

---

# **SPlisHSPlasH**

***Release 2.8.2***

**Interactive Computer Graphics**

**Jun 17, 2020**



## GETTING STARTED:

<b>1</b>	<b>Getting started</b>	<b>1</b>
1.1	SPH Simulator	1
1.2	Python bindings	2
1.3	Tools	2
1.4	partio2vtk	2
1.5	PartioViewer	2
1.6	SurfaceSampling	4
1.7	VolumeSampling	4
<b>2</b>	<b>SPlisHSPlasH Scene Files</b>	<b>5</b>
2.1	Configuration	5
2.2	FluidBlocks	9
2.3	FluidModels	9
2.4	Emitters	10
2.5	RigidBodies	11
2.6	Materials	12
2.7	Animation fields	14
<b>3</b>	<b>pySPlisHSPlasH</b>	<b>17</b>
3.1	Python bindings for the SPlisHSPlasH library	17
3.2	Requirements	17
3.3	Installation	17
3.4	I want to see something very very quickly	18
3.5	Minimal working example	18
3.6	SPH Simulator.py	19
3.7	Modifying other properties	19
<b>4</b>	<b>Creating Scenes</b>	<b>21</b>
4.1	Loading the empty scene	21
4.2	Recreating the double dam break scenario	21
4.3	Putting it all together	22
4.4	Loading a scene from file	22
<b>5</b>	<b>Restrictions</b>	<b>23</b>
<b>6</b>	<b>Library API</b>	<b>25</b>
6.1	Class Hierarchy	25
6.2	File Hierarchy	25
6.3	Full API	25
<b>7</b>	<b>Indices and tables</b>	<b>151</b>



## GETTING STARTED

This page should give you a short overview of SPLisHSPlasH.

SPLisHSPlasH currently consists of a simulators and different tools which are introduced in the following:

### 1.1 SPH Simulator

This application reads a SPLisHSPlasH scene file and performs a simulation of the scene.

The scene file format is explained [here](#).

#### 1.1.1 Command line options:

- -h, -help: Print help text.
- -no-cache: Disable caching of boundary samples/maps.
- -state-file: Load a simulation state of the corresponding scene.
- -output-dir: Output directory for log file and partio files.
- -no-initial-pause: Disable caching of boundary samples/maps.
- -no-gui: Disable graphical user interface. The simulation is run only in the command line without graphical output. The “stopAt” option must be set in the scene file or by the next parameter.
- -stopAt arg: Sets or overwrites the stopAt parameter of the scene.
- -param arg: Sets or overwrites a parameter of the scene.
  - Setting a fluid parameter: ::
    - \* Example: -param Fluid:viscosity:0.01
  - Setting a configuration parameter: :
    - \* Example: -param cflMethod:1

## 1.1.2 Hotkeys

- Space: pause/continue simulation
- r: reset simulation
- w: wireframe rendering of meshes
- i: print all field information of the selected particles to the console
- s: save current simulation state
- l: load simulation state (currently only Windows)
- ESC: exit

## 1.2 Python bindings

SPlisHSPlasH implements bindings for python using [pybind11](#). See the *getting started guide*.

### 1.2.1 Impatient installation guide

In order to install, simply clone the repository and run pip install on the repository. It is recommended, that you set up a **virtual environment** for this, because cache files will be stored in the directory of the python installation along with models and scene files.

```
git clone https://github.com/InteractiveComputerGraphics/SPlisHSPlasH.git
pip install SPlisHSPlasH/
```

## 1.3 Tools

### 1.4 partio2vtk

A tool to convert partion files in vtk files. In this way the particle data which is exported from SPlisHSPlasH can be converted to the vtk format. This is useful to import the data in ParaView for visualization.

### 1.5 PartioViewer

The simulators can export the particle simulation data using the partio file format. The PartioViewer can read such a file and render the particle data using OpenGL. This tool is able to handle multiphase data and rigid body data. It can create image sequences and movies (using ffmpeg).

To visualize a sequence of partio files or a single file, call (the index in the file name is used for the sequence):

```
PartioViewer fluid_data_1.bgeo
```

This tool is also able to read a complete output directory:

```
PartioViewer output/DamBreakModel
```

In this case the tool searches for the partio files of multiple phases in the subdirectory “partio” and for rigid body data in “rigid\_bodies”.

Note: To generate videos you must tell PartioViewer where it can find the ffmpeg executable.

### 1.5.1 Command line options:

- -h, -help: Print help
- -renderSequence: Render a sequence from startFrame to endFrame as jpeg.
- -renderVideo: Render a sequence from startFrame to endFrame as video. This function requires ffmpeg which must be in the PATH or the ffmpegPath parameter must be set.
- -noOverwrite: Do not overwrite existing frames when using -renderSequence option. Existing frames are not loaded at all which accelerates the image sequence generation.
- -o, -outdir arg: Output directory for images
- -rbData arg: Rigid body data to visualize (bin file)
- -ffmpegPath arg: Path of the ffmpeg executable.
- -width arg: Width of the image in pixels. (default: 1024)
- -height arg: Height of the image in pixels. (default: 768)
- -fps arg: Frame rate of video. (default: 25)
- -r, -radius arg: Particle radius (default: 0.025)
- -s, -startFrame arg: Start frame (only used if value is  $\geq 0$ ) (default: -1)
- -e, -endFrame arg: End frame (only used if value is  $\geq 0$ ) (default: -1)
- -colorField arg: Name of field that is used for the color. (default: velocity)
- -colorMapType arg: Color map (0=None, 1=Jet, 2=Plasma) (default: 1)
- -renderMinValue arg: Min value of field. (default: 0.0)
- -renderMaxValue arg: Max value of field. (default: 10.0)
- -camPos arg: Camera position (e.g. -camPos “0 1 5”) (default: 0 3 10)
- -camLookat arg: Camera lookat (e.g. -camLookat “0 0 0”) (default: 0 0 0)

### 1.5.2 Hotkeys

- Space: pause/continue simulation
- r: reset simulation
- w: wireframe rendering of meshes
- i: print all field information of the selected particles to the console
- s: save current frame as jpg image
- v: generate video
- j: generate image sequence
- +: step to next frame
- -: step to previous frame

- ESC: exit

## 1.6 SurfaceSampling

A popular boundary handling method which is also implemented in SPlisHSPlasH uses a particle sampling of the surfaces of all boundary objects. This command line tool can generate such a surface sampling. Note that the same surface sampling is also integrated in the simulators and the samplings are generated automatically if they are required. However, if you want to generate a surface sampling manually, then you can use this tool.

## 1.7 VolumeSampling

The simulators can load particle data from partio files. This particle data then defines the initial configuration of the particles in the simulation. The VolumeSampling tool allows you to sample a volumetric object with particle data. This means you can load an OBJ file with a closed surface geometry and sample the interior with particles.



## SPLISHSPLASH SCENE FILES

A SPLisHSPlasH scene file is a json file which can contain the following blocks:

- Configuration
- FluidBlocks
- FluidModels
- Emitters
- RigidBodies
- Fluid parameter block
- Animation fields

### 2.1 Configuration

This part contains the general settings of the simulation and the pressure solver.

Example code:

```
"Configuration":
{
  "pause": true,
  "sim2D": false,
  "timeStepSize": 0.001,
  "numberOfStepsPerRenderUpdate": 2,
  "particleRadius": 0.025,
  "simulationMethod": 4,
  "gravitation": [0.0,-9.81,0],
  "cflMethod": 1,
  "cflFactor": 1,
  "cflMaxTimeStepSize": 0.005,
  "maxIterations": 100,
  "maxError": 0.01,
  "maxIterationsV": 100,
  "maxErrorV": 0.1,
  "stiffness": 50000,
  "exponent": 7,
  "velocityUpdateMethod": 0,
  "enableDivergenceSolver": true
}
```

### 2.1.1 General:

- `pause (bool)`: Pause simulation at beginning.
- `pauseAt (float)`: Pause simulation at the given time. When the value is negative, the simulation is not paused.
- `stopAt (float)`: Stop simulation at the given time and exit. When the value is negative, the simulation is not stopped.
- `cameraPosition (vec3)`: Initial position of the camera.
- `cameraLookat (vec3)`: Lookat point of the camera.

### 2.1.2 Visualization:

- `numberOfStepsPerRenderUpdate (int)`: Number of simulation steps per rendered frame
- `renderWalls (int)`:
  - 0: None
  - 1: Particles (all)
  - 2: Particles (no walls)
  - 3: Geometry (all)
  - 4: Geometry (no walls)

### 2.1.3 Export

- `enablePartioExport (bool)`: Enable/disable partio export (default: false).
- `enableVTKExport (bool)`: Enable/disable VTK export (default: false).
- `enableRigidBodyExport (bool)`: Enable/disable rigid body export (default: false).
- `dataExportFPS (float)`: Frame rate of particle and rigid body export (default: 25).
- `particleAttributes (string)`: A list of attribute names separated by “;” that should be exported in the particle files (e.g. “velocity;density”) (default: “velocity”).
- `enableStateExport (bool)`: Enable/disable export of complete simulation state (default: false).
- `stateExportFPS (float)`: Frame rate of simulation state export (default: 1).

### 2.1.4 Simulation:

- `timeStepSize (float)`: The initial time step size used for the time integration. If you use an adaptive time stepping, this size will change during the simulation (default: 0.001).
- `particleRadius (float)`: The radius of the particles in the simulation (all have the same radius) (default: 0.025).
- `sim2D (bool)`: If this parameter is set to true, a 2D simulation is performed instead of a 3D simulation (default: false).
- `enableZSort (bool)`: Enable z-sort to improve cache hits and therefore to improve the performance (default: true).
- `gravitation (vec3)`: Vector to define the gravitational acceleration (default: [0,-9.81,0]).
- `maxIterations (int)`: Maximal number of iterations of the pressure solver (default: 100).

- `maxError` (float): Maximal density error in percent which the pressure solver tolerates (default: 0.01).
- `boundaryHandlingMethod` (int): The boundary handling method that is used in the simulation (default: 2, Volume Maps):
  - 0: particle-based boundaries (Akinici et al. 2012)
  - 1: density maps (Koschier et al. 2017)
  - 2: volume maps (Bender et al. 2019)
- `simulationMethod` (int): The pressure solver method used in the simulation (default: 4, DFSPH):
  - 0: Weakly compressible SPH for free surface flows (WCSPH)
  - 1: Predictive-corrective incompressible SPH (PCISPH)
  - 2: Position based fluids (PBF)
  - 3: Implicit incompressible SPH (IISPH)
  - 4: Divergence-free smoothed particle hydrodynamics (DFSPH)
  - 5: Projective Fluids (dynamic boundaries not supported yet)

### 2.1.5 WCSPH parameters:

- `stiffness` (float): Stiffness coefficient of the equation of state.
- `exponent` (float): Exponent in the equation of state.

### 2.1.6 PBF parameters:

- `velocityUpdateMethod` (int):
  - 0: First Order Update
  - 1: Second Order Update

### 2.1.7 DFSPH parameters:

- `enableDivergenceSolver` (bool): Turn divergence solver on/off.
- `maxIterationsV` (int): Maximal number of iterations of the divergence solver.
- `maxErrorV` (float): Maximal divergence error in percent which the pressure solver tolerates.

