"
```

#### Classes

· class scenarios::Eisbach

#### 8.52.1 Detailed Description

This file is part of SWE1D

SWE1D is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE1D is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE1D. If not, see http-://www.gnu.org/licenses/.

Diese Datei ist Teil von SWE1D.

SWE1D ist Freie Software: Sie koennen es unter den Bedingungen der GNU General Public License, wie von der Free Software Foundation, Version 3 der Lizenz oder (nach Ihrer Option) jeder spaeteren veroeffentlichten Version, weiterverbreiten und/oder modifizieren.

SWE1D wird in der Hoffnung, dass es nuetzlich sein wird, aber OHNE JEDE GEWAEHELEISTUNG, bereitgestellt; sogar ohne die implizite Gewaehrleistung der MARKTFAEHIGKEIT oder EIGNUNG FUER EINEN BESTIMMTEN ZWECK. Siehe die GNU General Public License fuer weitere Details.

Sie sollten eine Kopie der GNU General Public License zusammen mit diesem Programm erhalten haben. Wenn nicht, siehe http://www.gnu.org/licenses/.

Copyright

2013 Technische Universitaet Muenchen

### Author

```
Sebastian Rettenberger rettenbs@in.tum.de Raphael Dümig duemig@in.tum.de
```

## 8.53 /home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/scenarios/rarerare.h

#include "types.h"

#### Classes

· class scenarios::RareRare

## 8.53.1 Detailed Description

This file is part of SWE1D

SWE1D is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE1D is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE1D. If not, see http-://www.gnu.org/licenses/.

Diese Datei ist Teil von SWE1D.

SWE1D ist Freie Software: Sie koennen es unter den Bedingungen der GNU General Public License, wie von der Free Software Foundation, Version 3 der Lizenz oder (nach Ihrer Option) jeder spaeteren veroeffentlichten Version, weiterverbreiten und/oder modifizieren.

SWE1D wird in der Hoffnung, dass es nuetzlich sein wird, aber OHNE JEDE GEWAEHELEISTUNG, bereitgestellt; sogar ohne die implizite Gewaehrleistung der MARKTFAEHIGKEIT oder EIGNUNG FUER EINEN BESTIMMTEN ZWECK. Siehe die GNU General Public License fuer weitere Details.

Sie sollten eine Kopie der GNU General Public License zusammen mit diesem Programm erhalten haben. Wenn nicht, siehe http://www.gnu.org/licenses/.

### Copyright

2013 Technische Universitaet Muenchen

Author

Raphael Dümig duemig@in.tum.de

# 8.54 /home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/scenarios/-ShockShock.h File Reference

```
#include "types.h"
```

#### Classes

· class scenarios::ShockShock

## 8.54.1 Detailed Description

This file is part of SWE1D

SWE1D is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE1D is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE1D. If not, see http-://www.gnu.org/licenses/.

Diese Datei ist Teil von SWE1D.

SWE1D ist Freie Software: Sie koennen es unter den Bedingungen der GNU General Public License, wie von der Free Software Foundation, Version 3 der Lizenz oder (nach Ihrer Option) jeder spaeteren veroeffentlichten Version, weiterverbreiten und/oder modifizieren.

SWE1D wird in der Hoffnung, dass es nuetzlich sein wird, aber OHNE JEDE GEWAEHELEISTUNG, bereitgestellt; sogar ohne die implizite Gewaehrleistung der MARKTFAEHIGKEIT oder EIGNUNG FUER EINEN BESTIMMTEN ZWECK. Siehe die GNU General Public License fuer weitere Details.

Sie sollten eine Kopie der GNU General Public License zusammen mit diesem Programm erhalten haben. Wenn nicht, siehe http://www.gnu.org/licenses/.

#### Copyright

2013 Technische Universitaet Muenchen

#### **Author**

Sebastian Rettenberger rettenbs@in.tum.de

## 8.55 /home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/scenarios/-Subcritical flow.h File Reference

```
#include "types.h"
```

## Classes

· class scenarios::Subcrit

### 8.55.1 Detailed Description

This file is part of SWE1D

SWE1D is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE1D is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE1D. If not, see http-://www.gnu.org/licenses/.

Diese Datei ist Teil von SWE1D.

## /home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/scenarios/Supercritical\_flow.h File Reference

SWE1D ist Freie Software: Sie koennen es unter den Bedingungen der GNU General Public License, wie von der Free Software Foundation, Version 3 der Lizenz oder (nach Ihrer Option) jeder spaeteren veroeffentlichten Version, weiterverbreiten und/oder modifizieren.

SWE1D wird in der Hoffnung, dass es nuetzlich sein wird, aber OHNE JEDE GEWAEHELEISTUNG, bereitgestellt; sogar ohne die implizite Gewaehrleistung der MARKTFAEHIGKEIT oder EIGNUNG FUER EINEN BESTIMMTEN ZWECK. Siehe die GNU General Public License fuer weitere Details.

Sie sollten eine Kopie der GNU General Public License zusammen mit diesem Programm erhalten haben. Wenn nicht, siehe http://www.gnu.org/licenses/.

#### Copyright

2013 Technische Universitaet Muenchen

#### **Author**

```
Sebastian Rettenberger rettenbs@in.tum.de
Thomas Blocherblocher@in.tum.de
```

## 8.56 /home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/scenarios/-Supercritical\_flow.h File Reference

```
#include "types.h"
```

#### Classes

· class scenarios::Supercrit

## 8.56.1 Detailed Description

This file is part of SWE1D

SWE1D is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE1D is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE1D. If not, see http-://www.gnu.org/licenses/.

Diese Datei ist Teil von SWE1D.

SWE1D ist Freie Software: Sie koennen es unter den Bedingungen der GNU General Public License, wie von der Free Software Foundation, Version 3 der Lizenz oder (nach Ihrer Option) jeder spaeteren veroeffentlichten Version, weiterverbreiten und/oder modifizieren.

SWE1D wird in der Hoffnung, dass es nuetzlich sein wird, aber OHNE JEDE GEWAEHELEISTUNG, bereitgestellt; sogar ohne die implizite Gewaehrleistung der MARKTFAEHIGKEIT oder EIGNUNG FUER EINEN BESTIMMTEN ZWECK. Siehe die GNU General Public License fuer weitere Details.

Sie sollten eine Kopie der GNU General Public License zusammen mit diesem Programm erhalten haben. Wenn nicht, siehe http://www.gnu.org/licenses/.

#### Copyright

2013 Technische Universitaet Muenchen

#### **Author**

```
Sebastian Rettenberger rettenbs@in.tum.de
Thomas Blocherblocher@in.tum.de
```

# 8.57 /home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/tools/args.cpp File Reference

```
#include "args.h"
```

## 8.57.1 Detailed Description

This file is part of SWE1D

SWE1D is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE1D is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE1D. If not, see http-://www.gnu.org/licenses/.

Diese Datei ist Teil von SWE1D.

SWE1D ist Freie Software: Sie koennen es unter den Bedingungen der GNU General Public License, wie von der Free Software Foundation, Version 3 der Lizenz oder (nach Ihrer Option) jeder spaeteren veroeffentlichten Version, weiterverbreiten und/oder modifizieren.

SWE1D wird in der Hoffnung, dass es nuetzlich sein wird, aber OHNE JEDE GEWAEHELEISTUNG, bereitgestellt; sogar ohne die implizite Gewaehrleistung der MARKTFAEHIGKEIT oder EIGNUNG FUER EINEN BESTIMMTEN ZWECK. Siehe die GNU General Public License fuer weitere Details.

Sie sollten eine Kopie der GNU General Public License zusammen mit diesem Programm erhalten haben. Wenn nicht, siehe http://www.gnu.org/licenses/.

#### Copyright

2013 Technische Universitaet Muenchen

#### **Author**

Sebastian Rettenberger rettenbs@in.tum.de

## 8.58 /home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/tools/args.h File Reference

```
#include "tools/logger.h"
#include <getopt.h>
#include <cstdlib>
#include <iostream>
#include <sstream>
```

#### **Classes**

class tools::Args

## 8.58.1 Detailed Description

This file is part of SWE1D

SWE1D is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE1D is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE1D. If not, see http-://www.gnu.org/licenses/.

Diese Datei ist Teil von SWE1D.

SWE1D ist Freie Software: Sie koennen es unter den Bedingungen der GNU General Public License, wie von der Free Software Foundation, Version 3 der Lizenz oder (nach Ihrer Option) jeder spaeteren veroeffentlichten Version, weiterverbreiten und/oder modifizieren.

SWE1D wird in der Hoffnung, dass es nuetzlich sein wird, aber OHNE JEDE GEWAEHELEISTUNG, bereitgestellt; sogar ohne die implizite Gewaehrleistung der MARKTFAEHIGKEIT oder EIGNUNG FUER EINEN BESTIMMTEN ZWECK. Siehe die GNU General Public License fuer weitere Details.

Sie sollten eine Kopie der GNU General Public License zusammen mit diesem Programm erhalten haben. Wenn nicht, siehe http://www.gnu.org/licenses/.

#### Copyright

2013 Technische Universitaet Muenchen

#### Author

Sebastian Rettenberger rettenbs@in.tum.de

## 8.59 /home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/tools/logger.cpp File Reference

```
#include "logger.h"
```

## 8.59.1 Detailed Description

This file is part of SWE1D

SWE1D is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE1D is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE1D. If not, see http-://www.gnu.org/licenses/.

Diese Datei ist Teil von SWE1D.

SWE1D ist Freie Software: Sie koennen es unter den Bedingungen der GNU General Public License, wie von der Free Software Foundation, Version 3 der Lizenz oder (nach Ihrer Option) jeder spaeteren veroeffentlichten Version, weiterverbreiten und/oder modifizieren.

SWE1D wird in der Hoffnung, dass es nuetzlich sein wird, aber OHNE JEDE GEWAEHELEISTUNG, bereitgestellt; sogar ohne die implizite Gewaehrleistung der MARKTFAEHIGKEIT oder EIGNUNG FUER EINEN BESTIMMTEN ZWECK. Siehe die GNU General Public License fuer weitere Details.

Sie sollten eine Kopie der GNU General Public License zusammen mit diesem Programm erhalten haben. Wenn nicht, siehe http://www.gnu.org/licenses/.

#### Copyright

2013 Technische Universitaet Muenchen

**Author** 

Sebastian Rettenberger rettenbs@in.tum.de

## 8.60 /home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/tools/logger.h

```
#include <cstdlib>
#include <iostream>
```

#### Classes

· class tools::Logger

## 8.60.1 Detailed Description

This file is part of SWE1D

SWE1D is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE1D is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE1D. If not, see http-://www.gnu.org/licenses/.

Diese Datei ist Teil von SWE1D.

SWE1D ist Freie Software: Sie koennen es unter den Bedingungen der GNU General Public License, wie von der Free Software Foundation, Version 3 der Lizenz oder (nach Ihrer Option) jeder spaeteren veroeffentlichten Version, weiterverbreiten und/oder modifizieren.

SWE1D wird in der Hoffnung, dass es nuetzlich sein wird, aber OHNE JEDE GEWAEHELEISTUNG, bereitgestellt; sogar ohne die implizite Gewaehrleistung der MARKTFAEHIGKEIT oder EIGNUNG FUER EINEN BESTIMMTEN ZWECK. Siehe die GNU General Public License fuer weitere Details.

Sie sollten eine Kopie der GNU General Public License zusammen mit diesem Programm erhalten haben. Wenn nicht, siehe http://www.gnu.org/licenses/.

#### Copyright

2013 Technische Universitaet Muenchen

**Author** 

Sebastian Rettenberger rettenbs@in.tum.de

# 8.61 /home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/types.h File Reference

## **Typedefs**

· typedef float T

## 8.61.1 Detailed Description

This file is part of SWE1D

SWE1D is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE1D is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE1D. If not, see http-://www.gnu.org/licenses/.

Diese Datei ist Teil von SWE1D.

SWE1D ist Freie Software: Sie koennen es unter den Bedingungen der GNU General Public License, wie von der Free Software Foundation, Version 3 der Lizenz oder (nach Ihrer Option) jeder spaeteren veroeffentlichten Version, weiterverbreiten und/oder modifizieren.

SWE1D wird in der Hoffnung, dass es nuetzlich sein wird, aber OHNE JEDE GEWAEHELEISTUNG, bereitgestellt; sogar ohne die implizite Gewaehrleistung der MARKTFAEHIGKEIT oder EIGNUNG FUER EINEN BESTIMMTEN ZWECK. Siehe die GNU General Public License fuer weitere Details.

Sie sollten eine Kopie der GNU General Public License zusammen mit diesem Programm erhalten haben. Wenn nicht, siehe http://www.gnu.org/licenses/.

## Copyright

2013 Technische Universitaet Muenchen

#### **Author**

Sebastian Rettenberger rettenbs@in.tum.de

## 8.62 /home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/Wave-Propagation.cpp File Reference

#include "WavePropagation.h"

## 8.62.1 Detailed Description

This file is part of SWE1D

SWE1D is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE1D is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE1D. If not, see http-://www.gnu.org/licenses/.

Diese Datei ist Teil von SWE1D.

SWE1D ist Freie Software: Sie koennen es unter den Bedingungen der GNU General Public License, wie von der Free Software Foundation, Version 3 der Lizenz oder (nach Ihrer Option) jeder spaeteren veroeffentlichten Version, weiterverbreiten und/oder modifizieren.

SWE1D wird in der Hoffnung, dass es nuetzlich sein wird, aber OHNE JEDE GEWAEHELEISTUNG, bereitgestellt; sogar ohne die implizite Gewaehrleistung der MARKTFAEHIGKEIT oder EIGNUNG FUER EINEN BESTIMMTEN ZWECK. Siehe die GNU General Public License fuer weitere Details.

Sie sollten eine Kopie der GNU General Public License zusammen mit diesem Programm erhalten haben. Wenn nicht, siehe http://www.gnu.org/licenses/.

#### Copyright

2013 Technische Universitaet Muenchen

#### **Author**

Sebastian Rettenberger rettenbs@in.tum.de

## 8.63 /home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/Wave-Propagation.h File Reference