### 2.1.8 Projective Fluids parameters:

- `stiffness` (float): Stiffness coefficient used by the pressure solver.

### 2.1.9 Kernel:

- **kernel (int):** Kernel function used in the SPH model.
  - For a 3D simulation:
    - \* 0: Cubic spline
    - \* 1: Wendland quintic C2
    - \* 2: Poly6
    - \* 3: Spiky
    - \* 4: Precomputed cubic spline (faster than cubic spline)
  - For a 2D simulation:
    - \* 0: Cubic spline
    - \* 1: Wendland quintic C2
- **gradKernel (int):** Gradient of the kernel function used in the SPH model.
  - For a 3D simulation:
    - \* 0: Cubic spline
    - \* 1: Wendland quintic C2
    - \* 2: Poly6
    - \* 3: Spiky
    - \* 4: Precomputed cubic spline (faster than cubic spline)
  - For a 2D simulation:
    - \* 0: Cubic spline
    - \* 1: Wendland quintic C2

### 2.1.10 CFL:

- **cflMethod (int):** CFL method used for adaptive time stepping.
  - 0: No adaptive time stepping
  - 1: Use CFL condition
  - 2: Use CFL condition and consider number of pressure solver iterations
- **cflFactor (float):** Factor to scale the CFL time step size.
- **cflMinTimeStepSize (float):** Min. allowed time step size.
- **cflMaxTimeStepSize (float):** Max. allowed time step size.

## 2.2 FluidBlocks

In this part the user can define multiple axis-aligned blocks of fluid particles.

Example code:

```
"FluidBlocks": [
  {
    "denseMode": 0,
    "start": [-2.0, 0.0, -1],
    "end": [-0.5, 1.5, 1],
    "translation": [1.0, 0.0, 0.0],
    "scale": [1, 1, 1]
  }
]
```

- start (vec3): Minimum coordinate of the box which defines the fluid block.
- end (vec3): Maximum coordinate of the box which defines the fluid block.
- translation (vec3): Translation vector of the block.
- scale (vec3): Scaling vector of the block.
- denseMode (int):
  - 0: regular sampling
  - 1: more dense sampling
  - 2: dense sampling
- initialVelocity (vec3): The initial velocity is set for all particles in the block.
- id (string): This id is used in the “Fluid parameter block” (see below) to define the properties of the fluid block. If no id is defined, then the standard id “Fluid” is used.

## 2.3 FluidModels

This part can be used to import one or more partio particle files in the scene.

Example code:

```
"FluidModels": [
  {
    "particleFile": "../models/bunny.bgeo",
    "translation": [-2.0, 0.1, 0.0],
    "rotationAxis": [0, 1, 0],
    "rotationAngle": 3.14159265359,
    "scale": 1
  }
]
```

- particleFile (string): Path of the partio file which contains the particle data.
- translation (vec3): Translation vector of the fluid model.
- scale (vec3): Scaling vector of the fluid model.
- rotationAxis (vec3): Axis used to rotate the particle data after loading.

- `rotationAngle` (float): Rotation angle for the initial rotation of the particle data.
- `id`: This id is used in the “Fluid parameter block” (see below) to define the properties of the fluid block. If no id is defined, then the standard id “Fluid” is used.

## 2.4 Emitters

In this part the user can define one or more emitters which generate fluid particles.

Example code:

```
"Emitters": [  
  {  
    "width": 5,  
    "height": 5,  
    "translation": [-1,0.75,0.0],  
    "rotationAxis": [0, 1, 0],  
    "rotationAngle": 3.1415926535897932384626433832795,  
    "velocity": 2,  
    "emitStartTime": 2,  
    "emitEndTime": 6,  
    "type": 0  
  }  
]
```

- `type` (int): Defines the shape of the emitter (default: 0).
  - 0: box
  - 1: circle
- `width` (int): Width of the box or radius of the circle emitter (default: 5).
- `height` (int): Height of the box (is only used for type 0) (default: 5).
- `translation` (vec3): Translation vector of the emitter (default: [0,0,0]).
- `rotationAxis` (vec3): Axis used to rotate the emitter. Note that in 2D simulations the axis is always set to [0,0,1] (default: [0,0,1]).
- `rotationAngle` (float): Rotation angle for the initial rotation of the emitter (default: 0).
- `velocity` (float): Initial velocity of the emitted particles in direction of the emitter (default: 1).
- `id`: This id is used in the “Fluid parameter block” (see below) to define the properties of the fluid block. If no id is defined, then the standard id “Fluid” is used (default: “Fluid”).
- `emitStartTime` (float): Start time of the emitter (default: 0).
- `emitEndTime` (float): End time of the emitter (default: REAL\_MAX).

## 2.5 RigidBodyes

Here, the static and dynamic rigid bodies are defined which define the boundary in the scene. In case of dynamic rigid bodies, the PositionBasedDynamics library is used for their simulation. Note that in this case the PositionBasedDynamics library also reads this json scene files and picks out the relevant parts. That means if you want to define for example a hinge joint or a motor, then just use the json format of PositionBasedDynamics in this scene file.

Example code:

```
"RigidBodyes": [
  {
    "geometryFile": "../models/UnitBox.obj",
    "translation": [0,2,0],
    "rotationAxis": [1, 0, 0],
    "rotationAngle": 0,
    "scale": [2.5, 4, 1.0],
    "color": [0.1, 0.4, 0.6, 1.0],
    "isDynamic": false,
    "isWall": true,
    "mapInvert": true,
    "mapThickness": 0.0,
    "mapResolution": [20,20,20],
    "samplingMode": 1
  }
]
```

- **geometryFile** (string): Path to a OBJ file which contains the geometry of the body.
- **particleFile** (string): Path to a partio file which contains a surface sampling of the body. Note that the surface sampling is done automatically if this parameter is missing.
- **translation** (vec3): Translation vector of the rigid body.
- **scale** (vec3): Scaling vector of the rigid body.
- **rotationAxis** (vec3): Axis used to rotate the rigid body after loading.
- **rotationAngle** (float): Rotation angle for the initial rotation of the rigid body.
- **isDynamic** (bool): Defines if the body is static or dynamic.
- **isWall** (bool): Defines if this is a wall. Walls are typically not rendered. This is the only difference.
- **color** (vec4): RGBA color of the body.
- **mapInvert** (bool): Invert the map when using density or volume maps, flips inside/outside (default: false)
- **mapThickness** (float): Additional thickness of a volume or density map (default: 0.0)
- **mapResolution** (vec3): Resolution of a volume or density map (default: [20,20,20])
- **samplingMode** (int): Surface sampling mode. 0 Poisson disk sampling, 1 Regular triangle sampling (default: 0).

## 2.6 Materials

```
"Materials": [  
  {  
    "id": "Fluid",  
    "density0": 1000,  
    "colorField": "velocity",  
    "colorMapType": 1,  
    "renderMinValue": 0.0,  
    "renderMaxValue": 5.0,  
    "surfaceTension": 0.2,  
    "surfaceTensionMethod": 0,  
    "viscosity": 0.01,  
    "viscosityMethod": 1,  
    "vorticityMethod": 1,  
    "vorticity": 0.15,  
    "viscosityOmega": 0.05,  
    "inertiaInverse": 0.5,  
    "maxEmitterParticles": 1000,  
    "emitterReuseParticles": false,  
    "emitterBoxMin": [-4.0, -1.0, -4.0],  
    "emitterBoxMax": [0.0, 4, 4.0]  
  }  
]
```

### 2.6.1 General

- **id (string):** Defines the id of the material. You have to give the same id to a FluidBlock, a FluidModel or an Emitter if they should have the defined material behavior.
- **density0 (float):** Rest density of the corresponding fluid.

### 2.6.2 Particle Coloring

- **colorField (string):** Choose vector or scalar field for particle coloring.
- **colorMapType (int):** Selection of a color map for coloring the scalar/vector field.
  - 0: None
  - 1: Jet
  - 2: Plasma
- **renderMinValue (float):** Minimal value used for color-coding the color field in the rendering process.
- **renderMaxValue (float):** Maximal value used for color-coding the color field in the rendering process.



### 2.6.3 Viscosity

- viscosityMethod (int): Viscosity method
  - 0: None
  - 1: Standard
  - 2: XSPH
  - 3: Bender and Koschier 2017
  - 4: Peer et al. 2015
  - 5: Peer et al. 2016
  - 6: Takahashi et al. 2015 (improved)
  - 7: Weiler et al. 2018
- viscosity (float): Coefficient for the viscosity force computation
- viscoMaxIter (int): (Implicit solvers) Max. iterations of the viscosity solver.
- viscoMaxError (float): (Implicit solvers) Max. error of the viscosity solver.
- viscoMaxIterOmega (int): (Peer et al. 2016) Max. iterations of the vorticity diffusion solver.
- viscoMaxErrorOmega (float): (Peer et al. 2016) Max. error of the vorticity diffusion solver.
- viscosityBoundary (float): (Weiler et al. 2018) Coefficient for the viscosity force computation at the boundary.

### 2.6.4 Vorticity

- vorticityMethod (int): Vorticity method
  - 0: None
  - 1: Micropolar model
  - 2: Vorticity confinement
- vorticity (float): Coefficient for the vorticity force computation
- viscosityOmega (float): (Micropolar model) Viscosity coefficient for the angular velocity field.
- inertiaInverse (float): (Micropolar model) Inverse microinertia used in the micropolar model.

### 2.6.5 Drag force

- dragMethod (int): Drag force method
  - 0: None
  - 1: Macklin et al. 2014
  - 2: Gissler et al. 2017
- drag (float): Coefficient for the drag force computation

### 2.6.6 Surface tension

- `surfaceTensionMethod` (int): Surface tension method
  - 0: None
  - 1: Becker & Teschner 2007
  - 2: Akinici et al. 2013
  - 3: He et al. 2014
- `surfaceTension` (float): Coefficient for the surface tension computation

### 2.6.7 Elasticity

- `elasticityMethod` (int): Elasticity method
  - 0: None
  - 1: Becker et al. 2009
  - 2: Peer et al. 2018
- `youngsModulus` (float): Young’s modulus - coefficient for the stiffness of the material (default: 100000.0)
- `poissonsRatio` (float): Poisson’s ratio - measure of the Poisson effect (default: 0.3)
- `alpha` (float): Coefficient for zero-energy modes suppression method (default: 0.0)
- `elasticityMaxIter` (float): (Peer et al. 2018) Maximum solver iterations (default: 100)
- `elasticityMaxError` (float): (Peer et al. 2019) Maximum elasticity error allowed by the solver (default: 1.0e-4)

### 2.6.8 Emitters

- `maxEmitterParticles` (int): Maximum number of particles the emitter generates. Note that reused particles (see below) are not counted here.
- `emitterReuseParticles` (bool): Reuse particles if they are outside of the bounding box defined by `emitterBoxMin`, `emitterBoxMax`
- `emitterBoxMin` (vec3): Minimum coordinates of an axis-aligned box (used in combination with `emitterReuseParticles`)
- `emitterBoxMax` (vec3): Maximum coordinates of an axis-aligned box (used in combination with `emitterReuseParticles`)

## 2.7 Animation fields

In this part the user can define one or more animation fields which animate fluid particles. The user can define math expressions for the components of the field quantity. The typical math terms like `cos`, `sin`, `...` can be used.

Available expression variables:

- `t`: Current time.
- `dt`: Current time step size.
- `x`, `y`, `z`: Position of the particle which is in the animation field.

- vx, vy, vz: Velocity of the particle which is in the animation field.
- valutex, valuey, valuez: Value of the field quantity of the particle which is in the animation field.

Example:

```
"particleField": "angular velocity",
"expression_x": "valutex + cos(2*t)"
```

This means that in each step we add  $\cos(2t)$  to the x-component of the angular velocity.

Example code:

```
"AnimationFields": [
  {
    "particleField": "velocity",
    "translation": [-0.5, -0.5, 0],
    "rotationAxis": [0, 0, 1],
    "rotationAngle": 0.0,
    "scale": [0.5, 0.25, 0.8],
    "shapeType": 0,
    "expression_x": "cos(2*t)*0.1",
    "expression_y": "",
    "expression_z": ""
  }
]
```

- shapeType (int): Defines the shape of the animation field (default: 0).
  - 0: box
  - 1: sphere
  - 2: cylinder
- particleField (string): Defines the field quantity that should be modified by the field (e.g. velocity, angular velocity, position) (default: velocity)
- translation (vec3): Translation vector of the animation field (default: [0,0,0]).
- rotationAxis (vec3): Axis used to rotate the animation field (default: [0,0,1]).
- rotationAngle (float): Rotation angle for the initial rotation of the animation field (default: 0).
- scale (vec3): Scaling vector of the animation field.
  - shapeType=0 (box): This vector defines the width, height, depth of the box.
  - shapeType=1 (sphere): The x-component of the vector defines the radius of the sphere. The other components are ignored.
  - shapeType=2 (cylinder): The x- and y-component of the vector defines the height and radius of the cylinder, respectively. The z-component is ignored.
- expression\_x (string): Math expression for the x-component of the field quantity (default="").
- expression\_y (string): Math expression for the y-component of the field quantity (default="").
- expression\_z (string): Math expression for the z-component of the field quantity (default="").



## PYSPLISHSPLASH

### 3.1 Python bindings for the SPLisHSPlasH library

### 3.2 Requirements

Currently the generation of python bindings is only tested on

- Linux Debian, gcc 8.3, Python 3.7/3.8 (Anaconda), CMake 3.13
- Windows 10, Visual Studio 15/17/19, Python 3.7/3.8 (Anaconda), CMake 3.13

Note that the compiler, the python installation as well as cmake have to be available from the command line for the installation process to work. MacOS builds should work but have not been tested.

### 3.3 Installation

In order to install it is advised that you create a new virtual environment so that any faults during installation can not mess up your python installation. This is done as follows for

#### conda

```
conda create --name venv python=3.7
conda activate venv
```

#### virtualenv

```
python3 -m virtualenv venv --python=python3.7
source venv/bin/activate
```

Now you can clone the repository by

```
git clone https://github.com/InteractiveComputerGraphics/SPlisHSPlasH.git
```

And finally you should be able to install SPLisHSPlasH using pip. **The trailing slash is important** otherwise pip will try to download the package, which is not supported yet at least. Also note, that `pip install SPLisHSPlasH` should be called from **one directory above** the cloned source directory and **not within** the directory itself.

```
pip install SPLisHSPlasH/
```

While `pip install` is useful if SPLisHSPlasH should only be installed once, for development purposes it might be more sensible to build differently. Change into the SPLisHSPlasH directory and build a python wheel file as follows

```
cd SPlisHSPlasH
python setup.py bdist_wheel
pip install -I build/dist/*.whl
```

When building a new version of SPlisHSPlasH simply run these commands again and the installation will be updated. The compile times will be lower, because the build files from previous installations remain. If you are getting compile errors please try to compile the pysplishsplash target of the CMake project separately.

Now check your installation by running

```
python -c "import pysplishsplash"
```

**Note:** You may have to install numpy. Future releases may already contain numpy as a dependency.

```
pip install numpy
```

## 3.4 I want to see something very very quickly

If you're very impatient, just run the following command after installing

```
splash
```

You will be prompted to select a preconfigured scene file which will then be run in a User Interface. For more options and functionality run. The keybindings in the GUI are the same as for the regular SPlisHSPlasH version.

```
splash --help
```

## 3.5 Minimal working example

The following examples should work, if SPlisHSPlasH was installed correctly. If you want to load other scene files, be sure to place them into the SPlisHSPlasH data directory structure.

### With GUI

```
import pysplishsplash as sph

def main():
    base = sph.Exec.SimulatorBase()
    base.init()
    gui = sph.GUI.Simulator_GUI_TweakBar(base)
    base.setGui(gui)
    base.run()

if __name__ == "__main__":
    main()
```

### Without GUI

```
import pysplishsplash as sph

def main():
    base = sph.Exec.SimulatorBase()
```

(continues on next page)

(continued from previous page)

```

base.init(useGui=False)
base.setValueFloat(base.STOP_AT, 10.0) # Important to have the dot to denote a_
↪float
base.run()

if __name__ == "__main__":
    main()

```

### Outputting the results to a specific directory without GUI

```

import pysplishsplash as sph
from pysplishsplash.Extras import Scenes
import os

def main():
    base = sph.Exec.SimulatorBase()
    output_dir = os.path.abspath("where/you/want/the/data")
    base.init(useGui=False, outputDir=output_dir, sceneFile=Scenes.DoubleDamBreak)
    base.setValueFloat(base.STOP_AT, 20.0) # Important to have the dot to denote a_
↪float
    base.setValueBool(base.VTK_EXPORT, True)
    # Uncomment the next line to set the output FPS value (must be float)
    # base.setValueFloat(base.DATA_EXPORT_FPS, 10000.)
    base.run()

if __name__ == "__main__":
    main()

```

## 3.6 SPHSimulator.py

If you want to start the simulator in the same way as the C++ version, just use the SPHSimulator.py in the examples directory.

## 3.7 Modifying other properties

The bindings cover most of the public interface of the SPlisHSPlasH library. As such, it is possible to change components of the simulation dynamically. In the following example, the second cube in the well known double dam break scenario is replaced with a slightly larger cube.

```

import pysplishsplash
import pysplishsplash.Utilities.SceneLoaderStructs as Scene

def main():
    base = pysplishsplash.Exec.SimulatorBase()
    args = base.init()
    gui = pysplishsplash.GUI.Simulator_GUI_TweakBar(base)
    base.setGui(gui)
    scene = base.getScene()
    add_block = Scene.FluidBlock('Fluid', Scene.Box([0.0, 0.0, 0.0], [1.0, 1.0, 1.0]),
↪ 0, [0.0, 0.0, 0.0])
    scene.fluidBlocks[1] = add_block # In Place construction not supported yet
    base.run()

```

(continues on next page)

(continued from previous page)

```
if __name__ == "__main__":  
    main()
```



## CREATING SCENES

### 4.1 Loading the empty scene

Right now the easiest way to create a custom scene without specifying a `Scene.json` file, is to load the predefined empty scene.

```
import pysplishsplash as sph
import pysplishsplash.Utilities.SceneLoaderStructs as Scenes

base = sph.Exec.SimulatorBase()
base.init(sceneFile=Scenes.Empty)
```

This scene will set the default simulation method to be DFSPH and some other default values, which can all be changed later on.

### 4.2 Recreating the double dam break scenario

In order to recreate the double dam break scenario, we need to add a bounding box as well as two fluid cubes. The bounding box can be added as follows

```
scene = base.getScene()
scene.boundaryModels.append(Scenes.BoundaryData(meshFile="../models/UnitBox.obj",
↪ translation=[0., 3.0, 0.], scale=[4., 6., 4.], color=[0.1, 0.4, 0.5, 1.0],
↪ isWall=True, mapInvert=True, mapResolution=[25, 25, 25]))
```

The two fluid blocks can at the end be added using

```
scene.fluidBlocks.append(Scenes.FluidBlock(id='Fluid', box=Scenes.Box([-1.5, 0.0, -1.
↪ 5], [-0.5, 2.0, -0.5]), mode=0, initialVelocity=[0.0, 0.0, 0.0]))
scene.fluidBlocks.append(Scenes.FluidBlock(id='Fluid', box=Scenes.Box([0.5, 0.0, 0.5],
↪ [1.5, 2.0, 1.5]), mode=0, initialVelocity=[0.0, 0.0, 0.0]))
```

This will recreate a somewhat larger scene than the default double dam break

## 4.3 Putting it all together

The following shows a script detailing how to build and run a custom double dam break. Follow the instruction from before to activate/ deactivate the GUI.

```
import pysplishsplash as sph
import pysplishsplash.Utilities.SceneLoaderStructs as Scenes

def main():
    # Set up the simulator
    base = sph.Exec.SimulatorBase()
    base.init(useGui=True, sceneFile=sph.Extras.Scenes.Empty)

    # Create a tweak bar simulator
    gui = sph.GUI.Simulator_GUI_TweakBar(base)
    base.setGui(gui)

    # Get the scene and add objects
    scene = base.getScene()
    scene.boundaryModels.append(Scenes.BoundaryData(meshFile="../models/UnitBox.obj",
    ↪translation=[0., 3.0, 0.], scale=[4., 6., 4.], color=[0.1, 0.4, 0.5, 1.0],
    ↪isWall=True, mapInvert=True, mapResolution=[25, 25, 25]))
    scene.fluidBlocks.append(Scenes.FluidBlock(id='Fluid', box=Scenes.Box([-1.5, 0.0,
    ↪-1.5], [-0.5, 2.0, -0.5]), mode=0, initialVelocity=[0.0, 0.0, 0.0]))
    scene.fluidBlocks.append(Scenes.FluidBlock(id='Fluid', box=Scenes.Box([0.5, 0.0,
    ↪0.5], [1.5, 2.0, 1.5]), mode=0, initialVelocity=[0.0, 0.0, 0.0]))

    # Run the GUI
    base.run()

if __name__ == "__main__":
    main()
```

## 4.4 Loading a scene from file

Loading a scene from a file is as simple as simply specifying a custom scene file in the init function. This must be an **absolute path**!

```
custom_scene = os.path.abspath("scene.json")
base.init(sceneFile=custom_scene)
```

If you want to use a gui to locate the scene file you may want to use tkinter

```
import tkinter as tk
from tkinter import filedialog

tk.Tk().withdraw() # Dont show main window
custom_scene = filedialog.askopenfilename()
base.init(sceneFile=custom_scene)
```

## RESTRICTIONS

- When modifying simulation parameters this is the recommended structure, as modification will only work after `base.initSimulation()` has been called.

```
base.initSimulation()  
sim = sph.Simulation.getCurrent()  
sim.setValue...()  
base.runSimulation()  
base.cleanup()
```