```
#include "types.h"
#include "solvers/FWave.hpp"
```

#### **Classes**

· class WavePropagation

#### 8.63.1 Detailed Description

This file is part of SWE1D

SWE1D is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE1D is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE1D. If not, see http-://www.gnu.org/licenses/.

Diese Datei ist Teil von SWE1D.

SWE1D ist Freie Software: Sie koennen es unter den Bedingungen der GNU General Public License, wie von der Free Software Foundation, Version 3 der Lizenz oder (nach Ihrer Option) jeder spaeteren veroeffentlichten Version, weiterverbreiten und/oder modifizieren.

SWE1D wird in der Hoffnung, dass es nuetzlich sein wird, aber OHNE JEDE GEWAEHELEISTUNG, bereitgestellt; sogar ohne die implizite Gewaehrleistung der MARKTFAEHIGKEIT oder EIGNUNG FUER EINEN BESTIMMTEN ZWECK. Siehe die GNU General Public License fuer weitere Details.

Sie sollten eine Kopie der GNU General Public License zusammen mit diesem Programm erhalten haben. Wenn nicht, siehe http://www.gnu.org/licenses/.

Copyright

2013 Technische Universitaet Muenchen

**Author** 

Sebastian Rettenberger rettenbs@in.tum.de

# 8.64 /home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/writer/ConsoleWriter.h File Reference

```
#include "types.h"
#include <iostream>
```

#### **Classes**

· class writer::ConsoleWriter

### 8.64.1 Detailed Description

This file is part of SWE1D

SWE1D is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE1D is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE1D. If not, see http-://www.gnu.org/licenses/.

Diese Datei ist Teil von SWE1D.

SWE1D ist Freie Software: Sie koennen es unter den Bedingungen der GNU General Public License, wie von der Free Software Foundation, Version 3 der Lizenz oder (nach Ihrer Option) jeder spaeteren veroeffentlichten Version, weiterverbreiten und/oder modifizieren.

SWE1D wird in der Hoffnung, dass es nuetzlich sein wird, aber OHNE JEDE GEWAEHELEISTUNG, bereitgestellt; sogar ohne die implizite Gewaehrleistung der MARKTFAEHIGKEIT oder EIGNUNG FUER EINEN BESTIMMTEN ZWECK. Siehe die GNU General Public License fuer weitere Details.

Sie sollten eine Kopie der GNU General Public License zusammen mit diesem Programm erhalten haben. Wenn nicht, siehe http://www.gnu.org/licenses/.

Copyright

2013 Technische Universitaet Muenchen

**Author** 

Sebastian Rettenberger rettenbs@in.tum.de

## 8.65 /home/raphael/Programmieren/BPraktikum/SWE/submodules/SWE1D/src/writer/-VtkWriter.h File Reference