- `setValue...()` and `getValue...()` functions cannot accept vectors as arguments yet



## LIBRARY API

### 6.1 Class Hierarchy

### 6.2 File Hierarchy

### 6.3 Full API

#### 6.3.1 Namespaces

Namespace @54

Namespace Eigen

##### Contents

- *Namespaces*

#### Namespaces

- *Namespace Eigen::internal*

#### Namespace Eigen::internal

##### Contents

- *Classes*

## Classes

- *Template Struct generic\_product\_impl< MatrixReplacement, Rhs, SparseShape, DenseShape, GemvProduct >*
- *Template Struct traits< SPH::MatrixReplacement >*

## Namespace GenParam

## Namespace SPH

### Contents

- *Classes*
- *Enums*
- *Variables*

## Classes

- *Struct FieldDescription*
- *Struct PoissonDiskSampling::CellPosHasher*
- *Struct PoissonDiskSampling::HashEntry*
- *Struct PoissonDiskSampling::InitialPointInfo*
- *Class AdhesionKernel*
- *Class AnimationField*
- *Class AnimationFieldSystem*
- *Class BlockJacobiPreconditioner3D*
- *Class BoundaryModel*
- *Class BoundaryModel\_Akinci2012*
- *Class BoundaryModel\_Bender2019*
- *Class BoundaryModel\_Koschier2017*
- *Class CohesionKernel*
- *Class CubicKernel*
- *Class CubicKernel2D*
- *Class DragBase*
- *Class DragForce\_Gissler2017*
- *Class DragForce\_Macklin2014*
- *Class Elasticity\_Becker2009*
- *Class Elasticity\_Peer2018*
- *Class ElasticityBase*

- *Class Emitter*
- *Class EmitterSystem*
- *Class FluidModel*
- *Class GaussQuadrature*
- *Class JacobiPreconditioner1D*
- *Class JacobiPreconditioner3D*
- *Class MathFunctions*
- *Class MatrixReplacement*
- *Class MicropolarModel\_Bender2017*
- *Class NonPressureForceBase*
- *Class PoissonDiskSampling*
- *Class Poly6Kernel*
- *Template Class PrecomputedKernel*
- *Class RegularSampling2D*
- *Class RegularTriangleSampling*
- *Class RigidBodyObject*
- *Class SimpleQuadrature*
- *Class Simulation*
- *Class SimulationDataDFSPH*
- *Class SimulationDataIISPH*
- *Class SimulationDataPBF*
- *Class SimulationDataPCISPH*
- *Class SimulationDataPF*
- *Class SimulationDataWCSPH*
- *Class SpikyKernel*
- *Class StaticRigidBody*
- *Class SurfaceTension\_Akinci2013*
- *Class SurfaceTension\_Becker2007*
- *Class SurfaceTension\_He2014*
- *Class SurfaceTensionBase*
- *Class TimeIntegration*
- *Class TimeManager*
- *Class TimeStep*
- *Class TimeStepDFSPH*
- *Class TimeStepIISPH*
- *Class TimeStepPBF*

- *Class TimeStepPCISPH*
- *Class TimeStepPF*
- *Class TimeStepWCSPH*
- *Class TriangleMesh*
- *Class Viscosity\_Bender2017*
- *Class Viscosity\_Peer2015*
- *Class Viscosity\_Peer2016*
- *Class Viscosity\_Standard*
- *Class Viscosity\_Takahashi2015*
- *Class Viscosity\_Weiler2018*
- *Class Viscosity\_XSPH*
- *Class ViscosityBase*
- *Class VorticityBase*
- *Class VorticityConfinement*
- *Class WendlandQuinticC2Kernel*
- *Class WendlandQuinticC2Kernel2D*

## **Enums**

- *Enum BoundaryHandlingMethods*
- *Enum DragMethods*
- *Enum ElasticityMethods*
- *Enum FieldType*
- *Enum ParticleState*
- *Enum SimulationMethods*
- *Enum SurfaceSamplingMode*
- *Enum SurfaceTensionMethods*
- *Enum ViscosityMethods*
- *Enum VorticityMethods*

## **Variables**

- *Variable SPH::gaussian\_abscissae\_1*
- *Variable SPH::gaussian\_n\_1*
- *Variable SPH::gaussian\_weights\_1*



## Namespace std

## Namespace Utilities

### Contents

- *Classes*

## Classes

- *Struct SceneLoader::AnimationFieldData*
- *Struct SceneLoader::BoundaryData*
- *Struct SceneLoader::Box*
- *Struct SceneLoader::EmitterData*
- *Struct SceneLoader::FluidBlock*
- *Struct SceneLoader::FluidData*
- *Struct SceneLoader::MaterialData*
- *Struct SceneLoader::Scene*
- *Class SceneLoader*
- *Class SDFFunctions*
- *Class VolumeSampling*
- *Class WindingNumbers*

## 6.3.2 Classes and Structs

**Template Struct** `generic_product_impl< MatrixReplacement, Rhs, SparseShape, DenseShape, GemvProduct >`

- Defined in file `_SPlisHSPlasH_Uilities_MatrixFreeSolver.h`

## Inheritance Relationships

### Base Type

- `public generic_product_impl_base< MatrixReplacement, Rhs, generic_product_impl< MatrixReplacement, Rhs > >`

## Struct Documentation

template<typename **Rhs**>

**struct** Eigen::internal::generic\_product\_impl<MatrixReplacement, *Rhs*, SparseShape, DenseShape, GemvProduct>  
Implementation of the matrix-free matrix vector product

### Public Types

**typedef** Product<MatrixReplacement, *Rhs*>::Scalar **Scalar**

### Public Static Functions

template<typename **Dest**>

void **scaleAndAddTo** (*Dest* &*dst*, const MatrixReplacement &*lhs*, const *Rhs* &*rhs*, const *Scalar* &*alpha*)

## Template Struct traits< SPH::MatrixReplacement >

- Defined in file\_SPlisHSPlasH\_Uilities\_MatrixFreeSolver.h

## Inheritance Relationships

### Base Type

- public Eigen::internal::traits< SystemMatrixType >

## Struct Documentation

template<>

**struct** traits<SPH::MatrixReplacement> : public Eigen::internal::traits<SystemMatrixType>

### Struct FieldDescription

- Defined in file\_SPlisHSPlasH\_FluidModel.h

## Struct Documentation

**struct** SPH::FieldDescription

### Public Functions

```
FieldDescription (const std::string &n, const FieldType &t, const
                  std::function<void*> const unsigned int
                  > &fct, const bool s = false
```

### Public Members

```
std::string name
FieldType type
std::function<void*> (const unsigned int) > getFct
bool storeData
```

### Struct PoissonDiskSampling::CellPosHasher

- Defined in file\_SPlisHSPlasH\_Uilities\_PoissonDiskSampling.h

### Nested Relationships

This struct is a nested type of *Class PoissonDiskSampling*.

### Struct Documentation

```
struct SPH::PoissonDiskSampling::CellPosHasher
```

### Public Functions

```
std::size_t operator () (const CellPos &k) const
```

### Struct PoissonDiskSampling::HashEntry

- Defined in file\_SPlisHSPlasH\_Uilities\_PoissonDiskSampling.h

### Nested Relationships

This struct is a nested type of *Class PoissonDiskSampling*.

## Struct Documentation

**struct** SPH::*PoissonDiskSampling*::**HashEntry**  
 Struct to store the hash entry (spatial hashing)

### Public Functions

**HashEntry** ( )

### Public Members

std::vector<unsigned int> **samples**  
 unsigned int **startIndex**

## Struct PoissonDiskSampling::InitialPointInfo

- Defined in file\_SPlisHSPlasH\_Uilities\_PoissonDiskSampling.h

## Nested Relationships

This struct is a nested type of *Class PoissonDiskSampling*.

## Struct Documentation

**struct** SPH::*PoissonDiskSampling*::**InitialPointInfo**  
 Struct to store the information of the initial points.

### Public Members

CellPos **cP**  
*Vector3r* **pos**  
 unsigned int **ID**

## Struct SceneLoader::AnimationFieldData

- Defined in file\_SPlisHSPlasH\_Uilities\_SceneLoader.h

## Nested Relationships

This struct is a nested type of *Class SceneLoader*.

## Struct Documentation

**struct** Utilities::*SceneLoader*::**AnimationFieldData**  
Struct to store an animation field object.

### Public Members

std::string **particleFieldName**

std::string **expression**[3]

unsigned int **shapeType**

*Vector3r* **x**

*Matrix3r* **rotation**

*Vector3r* **scale**

*Real* **startTime**

*Real* **endTime**

## Struct SceneLoader::BoundaryData

- Defined in file\_SPlisHSPlasH\_Uutilities\_SceneLoader.h

## Nested Relationships

This struct is a nested type of *Class SceneLoader*.

## Struct Documentation

**struct** Utilities::*SceneLoader*::**BoundaryData**  
Struct to store a boundary object.

### Public Members

std::string **samplesFile**

std::string **meshFile**

*Vector3r* **translation**

*Matrix3r* **rotation**

*Vector3r* **scale**

*Real* **density**

bool **dynamic**

```
bool isWall  
Eigen::Matrix<float, 4, 1, Eigen::DontAlign> color  
void *rigidBody  
std::string mapFile  
bool mapInvert  
Real mapThickness  
Eigen::Matrix<unsigned int, 3, 1, Eigen::DontAlign> mapResolution  
unsigned int samplingMode
```

### Struct SceneLoader::Box

- Defined in file\_SPlisHSPlasH\_Uilities\_SceneLoader.h

### Nested Relationships

This struct is a nested type of *Class SceneLoader*.

### Struct Documentation

```
struct Utilities::SceneLoader::Box  
    Struct for an AABB.
```

### Public Members

```
Vector3r m_minX  
Vector3r m_maxX
```

### Struct SceneLoader::EmitterData

- Defined in file\_SPlisHSPlasH\_Uilities\_SceneLoader.h

### Nested Relationships

This struct is a nested type of *Class SceneLoader*.

## Struct Documentation

**struct** Utilities::*SceneLoader*::**EmitterData**  
Struct to store an emitter object.

### Public Members

std::string **id**  
 unsigned int **width**  
 unsigned int **height**  
*Vector3r* **x**  
*Real* **velocity**  
*Matrix3r* **rotation**  
*Real* **emitStartTime**  
*Real* **emitEndTime**  
 unsigned int **type**

## Struct SceneLoader::FluidBlock

- Defined in file `_SPlisHSPlasH_Uilities_SceneLoader.h`

## Nested Relationships

This struct is a nested type of *Class SceneLoader*.

## Struct Documentation

**struct** Utilities::*SceneLoader*::**FluidBlock**  
Struct to store a fluid block.

### Public Members

std::string **id**  
*Box* **box**  
 unsigned char **mode**  
*Vector3r* **initialVelocity**

### Struct SceneLoader::FluidData

- Defined in file\_SPlisHSPlasH\_Uilities\_SceneLoader.h

### Nested Relationships

This struct is a nested type of *Class SceneLoader*.