```
#include "types.h"
#include <cassert>
#include <fstream>
#include <sstream>
#include <string>
```

## Classes

· class writer::VtkWriter

## 8.65.1 Detailed Description

This file is part of SWE1D

SWE1D is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

SWE1D is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with SWE1D. If not, see http-://www.gnu.org/licenses/.

Diese Datei ist Teil von SWE1D.

SWE1D ist Freie Software: Sie koennen es unter den Bedingungen der GNU General Public License, wie von der Free Software Foundation, Version 3 der Lizenz oder (nach Ihrer Option) jeder spaeteren veroeffentlichten Version, weiterverbreiten und/oder modifizieren.

SWE1D wird in der Hoffnung, dass es nuetzlich sein wird, aber OHNE JEDE GEWAEHELEISTUNG, bereitgestellt; sogar ohne die implizite Gewaehrleistung der MARKTFAEHIGKEIT oder EIGNUNG FUER EINEN BESTIMMTEN ZWECK. Siehe die GNU General Public License fuer weitere Details.

Sie sollten eine Kopie der GNU General Public License zusammen mit diesem Programm erhalten haben. Wenn nicht, siehe http://www.gnu.org/licenses/.

### Copyright

2013 Technische Universitaet Muenchen

#### **Author**

Sebastian Rettenberger rettenbs@in.tum.de

## Index

$\sim$ Logger	hh, 105
tools::Logger, 82	/home/raphael/Programmieren/BPraktikum/SWE/src/blocks/rusanov/-
$\sim$ NetCdfWriter	SWE_RusanovBlock.cpp, 109
io::NetCdfWriter, 22	/home/raphael/Programmieren/BPraktikum/SWE/src/blocks/rusanov/-
$\sim$ SWE_Block	SWE_RusanovBlock.hh, 110
SWE_Block, 46	/home/raphael/Programmieren/BPraktikum/SWE/src/blocks/rusanov/-
~SWE_BlockCUDA	SWE_RusanovBlockCUDA.cu, 111
SWE_BlockCUDA, 52	/home/raphael/Programmieren/BPraktikum/SWE/src/blocks/rusanov/-
$\sim$ SWE_RusanovBlock	SWE_RusanovBlockCUDA.hh, 112
SWE_RusanovBlock, 61	/home/raphael/Programmieren/BPraktikum/SWE/src/examples/swe-
~SWE_RusanovBlockCUDA	_DimensionalSplitting.cpp, 119
SWE_RusanovBlockCUDA, 63	/home/raphael/Programmieren/BPraktikum/SWE/src/examples/swe-
~SWE_VisInfo	_mpi.cpp, 120
SWE_VisInfo, 71	/home/raphael/Programmieren/BPraktikum/SWE/src/examples/swe-
~SWE_WavePropagationBlock	_simple.cpp, 123
SWE_WavePropagationBlock, 73	/home/raphael/Programmieren/BPraktikum/SWE/src/opengl/vbo
~SWE_WavePropagationBlockCuda	cpp, 124
SWE_WavePropagationBlockCuda, 77	/home/raphael/Programmieren/BPraktikum/SWE/src/opengl/vbo
~Shader	h, 124
Shader, 34	/home/raphael/Programmieren/BPraktikum/SWE/src/scenarios/-
~Simulation	SWE_ArtificialTsunamiScenario.hh, 125
Simulation, 35	/home/raphael/Programmieren/BPraktikum/SWE/src/scenarios/-
~Visualization	SWE_AsagiScenario.cpp, 126
Visualization, 88	/home/raphael/Programmieren/BPraktikum/SWE/src/scenarios/-
/home/raphael/Programmieren/BPraktikum/SWE/src/block	0145
SWE_Block.cpp, 115	/home/raphael/Programmieren/BPraktikum/SWE/src/scenarios/-
/home/raphael/Programmieren/BPraktikum/SWE/src/block	·
SWE_Block.hh, 116	/home/raphael/Programmieren/BPraktikum/SWE/src/scenarios/-
/home/raphael/Programmieren/BPraktikum/SWE/src/block	·
SWE_WavePropagationBlock.cpp, 117	/home/raphael/Programmieren/BPraktikum/SWE/src/scenarios/-
/home/raphael/Programmieren/BPraktikum/SWE/src/block	
SWE_WavePropagationBlock.hh, 118	/home/raphael/Programmieren/BPraktikum/SWE/src/scenarios/-
/home/raphael/Programmieren/BPraktikum/SWE/src/block	•
SWE_BlockCUDA.cu, 93	/home/raphael/Programmieren/BPraktikum/SWE/src/scenarios/-
/home/raphael/Programmieren/BPraktikum/SWE/src/block	
SWE_BlockCUDA.hh, 94	/home/raphael/Programmieren/BPraktikum/SWE/src/scenarios/-
/home/raphael/Programmieren/BPraktikum/SWE/src/block	
SWE_BlockCUDA_kernels.cu, 96	/home/raphael/Programmieren/BPraktikum/SWE/src/testing/testing-
/home/raphael/Programmieren/BPraktikum/SWE/src/block	
SWE_BlockCUDA_kernels.hh, 98	/home/raphael/Programmieren/BPraktikum/SWE/src/tools/-
/home/raphael/Programmieren/BPraktikum/SWE/src/block	
SWE_WavePropagationBlockCuda.cu, 100	/home/raphael/Programmieren/BPraktikum/SWE/src/tools/-
/home/raphael/Programmieren/BPraktikum/SWE/src/block	•
SWE_WavePropagationBlockCuda.hh, 101	/home/raphael/Programmieren/BPraktikum/SWE/src/tools/-
/home/raphael/Programmieren/BPraktikum/SWE/src/block	•
SWE_WavePropagationBlockCuda_kernels	/home/raphael/Programmieren/BPraktikum/SWE/src/tools/help
cu, 101	hh, 133
	ধ্যেদি <b>অাবe</b> /raphael/Programmieren/BPraktikum/SWE/src/writer/-
SWE_WavePropagationBlockCuda_kernels	NetCdfWriter.cpp, 137
CTT_TTATOL TOPAGALIOTIDIOONOUGA_NETTICIS.	11010diffillollopp, 107

```
/home/raphael/Programmieren/BPraktikum/SWE/src/writer/-
                                                                                           SWE_AsagiJapanSmallVisInfo, 37
              NetCdfWriter.hh, 138
                                                                                           SWE VisInfo, 71
/home/raphael/Programmieren/BPraktikum/SWE/src/writerBoundaryEdge
              NetCdfWriterCP.cpp, 139
                                                                                           SWE Scenario.hh, 130
/home/raphael/Programmieren/BPraktikum/SWE/src/writerboundarySize
              NetCdfWriterCP.hh, 139
                                                                                           io::BoundarySize, 21
/home/raphael/Programmieren/BPraktikum/SWE/src/writerBoundaryType
              VtkWriter.cpp, 140
                                                                                           SWE Scenario.hh, 130
/home/raphael/Programmieren/BPraktikum/SWE/src/writer/-
               VtkWriter.hh, 141
                                                                                    Camera, 15
/home/raphael/Programmieren/BPraktikum/SWE/src/writer/-
                                                                                           Camera, 15
              Writer.hh, 141
                                                                                           displayImage, 15
/home/raphael/Programmieren/BPraktikum/SWE/submodules/-
                                                                                           orient, 15
              SWE1D/src/WavePropagation.cpp, 153
                                                                                           panning, 16
/home/raphael/Programmieren/BPraktikum/SWE/submodules/-
                                                                                           setCamera, 16
              SWE1D/src/WavePropagation.h, 154
                                                                                           startPanning, 16
/home/raphael/Programmieren/BPraktikum/SWE/submodules/-
                                                                                           viewDistance, 16
              SWE1D/src/main.cpp, 143
                                                                                           zoomln, 16
/home/raphael/Programmieren/BPraktikum/SWE/submodules/-
                                                                                           zoomOut, 16
              SWE1D/src/scenarios/ShockShock.h, 147
                                                                                    cleanUp
/home/raphael/Programmieren/BPraktikum/SWE/submodules/- Visualization, 88
              SWE1D/src/scenarios/Subcritical flow.h, 148
                                                                                    computeBathymetrySources
/home/raphael/Programmieren/BPraktikum/SWE/submodules/- SWE_RusanovBlock, 61
              SWE1D/src/scenarios/Supercritical_flow.h,
                                                                                    computeFlux
               149
                                                                                           SWE_RusanovBlock, 61
/home/raphael/Programmieren/BPraktikum/SWE/submodules/-computeLocalSV
              SWE1D/src/scenarios/constant_flow.h, 143
                                                                                           SWE RusanovBlock, 61
/home/raphael/Programmieren/BPraktikum/SWE/submodules/-computeMaxTimestep
              SWE1D/src/scenarios/dambreak.h, 144
                                                                                           SWE_Block, 46
/home/raphael/Programmieren/BPraktikum/SWE/submodules/-computeNetUpdates
              SWE1D/src/scenarios/dambreak_bathy.h,
                                                                                           solver::FWave, 36
               145
/home/raphael/Programmieren/BPraktikum/SWE/submodules/-
                                                                                           SWE WavePropagationBlockCuda kernels.cu,
              SWE1D/src/scenarios/eisbach.h, 146
                                                                                                  102
/home/raphael/Programmieren/BPraktikum/SWE/submodules/-
                                                                                           SWE\_Wave Propagation Block Cuda\_kernels.hh,
              SWE1D/src/scenarios/rarerare.h, 147
/home/raphael/Programmieren/BPraktikum/SWE/submodules/-
computeNumberOfBlockRows
                                                                                                  106
              SWE1D/src/tools/args.cpp, 150
/home/raphael/Programmieren/BPraktikum/SWE/submodules/-
computeNumericalFluxes
              SWE1D/src/tools/args.h, 150
                                                                                           SWE Block, 46
/home/raphael/Programmieren/BPraktikum/SWE/submodules/-
                                                                                           SWE_DimensionalSplitting, 56
              SWE1D/src/tools/logger.cpp, 151
                                                                                           SWE RusanovBlock, 61
/home/raphael/Programmieren/BPraktikum/SWE/submodules/-
                                                                                           SWE RusanovBlockCUDA, 63
              SWE1D/src/tools/logger.h, 152
                                                                                           SWE WavePropagationBlock, 73