### Struct Documentation

**struct** Utilities::SceneLoader::FluidData  
Struct to store a fluid object.

#### Public Members

std::string **id**  
std::string **samplesFile**  
*Vector3r* **translation**  
*Matrix3r* **rotation**  
*Vector3r* **scale**  
*Vector3r* **initialVelocity**  
unsigned char **mode**  
bool **invert**  
std::array<unsigned int, 3> **resolutionSDF**

### Struct SceneLoader::MaterialData

- Defined in file\_SPlisHSPlasH\_Uilities\_SceneLoader.h

### Nested Relationships

This struct is a nested type of *Class SceneLoader*.

### Struct Documentation

**struct** Utilities::SceneLoader::MaterialData  
Struct to store particle coloring information.



### Public Members

```

std::string id
std::string colorField
unsigned int colorMapType
Real minVal
Real maxVal
unsigned int maxEmitterParticles
bool emitterReuseParticles
Vector3r emitterBoxMin
Vector3r emitterBoxMax

```

### Struct SceneLoader::Scene

- Defined in file `_SPlisHSPlasH_Uilities_SceneLoader.h`

### Nested Relationships

This struct is a nested type of *Class SceneLoader*.

### Struct Documentation

```
struct Utilities::SceneLoader::Scene
    Struct to store scene information.
```

### Public Members

```

std::vector<BoundaryData*> boundaryModels
std::vector<FluidData*> fluidModels
std::vector<FluidBlock*> fluidBlocks
std::vector<EmitterData*> emitters
std::vector<AnimationFieldData*> animatedFields
std::vector<MaterialData*> materials
Real particleRadius
bool sim2D
Real timeStepSize
Vector3r camPosition
Vector3r camLookat

```

## Class Matrix3f8

- Defined in file\_SPlisHSPlasH\_Uutilities\_AVX\_math.h

## Class Documentation

**class** **Matrix3f8**

### Public Functions

```
Matrix3f8 ()  
Matrix3f8 (const Vector3f8 &m1, const Vector3f8 &m2, const Vector3f8 &m3)  
void setZero ()  
Scalarf8 &operator () (int i, int j)  
void setCol (int i, const Vector3f8 &v)  
void setCol (int i, const Scalarf8 &x, const Scalarf8 &y, const Scalarf8 &z)  
Matrix3f8 operator* (const Scalarf8 &b) const  
Vector3f8 operator* (const Vector3f8 &b) const  
Matrix3f8 operator* (const Matrix3f8 &b) const  
Matrix3f8 &operator+= (const Matrix3f8 &a)  
Matrix3f8 transpose () const  
Scalarf8 determinant () const  
void store (std::vector<Matrix3r> &Mf) const  
Matrix3r reduce () const
```

### Public Members

```
Scalarf8 m[3][3]
```

## Class Quaternion8f

- Defined in file\_SPlisHSPlasH\_Uutilities\_AVX\_math.h

## Class Documentation

**class** **Quaternion8f**

## Public Functions

```

Quaternion8f ()
Quaternion8f (Scalarf8 x, Scalarf8 y, Scalarf8 z, Scalarf8 w)
Quaternion8f (Vector3f8 &v)
Scalarf8 &operator[] (int i)
Scalarf8 operator[] (int i) const
Scalarf8 &x ()
Scalarf8 &y ()
Scalarf8 &z ()
Scalarf8 &w ()
Scalarf8 x () const
Scalarf8 y () const
Scalarf8 z () const
Scalarf8 w () const
const Quaternion8f operator* (const Quaternion8f &a) const
void toRotationMatrix (Matrix3f8 &R)
void toRotationMatrix (Vector3f8 &R1, Vector3f8 &R2, Vector3f8 &R3)
void store (std::vector<Quaternionr> &qf) const
void set (const std::vector<Quaternionr> &qf)

```

## Public Members

```
Scalarf8 q[4]
```

## Class Scalarf8

- Defined in file\_SPlisHSPlasH\_Utilityes\_AVX\_math.h

## Class Documentation

```
class Scalarf8
```

## Public Functions

```
Scalarf8 ()  
Scalarf8 (float f)  
Scalarf8 (Real f0, Real f1, Real f2, Real f3, Real f4, Real f5, Real f6, Real f7)  
Scalarf8 (float const *p)  
Scalarf8 (__m256 const &x)  
Scalarf8 &operator= (__m256 const &x)  
Scalarf8 sqrt () const  
Scalarf8 rsqrt () const  
Scalarf8 &load (float const *p)  
void store (float *p) const  
float reduce () const
```

## Public Members

```
__m256 v
```

## Class AdhesionKernel

- Defined in file\_SPlisHSPlasH\_SPHKernels.h

## Class Documentation

```
class SPH::AdhesionKernel
```

Adhesion kernel used for the surface tension method of Akinci et al. [2].

## Public Static Functions

```
Real getRadius ()  
void setRadius (Real val)  
Real W (const Real r)  
     $W(r,h) = (0.007/h^{3.25})(-4r^2/h + 6r - 2h)^{0.25}$  if  $h/2 < r \leq h$   
Real W (const Vector3r &r)  
Real W_zero ()
```

### Protected Static Attributes

*Real* **m\_radius**

*Real* **m\_k**

*Real* **m\_W\_zero**

### Class AnimationField

- Defined in file\_SPlisHSPlasH\_AnimationField.h

### Class Documentation

**class** SPH::AnimationField

#### Public Functions

**AnimationField** (**const** std::string &particleFieldName, **const** *Vector3r* &pos, **const** *Matrix3r* &rotation, **const** *Vector3r* &scale, **const** std::string expression[3], **const** unsigned int type = 0)

**~AnimationField** ()

void **setStartTime** (*Real* val)

void **setEndTime** (*Real* val)

void **step** ()

void **reset** ()

#### Protected Functions

**FORCE\_INLINE** bool inBox (const *Vector3r* &x, const *Vector3r* &xBox, const *Matrix3r* &rotB

**FORCE\_INLINE** bool inCylinder (const *Vector3r* &x, const *Vector3r* &xCyl, const *Matrix3r* &rotB

**FORCE\_INLINE** bool inSphere (const *Vector3r* &x, const *Vector3r* &pos, const *Matrix3r* &rotB

**FORCE\_INLINE** bool inShape (const int type, const *Vector3r* &x, const *Vector3r* &pos, const

#### Protected Attributes

std::string **m\_particleFieldName**

*Vector3r* **m\_x**

*Matrix3r* **m\_rotation**

*Vector3r* **m\_scale**

std::string **m\_expression**[3]

unsigned int **m\_type**

*Real* **m\_startTime**

*Real* **m\_endTime**

## Class AnimationFieldSystem

- Defined in file\_SPlisHSPlasH\_AnimationFieldSystem.h

## Class Documentation

**class** SPH::AnimationFieldSystem

### Public Functions

**AnimationFieldSystem**()

**~AnimationFieldSystem**()

void **addAnimationField**(**const** std::string &*particleFieldName*, **const** *Vector3r* &*pos*, **const** *Matrix3r* &*rotation*, **const** *Vector3r* &*scale*, **const** std::string *expression*[3], **const** unsigned int *type*)

unsigned int **numAnimationFields**() **const**

std::vector<*AnimationField*\*> &**getAnimationFields**()

void **step**()

void **reset**()

### Protected Attributes

std::vector<*AnimationField*\*> **m\_fields**

## Class BlockJacobiPreconditioner3D

- Defined in file\_SPlisHSPlasH\_Uutilities\_MatrixFreeSolver.h

## Class Documentation

**class** SPH::BlockJacobiPreconditioner3D

Matrix-free 3x3 block Jacobi preconditioner

### Public Types

**enum** [anonymous]

*Values:*

**enumerator** **ColsAtCompileTime** = Eigen::Dynamic

**enumerator** **MaxColsAtCompileTime** = Eigen::Dynamic

**typedef** *SystemMatrixType*::StorageIndex **StorageIndex**

**typedef** void (\***DiagonalMatrixElementFct**)(**const** unsigned int, *Matrix3r*&, void\*)

## Public Functions

**BlockJacobiPreconditioner3D** ()

void **init** (const unsigned int *dim*, *DiagonalMatrixElementFct* *fct*, void \**userData*)

Eigen::Index **rows** () const

Eigen::Index **cols** () const

Eigen::ComputationInfo **info** ()

template<typename **MatType**>

*BlockJacobiPreconditioner3D* &**analyzePattern** (const *MatType*&)

template<typename **MatType**>

*BlockJacobiPreconditioner3D* &**factorize** (const *MatType* &*mat*)

template<typename **MatType**>

*BlockJacobiPreconditioner3D* &**compute** (const *MatType* &*mat*)

template<typename **Rhs**, typename **Dest**>

void **\_solve\_impl** (const *Rhs* &*b*, *Dest* &*x*) const

template<typename **Rhs**>

const Eigen::Solve<*BlockJacobiPreconditioner3D*, *Rhs*> **solve** (const Eigen::MatrixBase<*Rhs*> &*b*) const

## Protected Attributes

unsigned int **m\_dim**

*DiagonalMatrixElementFct* **m\_diagonalElementFct**  
diagonal matrix element callback

void \***m\_userData**

std::vector<*Matrix3r*> **m\_invDiag**

## Class BoundaryModel

- Defined in file `_SPlisHSPlasH_BoundaryModel.h`

## Inheritance Relationships

## Derived Types

- public SPH::BoundaryModel\_Akinci2012 (*Class BoundaryModel\_Akinci2012*)
- public SPH::BoundaryModel\_Bender2019 (*Class BoundaryModel\_Bender2019*)
- public SPH::BoundaryModel\_Koschier2017 (*Class BoundaryModel\_Koschier2017*)

## Class Documentation

### **class** SPH::BoundaryModel

The boundary model stores the information required for boundary handling.

Subclassed by *SPH::BoundaryModel\_Akinci2012*, *SPH::BoundaryModel\_Bender2019*,  
*SPH::BoundaryModel\_Koschier2017*

### Public Functions

**BoundaryModel** ()

**~BoundaryModel** ()

void **reset** ()

void **performNeighborhoodSearchSort** ()

void **saveState** (BinaryFileWriter &*binWriter*)

void **loadState** (BinaryFileReader &*binReader*)

*RigidBodyObject* \***getRigidBodyObject** ()

**FORCE\_INLINE** void **addForce** (const Vector3r &*pos*, const Vector3r &*f*)

**FORCE\_INLINE** void **getPointVelocity** (const Vector3r &*x*, Vector3r &*res*)

void **getForceAndTorque** (*Vector3r* &*force*, *Vector3r* &*torque*)

void **clearForceAndTorque** ()

### Protected Attributes

*RigidBodyObject* \***m\_rigidBody**

std::vector<*Vector3r*> **m\_forcePerThread**

std::vector<*Vector3r*> **m\_torquePerThread**

### Class BoundaryModel\_Akinci2012

- Defined in file *\_SPlisHSPlasH\_BoundaryModel\_Akinci2012.h*

## Inheritance Relationships

### Base Type

- public SPH::BoundaryModel (*Class BoundaryModel*)



## Class Documentation

**class** SPH::BoundaryModel\_Akinci2012 : public SPH::BoundaryModel

The boundary model stores the information required for boundary handling using the approach of Akinci et al. 2012 [1].