/home/raphael/Programmieren/BPraktikum/SWE/submodules/-
                                                                                           SWE WavePropagationBlockCuda, 77
              SWE1D/src/types.h, 153
                                                                                           WavePropagation, 90
/home/raphael/Programmieren/BPraktikum/SWE/submodules/-
                                                                                    computeOneDPositionKernel
              SWE1D/src/writer/ConsoleWriter.h, 155
                                                                                           SWE WavePropagationBlockCuda kernels.cu,
/home/raphael/Programmieren/BPraktikum/SWE/submodules/-
              SWE1D/src/writer/VtkWriter.h, 156
/home/raphael/Programmieren/BPraktikum/SWE/submodules/f-\\ SWE\_WavePropagationBlockCuda\_kernels.hh, and the propagation of the
                                                                                                  108
              wave-solver/src/FWave.cpp, 142
/home/raphael/Programmieren/BPraktikum/SWE/submodules/ntroller, 16
                                                                                           Controller, 17
              wave-solver/src/FWave.hpp, 142
                                                                                           handleEvents, 18
bathyVerticalOffset
                                                                                           hasFocus, 18
                                                                                           isPaused, 18
       SWE BathymetryDamBreakVisInfo, 42
       SWE_VisInfo, 71
                                                                                    cout
bathyVerticalScaling
                                                                                           tools::Logger, 82
```

DimensionalSplittingTest, 18 disableShader	SWE_AsagiScenario, 40 getCUDA_bathymetry
Shader, 34	SWE_BlockCUDA, 53
displayImage	getCUDA_waterHeight
Camera, 15	SWE_BlockCUDA, 53
dynamicDisplacementAvailable	getCellCoord
SWE_AsagiScenario, 39	SWE_BlockCUDA.hh, 95
	getCellSize
enableShader	scenarios::ConstantFlow, 27
Shader, 34	scenarios::DamBreak, 28
endSimulation	scenarios::Eisbach, 29
SWE_AsagiScenario, 39	scenarios::RareRare, 30
exchangeBottomTopGhostLayers	scenarios::ShockShock, 30
swe_mpi.cpp, 121	scenarios::Subcrit, 31
exchangeLeftRightGhostLayers	scenarios::Supercrit, 32
swe_mpi.cpp, 122	getCudaNormalsPtr
_ 1 117	Visualization, 88
FWaveSolverTest, 20	getCudaWaterSurfacePtr
testBathymetry, 20	Visualization, 88
testEigencoefficients, 20	getDischarge_hu
testEigenvalues, 20	SWE_Block, 46
testFlux, 20	getDischarge hv
testSteadyState, 21	SWE_Block, 46
testSupersonic, 21	getEdgeCoord
finalize	SWE_BlockCUDA.hh, 95
SWE_BlockCUDA, 53	getHeight
finialize	scenarios::ConstantFlow, 27
VBO, 87	scenarios::DamBreak, 28
Float1D, 19	scenarios::Eisbach, 29
Float2D, 19	scenarios::RareRare, 30
Float2D, 19	scenarios::ShockShock, 30
Float2D, 19	scenarios::Subcrit, 31
	scenarios::Supercrit, 32
generateBaseFileName	getMaxTimestep
help.hh, 134	SWE_Block, 46
generateContainerFileName	getMomentum
help.hh, 134	scenarios::ConstantFlow, 27
generateFileName	scenarios::DamBreak, 28
help.hh, 134, 135	scenarios::Eisbach, 29
getBathyCoord	scenarios::RareRare, 30
SWE_BlockCUDA.hh, 95	scenarios::ShockShock, 31
getBathymetry	scenarios::Subcrit, 31
scenarios::ConstantFlow, 27	scenarios::Supercrit, 32
scenarios::DamBreak, 28	getName
scenarios::Eisbach, 29	VBO, 87
scenarios::Subcrit, 31	getScalingApproximation
scenarios::Supercrit, 32	Simulation, 35
SWE_AsagiScenario, 39	getUniformLocation
SWE_Block, 46	Shader, 34
getBathymetryAndDynamicDisplacement	getWaterHeight
SWE_AsagiScenario, 39	SWE_AsagiScenario, 40
getBoundaryPos	SWE_BathymetryDamBreakScenario, 41
SWE_AsagiScenario, 40	SWE_Block, 47
SWE_BathymetryDamBreakScenario, 41	grabGhostLayer
SWE_DamBreakScenario, 56	SWE_Block, 47
SWE_RadialDamBreakScenario, 59	SWE_BlockCUDA, 53
SWE_SplashingPoolScenario, 67	
SWE_TestingScenario, 68	handleEvents
getBoundaryType	Controller, 18

hasFocus	kernelMaximum
Controller, 18	SWE_BlockCUDA_kernels.hh, 99
help.hh	SWE_RusanovBlockCUDA_kernels.cu, 114
generateBaseFileName, 134	SWE_RusanovBlockCUDA_kernels.hh, 115
generateContainerFileName, 134	kernelRightBoundary
generateFileName, 134, 135	SWE_BlockCUDA_kernels.cu, 97
	SWE_BlockCUDA_kernels.hh, 99
init	kernelTopBoundary
SWE_BlockCUDA, 53	SWE_BlockCUDA_kernels.cu, 97
VBO, 87	SWE_BlockCUDA_kernels.hh, 99
Visualization, 88	kernelTopCopyLayer
initScenario	SWE_BlockCUDA_kernels.cu, 97
SWE_Block, 47	SWE_BlockCUDA_kernels.hh, 99
initWallClockTime	kernelTopGhostBoundary
tools::Logger, 82	SWE_BlockCUDA_kernels.cu, 97
io::BoundarySize, 21	SWE_BlockCUDA_kernels.hh, 100
boundarySize, 21	
io::NetCdfWriter, 22	Logger
$\sim$ NetCdfWriter, 22	tools::Logger, 80
NetCdfWriter, 22	logger
writeTimeStep, 24	tools::Logger, 86
io::VtkWriter, 24	
VtkWriter, 25	main
writeTimeStep, 25	swe_DimensionalSplitting.cpp, 119
io::Writer, 26	swe_mpi.cpp, 122
writeTimeStep, 26	swe_simple.cpp, 124
Writer, 26	mainpage.txt, 93
isExtensionSupported	maxTimestep
Visualization, 89	SWE_Block, 50
isPaused	
Controller, 18	NetCdfWriter
	io::NetCdfWriter, 22
kernelBottomBoundary	
SWE_BlockCUDA_kernels.cu, 97	operator<<
SWE_BlockCUDA_kernels.hh, 99	SWE_RusanovBlock, 62
kernelBottomCopyLayer	SWE_RusanovBlock.cpp, 110
SWE_BlockCUDA_kernels.cu, 97	SWE_RusanovBlock.hh, 111
SWE_BlockCUDA_kernels.hh, 99	SWE_RusanovBlockCUDA, 64
kernelBottomGhostBoundary	SWE_RusanovBlockCUDA.cu, 111