### Public Functions

```
BoundaryModel_Akinci2012 ()
~BoundaryModel_Akinci2012 ()
unsigned int numberOfParticles () const
unsigned int getPointSetIndex () const
void computeBoundaryVolume ()
void resize (const unsigned int numBoundaryParticles)
void reset ()
void performNeighborhoodSearchSort ()
void saveState (BinaryFileWriter &binWriter)
void loadState (BinaryFileReader &binReader)
void initModel (RigidBodyObject *rbo, const unsigned int numBoundaryParticles, Vector3r
               *boundaryParticles)
FORCE_INLINE Vector3r & getPosition0 (const unsigned int i)
FORCE_INLINE const Vector3r & getPosition0 (const unsigned int i) const
FORCE_INLINE void setPosition0 (const unsigned int i, const Vector3r &pos)
FORCE_INLINE Vector3r & getPosition (const unsigned int i)
FORCE_INLINE const Vector3r & getPosition (const unsigned int i) const
FORCE_INLINE void setPosition (const unsigned int i, const Vector3r &pos)
FORCE_INLINE Vector3r & getVelocity (const unsigned int i)
FORCE_INLINE const Vector3r & getVelocity (const unsigned int i) const
FORCE_INLINE void setVelocity (const unsigned int i, const Vector3r &vel)
FORCE_INLINE const Real & getVolume (const unsigned int i) const
FORCE_INLINE Real & getVolume (const unsigned int i)
FORCE_INLINE void setVolume (const unsigned int i, const Real &val)
```

### Protected Attributes

```
bool m_sorted
unsigned int m_pointSetIndex
std::vector<Vector3r> m_x0
std::vector<Vector3r> m_x
std::vector<Vector3r> m_v
std::vector<Real> m_V
```

### Class BoundaryModel\_Bender2019

- Defined in file `_SPlisHSPlasH_BoundaryModel_Bender2019.h`

### Inheritance Relationships

#### Base Type

- `public SPH::BoundaryModel` (*Class BoundaryModel*)

### Class Documentation

```
class SPH::BoundaryModel_Bender2019 : public SPH::BoundaryModel
```

The boundary model stores the information required for boundary handling using the approach of Bender et al. 2019 [10].

### Public Functions

```
BoundaryModel_Bender2019 ()
~BoundaryModel_Bender2019 ()
void initModel (RigidBodyObject *rbo)
void reset ()
Discregrid::DiscreteGrid *getMap ()
void setMap (Discregrid::DiscreteGrid *map)
Real getMaxDist () const
void setMaxDist (Real val)
Real getMaxVel () const
void setMaxVel (Real val)
```

```
FORCE_INLINE const Real & getBoundaryVolume (const unsigned int fluidIndex, const unsigned int cellIndex) const
FORCE_INLINE Real & getBoundaryVolume (const unsigned int fluidIndex, const unsigned int cellIndex)
FORCE_INLINE void setBoundaryVolume (const unsigned int fluidIndex, const unsigned int cellIndex, const Real val)
FORCE_INLINE Vector3r & getBoundaryXj (const unsigned int fluidIndex, const unsigned int cellIndex)
```

```
FORCE_INLINE const Vector3r & getBoundaryXj (const unsigned int fluidIndex, const unsigned int i, const unsigned int j)
FORCE_INLINE void setBoundaryXj (const unsigned int fluidIndex, const unsigned int i, const unsigned int j, const Vector3r & xj)
```

### Protected Attributes

```
Discregrid::DiscreteGrid *m_map
std::vector<std::vector<Real>> m_boundaryVolume
std::vector<std::vector<Vector3r>> m_boundaryXj
Real m_maxDist
Real m_maxVel
```

### Class BoundaryModel\_Koschier2017

- Defined in file `_SPlisHSPlasH_BoundaryModel_Koschier2017.h`

### Inheritance Relationships

#### Base Type

- public SPH::BoundaryModel (*Class BoundaryModel*)

### Class Documentation

```
class SPH::BoundaryModel_Koschier2017 : public SPH::BoundaryModel
```

The boundary model stores the information required for boundary handling using the approach of Koschier and Bender 2017 [15].

### Public Functions

```
BoundaryModel_Koschier2017 ()
~BoundaryModel_Koschier2017 ()
void initModel (RigidBodyObject *rbo)
void reset ()
Discregrid::DiscreteGrid *getMap ()
void setMap (Discregrid::DiscreteGrid *map)
Real getMaxDist () const
void setMaxDist (Real val)
Real getMaxVel () const
void setMaxVel (Real val)
FORCE_INLINE const Real & getBoundaryDensity (const unsigned int fluidIndex, const unsigned int i, const unsigned int j)
FORCE_INLINE Real & getBoundaryDensity (const unsigned int fluidIndex, const unsigned int i, const unsigned int j, const unsigned int k)
```

```
FORCE_INLINE void setBoundaryDensity (const unsigned int fluidIndex, const unsigned int i, const Real val)
FORCE_INLINE Vector3r & getBoundaryDensityGradient (const unsigned int fluidIndex, const unsigned int i, const unsigned int j)
FORCE_INLINE const Vector3r & getBoundaryDensityGradient (const unsigned int fluidIndex, const unsigned int i, const unsigned int j)
FORCE_INLINE void setBoundaryDensityGradient (const unsigned int fluidIndex, const unsigned int i, const unsigned int j, const Real val)
FORCE_INLINE Vector3r & getBoundaryXj (const unsigned int fluidIndex, const unsigned int i, const unsigned int j)
FORCE_INLINE const Vector3r & getBoundaryXj (const unsigned int fluidIndex, const unsigned int i, const unsigned int j)
FORCE_INLINE void setBoundaryXj (const unsigned int fluidIndex, const unsigned int i, const unsigned int j, const Real val)
```

### Protected Attributes

```
Discregrid::DiscreteGrid *m_map
std::vector<std::vector<Real>>> m_boundaryDensity
std::vector<std::vector<Vector3r>>> m_boundaryDensityGradient
std::vector<std::vector<Vector3r>>> m_boundaryXj
Real m_maxDist
Real m_maxVel
```

### Class CohesionKernel

- Defined in file\_SPlisHSPlasH\_SPHKernels.h

### Class Documentation

**class** SPH::CohesionKernel

Cohesion kernel used for the surface tension method of Akinci et al. [2].

### Public Static Functions

```
Real getRadius ()
void setRadius (Real val)
Real W (const Real r)
     $W(r,h) = (32/(\pi h^9))(h-r)^3 r^3$  if  $h/2 < r \leq h$   $(32/(\pi h^9))(2*(h-r)^3 r^3 - h^6/64)$  if  $0 < r \leq h/2$ 
Real W (const Vector3r &r)
Real W_zero ()
```

### Protected Static Attributes

*Real* **m\_radius**

*Real* **m\_k**

*Real* **m\_c**

*Real* **m\_W\_zero**

### Class CubicKernel

- Defined in file\_SPlisHSPlasH\_SPHKernels.h

### Class Documentation

**class** SPH::CubicKernel

Cubic spline kernel.

### Public Static Functions

*Real* **getRadius** ()

void **setRadius** (*Real* val)

*Real* **W** (const *Real* r)

*Real* **W** (const *Vector3r* &r)

*Vector3r* **gradW** (const *Vector3r* &r)

*Real* **W\_zero** ()

### Protected Static Attributes

*Real* **m\_radius**

*Real* **m\_k**

*Real* **m\_l**

*Real* **m\_W\_zero**

### Class CubicKernel2D

- Defined in file\_SPlisHSPlasH\_SPHKernels.h

## Class Documentation

### **class** SPH::CubicKernel2D

Cubic spline kernel (2D).

#### Public Static Functions

*Real* **getRadius** ()

void **setRadius** (*Real* val)

*Real* **W** (const *Real* r)

*Real* **W** (const *Vector3r* &r)

*Vector3r* **gradW** (const *Vector3r* &r)

*Real* **W\_zero** ()

#### Protected Static Attributes

*Real* **m\_radius**

*Real* **m\_k**

*Real* **m\_l**

*Real* **m\_W\_zero**

### Class DragBase

- Defined in file\_SPlisHSPlasH\_Drag\_DragBase.h

## Inheritance Relationships

### Base Type

- public SPH::NonPressureForceBase (*Class NonPressureForceBase*)

### Derived Types

- public SPH::DragForce\_Gissler2017 (*Class DragForce\_Gissler2017*)
- public SPH::DragForce\_Macklin2014 (*Class DragForce\_Macklin2014*)

## Class Documentation

**class** SPH::DragBase : public SPH::NonPressureForceBase

Base class for all drag force methods.

Subclassed by *SPH::DragForce\_Gissler2017*, *SPH::DragForce\_Macklin2014*

### Public Functions

**DragBase** (*FluidModel* \*model)

**~DragBase** (void)

### Public Static Attributes

int DRAG\_COEFFICIENT = -1

### Protected Functions

void **initParameters** ()

### Protected Attributes

*Real* m\_dragCoefficient

## Class DragForce\_Gissler2017

- Defined in file\_SPlisHSPlasH\_Drag\_DragForce\_Gissler2017.h

## Inheritance Relationships

### Base Type

- public SPH::DragBase (*Class DragBase*)

## Class Documentation

**class** SPH::DragForce\_Gissler2017 : public SPH::DragBase

This class implements the drag force computation introduced by Gissler et al. [11].

### Public Functions

```
DragForce_Gissler2017 (FluidModel *model)
~DragForce_Gissler2017 (void)
void step ()
void reset ()
```

### Protected Attributes

```
const Real rho_a = static_cast<Real>(1.2041)
const Real sigma = static_cast<Real>(0.0724)
const Real mu_l = static_cast<Real>(0.00102)
const Real C_F = static_cast<Real>(1.0 / 3.0)
const Real C_k = static_cast<Real>(8.0)
const Real C_d = static_cast<Real>(5.0)
const Real C_b = static_cast<Real>(0.5)
const Real mu_a = static_cast<Real>(0.00001845)
```

### Class DragForce\_Macklin2014

- Defined in file `_SPlisHSPlasH_Drag_DragForce_Macklin2014.h`

### Inheritance Relationships

#### Base Type

- `public SPH::DragBase (Class DragBase)`

### Class Documentation

```
class SPH::DragForce_Macklin2014 : public SPH::DragBase
```

This class implements the drag force computation introduced by Macklin et al. [18].

### Public Functions

```
DragForce_Macklin2014 (FluidModel *model)
~DragForce_Macklin2014 (void)
void step ()
void reset ()
```



## Class Elasticity\_Becker2009

- Defined in file\_SPlisHSPlasH\_Elasticity\_Elasticity\_Becker2009.h

## Inheritance Relationships

### Base Type

- public SPH::ElasticityBase (*Class ElasticityBase*)

## Class Documentation

**class** SPH::Elasticity\_Becker2009 : public SPH::ElasticityBase

This class implements the corotated SPH method for deformable solids introduced by Becker et al. [4].

### Public Functions

**Elasticity\_Becker2009** (*FluidModel \*model*)

**~Elasticity\_Becker2009** (void)

void **step** ()

void **reset** ()

void **performNeighborhoodSearchSort** ()

void **saveState** (BinaryFileWriter &*binWriter*)

void **loadState** (BinaryFileReader &*binReader*)

### Public Static Attributes

int **ALPHA** = -1

### Protected Functions

void **initValues** ()

void **computeRotations** ()

void **computeStress** ()

void **computeForces** ()

void **initParameters** ()

**FORCE\_INLINE** void **symMatTimesVec** (const Vector6r &*M*, const Vector3r &*v*, Vector3r &*res*)

### Protected Attributes

```
std::vector<unsigned int> m_current_to_initial_index
std::vector<unsigned int> m_initial_to_current_index
std::vector<std::vector<unsigned int>> m_initialNeighbors
std::vector<Real> m_restVolumes
std::vector<Matrix3r> m_rotations
std::vector<Vector6r> m_stress
std::vector<Matrix3r> m_F
Real m_alpha
```

### Class Elasticity\_Peer2018

- Defined in file\_SPlisHSPlasH\_Elasticity\_Elasticity\_Peer2018.h

### Inheritance Relationships

#### Base Type

- public SPH::ElasticityBase (*Class ElasticityBase*)

### Class Documentation

**class** SPH::Elasticity\_Peer2018 : public SPH::ElasticityBase

This class implements the implicit SPH formulation for incompressible linearly elastic solids introduced by Peer et al. [21].

### Public Functions

**Elasticity\_Peer2018** (*FluidModel \*model*)

**~Elasticity\_Peer2018** (void)

void **step** ()

void **reset** ()

void **performNeighborhoodSearchSort** ()

void **saveState** (BinaryFileWriter &*binWriter*)

void **loadState** (BinaryFileReader &*binReader*)

## Public Static Functions

void **matrixVecProd** (const *Real* \*vec, *Real* \*result, void \*userData)

## Public Static Attributes

int **ITERATIONS** = -1

int **MAX\_ITERATIONS** = -1

int **MAX\_ERROR** = -1

int **ALPHA** = -1

## Protected Types

**typedef** Eigen::ConjugateGradient<*MatrixReplacement*, Eigen::Lower | Eigen::Upper, Eigen::IdentityPreconditioner> **Solve**

## Protected Functions

void **initValues** ()

void **computeMatrixL** ()

void **computeRotations** ()

void **computeRHS** (VectorXr &rhs)

void **initParameters** ()

**FORCE\_INLINE** void **symMatTimesVec** (const Vector6r &M, const Vector3r &v, Vector3r &res)

## Protected Attributes

std::vector<unsigned int> **m\_current\_to\_initial\_index**

std::vector<unsigned int> **m\_initial\_to\_current\_index**

std::vector<std::vector<unsigned int>> **m\_initialNeighbors**

std::vector<*Real*> **m\_restVolumes**

std::vector<*Matrix3r*> **m\_rotations**

std::vector<*Vector6r*> **m\_stress**

std::vector<*Matrix3r*> **m\_L**

std::vector<*Matrix3r*> **m\_RL**

std::vector<*Matrix3r*> **m\_F**

unsigned int **m\_iterations**

unsigned int **m\_maxIter**

*Real* **m\_maxError**

*Real* **m\_alpha**

*Solver* **m\_solver**

## Class ElasticityBase

- Defined in file\_SPlisHSPlasH\_Elasticity\_ElasticityBase.h

## Inheritance Relationships

### Base Type

- `public SPH::NonPressureForceBase` (*Class NonPressureForceBase*)

### Derived Types

- `public SPH::Elasticity_Becker2009` (*Class Elasticity\_Becker2009*)
- `public SPH::Elasticity_Peer2018` (*Class Elasticity\_Peer2018*)

## Class Documentation

**class** `SPH::ElasticityBase` : **public** `SPH::NonPressureForceBase`  
 Base class for all elasticity methods.

Subclassed by *SPH::Elasticity\_Becker2009*, *SPH::Elasticity\_Peer2018*

### Public Functions

**ElasticityBase** (*FluidModel \*model*)

**~ElasticityBase** (void)

### Public Static Attributes

int **YOUNGS\_MODULUS** = -1

int **POISSON\_RATIO** = -1

### Protected Functions

void **initParameters** ()

### Protected Attributes

*Real* **m\_youngsModulus**

*Real* **m\_poissonRatio**

## Class Emitter

- Defined in file\_SPlisHSPlasH\_Emitter.h

## Class Documentation

**class** SPH::Emitter

### Public Functions

```

Emitter (FluidModel *model, const unsigned int width, const unsigned int height, const Vector3r
        &pos, const Matrix3r &rotation, const Real velocity, const unsigned int type = 0)
~Emitter ()
void emitParticles (std::vector<unsigned int> &reusedParticles, unsigned int &indexReuse, un-
    signed int &numEmittedParticles)
void emitParticlesCircle (std::vector<unsigned int> &reusedParticles, unsigned int &in-
    dexReuse, unsigned int &numEmittedParticles)
Real getNextEmitTime () const
void setNextEmitTime (Real val)
void setEmitStartTime (Real val)
void setEmitEndTime (Real val)
void step (std::vector<unsigned int> &reusedParticles, unsigned int &indexReuse, unsigned int &nu-
    mEmittedParticles)
void reset ()
void saveState (BinaryFileWriter &binWriter)
void loadState (BinaryFileReader &binReader)

```

### Public Static Functions

```

Vector3r getSize (const Real width, const Real height, const int type)

```

### Protected Functions

```

FORCE_INLINE bool inBox (const Vector3r &x, const Vector3r &xBox, const Matrix3r &rotB
FORCE_INLINE bool inCylinder (const Vector3r &x, const Vector3r &xCyl, const Matrix3r

```

### Protected Attributes

*FluidModel* \***m\_model**  
unsigned int **m\_width**  
unsigned int **m\_height**  
*Vector3r* **m\_x**  
*Matrix3r* **m\_rotation**  
*Real* **m\_velocity**  
unsigned int **m\_type**  
*Real* **m\_nextEmitTime**  
*Real* **m\_emitStartTime**  
*Real* **m\_emitEndTime**  
unsigned int **m\_emitCounter**

### Class EmitterSystem

- Defined in file `_SPlisHSPlasH_EmitterSystem.h`

### Class Documentation

**class** SPH::**EmitterSystem**

#### Public Functions

**EmitterSystem** (*FluidModel* \**model*)  
**~EmitterSystem** ()  
void **enableReuseParticles** (const *Vector3r* &*boxMin* = *Vector3r*(-1, -1, -1), const *Vector3r* &*boxMax* = *Vector3r*(1, 1, 1))  
void **disableReuseParticles** ()  
void **addEmitter** (const unsigned int *width*, const unsigned int *height*, const *Vector3r* &*pos*, const *Matrix3r* &*rotation*, const *Real* *velocity*, const unsigned int *type*)  
unsigned int **numEmitters** () const  
std::vector<*Emitter*\*> &**getEmitters** ()  
unsigned int **numReusedParticles** () const  
unsigned int **numEmittedParticles** () const  
void **step** ()  
void **reset** ()  
void **saveState** (BinaryFileWriter &*binWriter*)  
void **loadState** (BinaryFileReader &*binReader*)

### Protected Functions

void **reuseParticles** ()

### Protected Attributes

*FluidModel* \***m\_model**

bool **m\_reuseParticles**

*Vector3r* **m\_boxMin**

*Vector3r* **m\_boxMax**

unsigned int **m\_numberOfEmittedParticles**

unsigned int **m\_numReusedParticles**

std::vector<unsigned int> **m\_reusedParticles**

std::vector<*Emitter*\*> **m\_emitters**

### Protected Static Attributes

const unsigned int **m\_maxParticlesToReusePerStep** = 50000

## Class FluidModel

- Defined in file\_SPlisHSPlasH\_FluidModel.h

## Inheritance Relationships

### Base Type

- public ParameterObject

## Class Documentation

**class** SPH::FluidModel : public ParameterObject  
 The fluid model stores the particle and simulation information.

### Public Functions

**FluidModel** ()

**FluidModel** (const *FluidModel*&) = delete

*FluidModel* &operator= (const *FluidModel*&) = delete

**~FluidModel** ()

void **init** ()

std::string **getId** () const

```
FORCE_INLINE Real getDensity0 () const
void setDensity0 (const Real v)
unsigned int getPointSetIndex () const
void addField (const FieldDescription &field)
const std::vector<FieldDescription> &getFields ()
const FieldDescription &getField (const unsigned int i)
const FieldDescription &getField (const std::string &name)
const unsigned int numberOfFields ()
void removeFieldByName (const std::string &fieldName)
void setNumActiveParticles (const unsigned int num)
unsigned int numberOfParticles () const
EmitterSystem *getEmitterSystem ()
void reset ()
void performNeighborhoodSearchSort ()
void initModel (const std::string &id, const unsigned int nFluidParticles, Vector3r *fluidParticles,
               Vector3r *fluidVelocities, const unsigned int nMaxEmitterParticles)
const unsigned int numParticles () const
unsigned int numActiveParticles () const
unsigned int getNumActiveParticles0 () const
void setNumActiveParticles0 (unsigned int val)
void emittedParticles (const unsigned int startIndex)
int getSurfaceTensionMethod () const
void setSurfaceTensionMethod (const int val)
int getViscosityMethod () const
void setViscosityMethod (const int val)
int getVorticityMethod () const
void setVorticityMethod (const int val)
int getDragMethod () const
void setDragMethod (const int val)
int getElasticityMethod () const
void setElasticityMethod (const int val)
SurfaceTensionBase *getSurfaceTensionBase ()
ViscosityBase *getViscosityBase ()
VorticityBase *getVorticityBase ()
DragBase *getDragBase ()
ElasticityBase *getElasticityBase ()
```



```

void setDragMethodChangedCallback (std::function<void>
    > const &callBackFct)

void setSurfaceMethodChangedCallback (std::function<void>
    > const &callBackFct)

void setViscosityMethodChangedCallback (std::function<void>
    > const &callBackFct)

void setVorticityMethodChangedCallback (std::function<void>
    > const &callBackFct)

void setElasticityMethodChangedCallback (std::function<void>
    > const &callBackFct)

void computeSurfaceTension ()

void computeViscosity ()

void computeVorticity ()

void computeDragForce ()

void computeElasticity ()

void saveState (BinaryFileWriter &binWriter)

void loadState (BinaryFileReader &binReader)

FORCE_INLINE Vector3r & getPosition0 (const unsigned int i)

FORCE_INLINE const Vector3r & getPosition0 (const unsigned int i) const

FORCE_INLINE void setPosition0 (const unsigned int i, const Vector3r &pos)

FORCE_INLINE Vector3r & getPosition (const unsigned int i)

FORCE_INLINE const Vector3r & getPosition (const unsigned int i) const

FORCE_INLINE void setPosition (const unsigned int i, const Vector3r &pos)

FORCE_INLINE Vector3r & getVelocity (const unsigned int i)

FORCE_INLINE const Vector3r & getVelocity (const unsigned int i) const

FORCE_INLINE void setVelocity (const unsigned int i, const Vector3r &vel)

FORCE_INLINE Vector3r & getVelocity0 (const unsigned int i)

FORCE_INLINE const Vector3r & getVelocity0 (const unsigned int i) const

FORCE_INLINE void setVelocity0 (const unsigned int i, const Vector3r &vel)

FORCE_INLINE Vector3r & getAcceleration (const unsigned int i)

FORCE_INLINE const Vector3r & getAcceleration (const unsigned int i) const

FORCE_INLINE void setAcceleration (const unsigned int i, const Vector3r &accel)

FORCE_INLINE const Real getMass (const unsigned int i) const

FORCE_INLINE Real & getMass (const unsigned int i)

FORCE_INLINE void setMass (const unsigned int i, const Real mass)

FORCE_INLINE const Real & getDensity (const unsigned int i) const

FORCE_INLINE Real & getDensity (const unsigned int i)

FORCE_INLINE void setDensity (const unsigned int i, const Real &val)