SWE_BlockCUDA_kernels.cu, 97	SWE_RusanovBlockCUDA.hh, 112
SWE_BlockCUDA_kernels.hh, 99	tools::Logger, 82
kernelComputeBathymetrySources	orient
SWE_RusanovBlockCUDA_kernels.cu, 113	Camera, 15
SWE RusanovBlockCUDA kernels.hh, 115	,
kernelComputeFluxesF	panning
SWE_RusanovBlockCUDA_kernels.cu, 113	Camera, 16
SWE RusanovBlockCUDA kernels.hh, 115	printCellSize
kernelComputeFluxesG	tools::Logger, 82
SWE_RusanovBlockCUDA_kernels.cu, 114	printCpuCommunicationTime
SWE RusanovBlockCUDA kernels.hh, 115	tools::Logger, 83
kernelEulerTimestep	printCpuTime
SWE RusanovBlockCUDA kernels.cu, 114	tools::Logger, 83
SWE_RusanovBlockCUDA_kernels.hh, 115	printDeviceInformation
kernelHdBufferEdges	SWE_BlockCUDA, 53
SWE_BlockCUDA_kernels.cu, 97	printFinishMessage
SWE_BlockCUDA_kernels.cu, 97 SWE_BlockCUDA_kernels.hh, 99	tools::Logger, 83
kernelLeftBoundary	
· · · · · · · · · · · · · · · · · · ·	printIterationsDone
SWE_BlockCUDA_kernels.cu, 97	tools::Logger, 83
SWE_BlockCUDA_kernels.hh, 99	printNumberOfBlocks

table of a many 00	and Day and any Day 44
tools::Logger, 83	getBoundaryPos, 41
printNumberOfCells	getWaterHeight, 41
tools::Logger, 83	SWE_BathymetryDamBreakVisInfo, 42
printNumberOfCellsPerProcess	bathyVerticalOffset, 42
tools::Logger, 84	SWE_Block, 42
printNumberOfProcesses	$\sim$ SWE_Block, 46
tools::Logger, 84	computeMaxTimestep, 46
printOutputFileCreation	computeNumericalFluxes, 46
tools::Logger, 84	getBathymetry, 46
printOutputTime	getDischarge_hu, 46
tools::Logger, 84	getDischarge_hv, 46
printSimulationTime	getMaxTimestep, 46
tools::Logger, 84	getWaterHeight, 47
printSolverStatistics	grabGhostLayer, 47
tools::Logger, 85	initScenario, 47
printStartMessage	maxTimestep, 50
tools::Logger, 85	registerCopyLayer, 47
printStatisticsMessage	SWE_Block, 45
tools::Logger, 85	setBathymetry, 47, 48
printString	setBoundaryBathymetry, 48
tools::Logger, 85	setBoundaryConditions, 48
printWallClockTime	setBoundaryType, 48
tools::Logger, 85	setDodridary type, 40
printWelcomeMessage	_
tools::Logger, 86	setGhostLayer, 48
	setWaterHeight, 48
readNetCDF	simulate, 49
SWE_NetCDFCheckpointScenario, 58	SWE_Block, 45
SWE TsunamiScenario, 70	synchAfterWrite, 49
registerCopyLayer	synchBathymetryAfterWrite, 49
SWE_Block, 47	synchBathymetryBeforeRead, 49
SWE_BlockCUDA, 53	synchBeforeRead, 49
renderDisplay	synchCopyLayerBeforeRead, 49
Visualization, 89	synchDischargeAfterWrite, 49
resizeWindow	synchDischargeBeforeRead, 50
Visualization, 89	synchGhostLayerAfterWrite, 50
restart	synchWaterHeightAfterWrite, 50
Simulation, 35	synchWaterHeightBeforeRead, 50
runCuda	updateUnknowns, 50
Simulation, 35	SWE_Block1D, 51
runTimestep	SWE_BlockCUDA, 51
SWE DimensionalSplitting, 56	$\sim$ SWE_BlockCUDA, 52
OVVE_Dimensional opinting, 50	finalize, 53
SWE_ArtificialTsunamiScenario, 36	getCUDA_bathymetry, 53
SWE_AsagiGrid, 37	getCUDA_waterHeight, 53
SWE_AsagiJapanSmallVisInfo, 37	grabGhostLayer, 53
bathyVerticalScaling, 37	init, 53
waterVerticalScaling, 38	printDeviceInformation, 53
SWE_AsagiScenario, 38	registerCopyLayer, 53
dynamicDisplacementAvailable, 39	SWE_BlockCUDA, 52
endSimulation, 39	setBoundaryConditions, 54
getBathymetry, 39	SWE_BlockCUDA, 52
getBathymetryAndDynamicDisplacement, 39	synchAfterWrite, 54
getBoundaryPos, 40	synchBathymetryAfterWrite, 54
getBoundaryType, 40	synchBathymetryBeforeRead, 54
getWaterHeight, 40	synchBeforeRead, 54
-	-
SWE_AsagiScenario, 38	synchCopyLayerBeforeRead, 54
SWE_AsagiScenario, 38	synchDischargeAfterWrite, 54
SWE_BathymetryDamBreakScenario, 41	synchDischargeBeforeRead, 55

synchGhostLayerAfterWrite, 55	SWE_RusanovBlockCUDA, 62
synchWaterHeightAfterWrite, 55	computeNumericalFluxes, 63
synchWaterHeightBeforeRead, 55	operator<<, 64
SWE_BlockCUDA.hh	SWE_RusanovBlockCUDA, 63
getBathyCoord, 95	simulate, 63
getCellCoord, 95	simulateTimestep, 63
getEdgeCoord, 95	SWE_RusanovBlockCUDA, 63
SWE_BlockCUDA_kernels.cu	updateUnknowns, 64
kernelBottomBoundary, 97	SWE RusanovBlockCUDA.cu
kernelBottomCopyLayer, 97	operator<<, 111
kernelBottomGhostBoundary, 97	SWE RusanovBlockCUDA.hh
kernelHdBufferEdges, 97	operator<<, 112
kernelLeftBoundary, 97	SWE_RusanovBlockCUDA_kernels.cu
kernelRightBoundary, 97	kernelComputeBathymetrySources, 113
- · · · · · · · · · · · · · · · · · · ·	· · · · · · · · · · · · · · · · · · ·
kernelTopBoundary, 97	kernelComputeFluxesF, 113
kernelTopCopyLayer, 97	kernelComputeFluxesG, 114
kernelTopGhostBoundary, 97	kernelEulerTimestep, 114
SWE_BlockCUDA_kernels.hh	kernelMaximum, 114
kernelBottomBoundary, 99	SWE_RusanovBlockCUDA_kernels.hh
kernelBottomCopyLayer, 99	kernelComputeBathymetrySources, 115
kernelBottomGhostBoundary, 99	kernelComputeFluxesF, 115
kernelHdBufferEdges, 99	kernelComputeFluxesG, 115
kernelLeftBoundary, 99	kernelEulerTimestep, 115
kernelMaximum, 99	kernelMaximum, 115
kernelRightBoundary, 99	SWE_Scenario, 64
kernelTopBoundary, 99	SWE_Scenario.hh
kernelTopCopyLayer, 99	BoundaryEdge, 130
kernelTopGhostBoundary, 100	BoundaryType, 130
SWE_DamBreakScenario, 55	SWE_SeaAtRestScenario, 65
getBoundaryPos, 56	SWE_SplashingConeScenario, 66
SWE_DimensionalSplitting, 56	SWE_SplashingPoolScenario, 67
computeNumericalFluxes, 56	getBoundaryPos, 67
runTimestep, 56	SWE_TestingScenario, 67
simulate, 57	getBoundaryPos, 68
updateUnknowns, 57	SWE_TsunamiScenario, 68
SWE_NetCDFCheckpointScenario, 57	readNetCDF, 70
readNetCDF, 58	SWE_TsunamiScenario, 69
SWE_NetCDFCheckpointScenario, 57	SWE TsunamiScenario, 69
SWE_NetCDFCheckpointScenario, 57	SWE_TsunamiScenarioTest, 70
SWE_NetCDFScenario, 58	SWE_VisInfo, 71
SWE_RadialDamBreakScenario, 59	~SWE_VisInfo, 71
getBoundaryPos, 59	bathyVerticalOffset, 71
SWE RusanovBlock, 59	bathyVerticalScaling, 71
~SWE RusanovBlock, 61	waterVerticalScaling, 72
	<u> </u>
computeBathymetrySources, 61	SWE_WavePropagationBlock, 72
computeFlux, 61	~SWE_WavePropagationBlock, 73
computeLocalSV, 61	computeNumericalFluxes, 73
computeNumericalFluxes, 61	SWE_WavePropagationBlock, 72
operator<<, 62	simulate, 73
SWE_RusanovBlock, 61	simulateTimestep, 74
simulate, 61	SWE_WavePropagationBlock, 72
simulateTimestep, 62	updateUnknowns, 74
SWE_RusanovBlock, 61	SWE_WavePropagationBlockCuda, 74
updateUnknowns, 62	$\sim$ SWE_WavePropagationBlockCuda, 77
SWE_RusanovBlock.cpp	computeNumericalFluxes, 77
operator<<, 110	SWE_WavePropagationBlockCuda, 75
SWE_RusanovBlock.hh	simulate, 78
operator<<, 111	simulateTimestep, 78

SWE WavePropagationBlockCuda, 75	setGhostLayer
updateUnknowns, 78	SWE Block, 48
SWE_WavePropagationBlockCuda_kernels.cu	setOutflowBoundaryConditions
computeNetUpdatesKernel, 102	WavePropagation, 90
computeOneDPositionKernel, 104	setProcessRank
updateUnknownsKernel, 104	tools::Logger, 86
SWE_WavePropagationBlockCuda_kernels.hh	setRenderingMode
computeNetUpdatesKernel, 106	Visualization, 89
computeOneDPositionKernel, 108	setUniform
updateUnknownsKernel, 108	Shader, 34
scenarios::ConstantFlow, 27	setWaterHeight
getBathymetry, 27	SWE_Block, 48
getCellSize, 27	Shader, 32
getHeight, 27	$\sim$ Shader, 34
getMomentum, 27	disableShader, 34
scenarios::DamBreak, 28	enableShader, 34
getBathymetry, 28	getUniformLocation, 34
getCellSize, 28	setUniform, 34
getHeight, 28	Shader, 33
getMomentum, 28	shadersLoaded, 34
scenarios::Eisbach, 29	shadersLoaded
getBathymetry, 29	Shader, 34
getCellSize, 29	showNextText
getHeight, 29	Text, 79
getMomentum, 29	simulate
scenarios::RareRare, 30	SWE_Block, 49
getCellSize, 30	SWE_DimensionalSplitting, 57
getHeight, 30	SWE_RusanovBlock, 61
getMomentum, 30	SWE_RusanovBlockCUDA, 63
scenarios::ShockShock, 30	SWE_WavePropagationBlock, 73
getCellSize, 30	SWE_WavePropagationBlockCuda, 78
getHeight, 30	simulateTimestep
getMomentum, 31	SWE_RusanovBlock, 62
scenarios::Subcrit, 31	SWE_RusanovBlockCUDA, 63
getBathymetry, 31	SWE_WavePropagationBlock, 74
getCellSize, 31	SWE_WavePropagationBlockCuda, 78
getHeight, 31	Simulation, 34
getMomentum, 31	$\sim$ Simulation, 35
scenarios::Supercrit, 32	getScalingApproximation, 35
getBathymetry, 32	restart, 35
getCellSize, 32	runCuda, 35
getHeight, 32	setBathBuffer, 35
getMomentum, 32	Simulation, 35
setBathBuffer	solver::FWave
Simulation, 35	computeNetUpdates, 36
setBathymetry	solver::FWave< T >, 35
SWE_Block, 47, 48	startPanning
setBoundaryBathymetry	Camera, 16
SWE_Block, 48	swe_DimensionalSplitting.cpp
setBoundaryConditions	main, 119
SWE_Block, 48	swe_mpi.cpp
SWE_BlockCUDA, 54	computeNumberOfBlockRows, 121
setBoundaryType	exchangeBottomTopGhostLayers, 121
SWE_Block, 48	exchangeLeftRightGhostLayers, 122
setCamera	main, 122
Camera, 16	swe_simple.cpp
setDischarge	main, 124
SWE_Block, 48	synchAfterWrite

SWE_Block, 49	printNumberOfBlocks, 83
SWE_BlockCUDA, 54	printNumberOfCells, 83
synchBathymetryAfterWrite	printNumberOfCellsPerProcess, 84
SWE_Block, 49	printNumberOfProcesses, 84
SWE_BlockCUDA, 54	printOutputFileCreation, 84
synchBathymetryBeforeRead	printOutputTime, 84
SWE Block, 49	printSimulationTime, 84
SWE BlockCUDA, 54	printSolverStatistics, 85
synchBeforeRead	printStartMessage, 85
SWE Block, 49	printStatisticsMessage, 85
SWE BlockCUDA, 54	printString, 85
synchCopyLayerBeforeRead	printWallClockTime, 85
SWE_Block, 49	printWelcomeMessage, 86
SWE_BlockCUDA, 54	setProcessRank, 86
synchDischargeAfterWrite	updateCpuCommunicationTime, 86
SWE_Block, 49	updateCpuTime, 86
SWE BlockCUDA, 54	tools::ProgressBar, 86
synchDischargeBeforeRead	update, 86
SWE_Block, 50	apadio, oo
SWE BlockCUDA, 55	update
synchGhostLayerAfterWrite	tools::ProgressBar, 86
SWE_Block, 50	updateCpuCommunicationTime
SWE_BlockCUDA, 55	tools::Logger, 86
synchWaterHeightAfterWrite	updateCpuTime
SWE_Block, 50	tools::Logger, 86
	updateUnknowns
SWE_BlockCUDA, 55	SWE Block, 50
synchWaterHeightBeforeRead SWE Block, 50	SWE_DimensionalSplitting, 57
SWE_BlockCUDA, 55	SWE_RusanovBlock, 62
SWL_blockGodA, 55	SWE_RusanovBlockCUDA, 64
testBathymetry	SWE_WavePropagationBlock, 74
FWaveSolverTest, 20	SWE_WavePropagationBlockCuda, 78
testEigencoefficients	WavePropagation, 91
FWaveSolverTest, 20	updateUnknownsKernel
testEigenvalues	SWE_WavePropagationBlockCuda_kernels.cu,
FWaveSolverTest, 20	104
testFlux	SWE_WavePropagationBlockCuda_kernels.hh,
FWaveSolverTest, 20	108
testSteadyState	
FWaveSolverTest, 21	VBO, 87
testSupersonic	finialize, 87
FWaveSolverTest, 21	getName, 87
Text, 78	init, 87
showNextText, 79	viewDistance
toggleRenderingMode	Camera, 16
Visualization, 89	Visualization, 87
tools::Args, 79	$\sim$ Visualization, 88
tools::Logger, 79	cleanUp, 88
∼Logger, 82	getCudaNormalsPtr, 88
cout, 82	getCudaWaterSurfacePtr, 88
initWallClockTime, 82	init, 88
Logger, 80	isExtensionSupported, 89
logger, 86	renderDisplay, 89
operator<<, 82	resizeWindow, 89
printCellSize, 82	setRenderingMode, 89
printCpuCommunicationTime, 83	toggleRenderingMode, 89
printCpuTime, 83	Visualization, 88
printFinishMessage, 83	VtkWriter
printIterationsDone, 83	io::VtkWriter, 25

```
waterVerticalScaling
     SWE_AsagiJapanSmallVisInfo, 38
     SWE\_VisInfo,\, \textcolor{red}{72}
WavePropagation, 89
    computeNumericalFluxes, 90
     setOutflowBoundaryConditions, 90
     updateUnknowns, 91
     WavePropagation, 90
     WavePropagation, 90
write
     writer::ConsoleWriter, 91
    writer::VtkWriter, 92
writeTimeStep
    io::NetCdfWriter, 24
    io::VtkWriter, 25
    io::Writer, 26
Writer
     io::Writer, 26
writer::ConsoleWriter, 91
    write, 91
writer::VtkWriter, 91
    write, 92
zoomIn
     Camera, 16
zoomOut
     Camera, 16
```