```

```
FORCE_INLINE unsigned int & getParticleId (const unsigned int i)
FORCE_INLINE const unsigned int & getParticleId (const unsigned int i) const
FORCE_INLINE const ParticleState & getParticleState (const unsigned int i) const
FORCE_INLINE ParticleState & getParticleState (const unsigned int i)
FORCE_INLINE void setParticleState (const unsigned int i, const ParticleState &val)
FORCE_INLINE const Real getVolume (const unsigned int i) const
FORCE_INLINE Real & getVolume (const unsigned int i)
```

### Public Static Attributes

```
int NUM_PARTICLES = -1
int NUM_REUSED_PARTICLES = -1
int DENSITY0 = -1
int DRAG_METHOD = -1
int SURFACE_TENSION_METHOD = -1
int VISCOSITY_METHOD = -1
int VORTICITY_METHOD = -1
int ELASTICITY_METHOD = -1
int ENUM_DRAG_NONE = -1
int ENUM_DRAG_MACKLIN2014 = -1
int ENUM_DRAG_GISSLER2017 = -1
int ENUM_SURFACETENSION_NONE = -1
int ENUM_SURFACETENSION_BECKER2007 = -1
int ENUM_SURFACETENSION_AKINCI2013 = -1
int ENUM_SURFACETENSION_HE2014 = -1
int ENUM_VISCOSITY_NONE = -1
int ENUM_VISCOSITY_STANDARD = -1
int ENUM_VISCOSITY_XSPH = -1
int ENUM_VISCOSITY_BENDER2017 = -1
int ENUM_VISCOSITY_PEER2015 = -1
int ENUM_VISCOSITY_PEER2016 = -1
int ENUM_VISCOSITY_TAKAHASHI2015 = -1
int ENUM_VISCOSITY_WEILER2018 = -1
int ENUM_VORTICITY_NONE = -1
int ENUM_VORTICITY_MICROPOLAR = -1
int ENUM_VORTICITY_VC = -1
int ENUM_ELASTICITY_NONE = -1
```

```
int ENUM_ELASTICITY_BECKER2009 = -1
```

```
int ENUM_ELASTICITY_PEER2018 = -1
```

### Protected Functions

```
void initParameters ()
```

```
void initMasses ()
```

```
void resizeFluidParticles (const unsigned int newSize)
    Resize the arrays containing the particle data.
```

```
void releaseFluidParticles ()
    Release the arrays containing the particle data.
```

### Protected Attributes

```
std::string m_id
```

```
EmitterSystem *m_emitterSystem
```

```
std::vector<Real> m_masses
```

```
std::vector<Vector3r> m_a
```

```
std::vector<Vector3r> m_v0
```

```
std::vector<Vector3r> m_x0
```

```
std::vector<Vector3r> m_x
```

```
std::vector<Vector3r> m_v
```

```
std::vector<Real> m_density
```

```
std::vector<unsigned int> m_particleId
```

```
std::vector<ParticleState> m_particleState
```

```
Real m_V
```

```
SurfaceTensionMethods m_surfaceTensionMethod
```

```
SurfaceTensionBase *m_surfaceTension
```

```
ViscosityMethods m_viscosityMethod
```

```
ViscosityBase *m_viscosity
```

```
VorticityMethods m_vorticityMethod
```

```
VorticityBase *m_vorticity
```

```
DragMethods m_dragMethod
```

```
DragBase *m_drag
```

```
ElasticityMethods m_elasticityMethod
```

```
ElasticityBase *m_elasticity
```

```
std::vector<FieldDescription> m_fields
```

```
std::function<void ()> m_dragMethodChanged
```

```

std::function<void ()> m_surfaceTensionMethodChanged
std::function<void ()> m_viscosityMethodChanged
std::function<void ()> m_vorticityMethodChanged
std::function<void ()> m_elasticityMethodChanged
Real m_density0
unsigned int m_pointSetIndex
unsigned int m_numActiveParticles
unsigned int m_numActiveParticles0

```

## Class GaussQuadrature

- Defined in file `_SPlisHSPlasH_Uutilities_GaussQuadrature.h`

## Class Documentation

```
class SPH::GaussQuadrature
```

### Public Types

```

using Integrand = std::function<double (Eigen::Vector3d const&)>
using Domain = Eigen::AlignedBox3d

```

### Public Static Functions

```

double integrate (Integrand integrand, Domain const &domain, unsigned int p)
void exportSamples (unsigned int p)

```

## Class JacobiPreconditioner1D

- Defined in file `_SPlisHSPlasH_Uutilities_MatrixFreeSolver.h`

## Class Documentation

```
class SPH::JacobiPreconditioner1D
    Matrix-free Jacobi preconditioner
```

## Public Types

```
enum [anonymous]
    Values:

    enumerator ColsAtCompileTime = Eigen::Dynamic
    enumerator MaxColsAtCompileTime = Eigen::Dynamic

typedef SystemMatrixType::StorageIndex StorageIndex
typedef void (*DiagonalMatrixElementFct) (const unsigned int, Real&, void*)
```

## Public Functions

```
JacobiPreconditioner1D()
void init (const unsigned int dim, DiagonalMatrixElementFct fct, void *userData)
Eigen::Index rows () const
Eigen::Index cols () const
Eigen::ComputationInfo info ()
template<typename MatType>
JacobiPreconditioner1D &analyzePattern (const MatType&)
template<typename MatType>
JacobiPreconditioner1D &factorize (const MatType &mat)
template<typename MatType>
JacobiPreconditioner1D &compute (const MatType &mat)
template<typename Rhs, typename Dest>
void _solve_impl (const Rhs &b, Dest &x) const
template<typename Rhs>
const Eigen::Solve<JacobiPreconditioner1D, Rhs> solve (const Eigen::MatrixBase<Rhs> &b)
const
```

## Protected Attributes

```
unsigned int m_dim
DiagonalMatrixElementFct m_diagonalElementFct
    diagonal matrix element callback
void *m_userData
VectorXr m_invDiag
```

## Class JacobiPreconditioner3D

- Defined in file\_SPlisHSPlasH\_Uilities\_MatrixFreeSolver.h

## Class Documentation

**class** SPH::JacobiPreconditioner3D

Matrix-free Jacobi preconditioner

### Public Types

**enum** [anonymous]

*Values:*

**enumerator** ColsAtCompileTime = Eigen::Dynamic

**enumerator** MaxColsAtCompileTime = Eigen::Dynamic

**typedef** *SystemMatrixType*::StorageIndex **StorageIndex**

**typedef** void (\**DiagonalMatrixElementFct*) (const unsigned int, *Vector3r*&, void\*)

### Public Functions

**JacobiPreconditioner3D** ()

void **init** (const unsigned int *dim*, *DiagonalMatrixElementFct* *fct*, void \**userData*)

Eigen::Index **rows** () const

Eigen::Index **cols** () const

Eigen::ComputationInfo **info** ()

template<typename **MatType**>

*JacobiPreconditioner3D* &**analyzePattern** (const *MatType*&)

template<typename **MatType**>

*JacobiPreconditioner3D* &**factorize** (const *MatType* &*mat*)

template<typename **MatType**>

*JacobiPreconditioner3D* &**compute** (const *MatType* &*mat*)

template<typename **Rhs**, typename **Dest**>

void **\_solve\_impl** (const *Rhs* &*b*, *Dest* &*x*) const

template<typename **Rhs**>

const Eigen::Solve<*JacobiPreconditioner3D*, *Rhs*> **solve** (const Eigen::MatrixBase<*Rhs*> &*b*)  
const

## Protected Attributes

unsigned int **m\_dim**  
*DiagonalMatrixElementFct* **m\_diagonalElementFct**  
 diagonal matrix element callback  
 void \***m\_userData**  
 VectorXr **m\_invDiag**

## Class MathFunctions

- Defined in file\_SPlisHSPlasH\_Uilities\_MathFunctions.h

## Class Documentation

**class** SPH::MathFunctions

### Public Static Functions

void **extractRotation** (const *Matrix3r* &A, *Quaternionr* &q, const unsigned int *maxIter*)  
 Implementation of the paper: Matthias Müller, Jan Bender, Nuttapong Chentanez and Miles Macklin, “A Robust Method to Extract the Rotational Part of Deformations”, ACM SIGGRAPH Motion in Games, 2016

void **pseudoInverse** (const *Matrix3r* &a, *Matrix3r* &res)

void **svdWithInversionHandling** (const *Matrix3r* &A, *Vector3r* &sigma, *Matrix3r* &U, *Matrix3r* &VT)  
 Perform a singular value decomposition of matrix A:  $A = U * \sigma * V^T$ . This function returns two proper rotation matrices U and  $V^T$  which do not contain a reflection. Reflections are corrected by the inversion handling proposed by Irving et al. 2004.

void **eigenDecomposition** (const *Matrix3r* &A, *Matrix3r* &eigenVecs, *Vector3r* &eigenVals)

void **jacobiRotate** (*Matrix3r* &A, *Matrix3r* &R, int *p*, int *q*)

void **getOrthogonalVectors** (const *Vector3r* &vec, *Vector3r* &x, *Vector3r* &y)  
 Returns two orthogonal vectors to vec which are also orthogonal to each other.

## Class MatrixReplacement

- Defined in file\_SPlisHSPlasH\_Uilities\_MatrixFreeSolver.h

## Inheritance Relationships

### Base Type

- `public Eigen::EigenBase< MatrixReplacement >`

### Class Documentation

**class** SPH::MatrixReplacement : public Eigen::EigenBase<MatrixReplacement>  
Replacement of the matrix in the linear system which is required for a matrix-free solver.

#### Public Types

```
enum [anonymous]
    Values:

    enumerator ColsAtCompileTime = Eigen::Dynamic
    enumerator MaxColsAtCompileTime = Eigen::Dynamic
    enumerator IsRowMajor = false

typedef Real Scalar
typedef Real RealScalar
typedef int StorageIndex
typedef void (*MatrixVecProdFct)(const Real*, Real*, void*)
```

#### Public Functions

```
Index rows () const
Index cols () const

template<typename Rhs>
Eigen::Product<MatrixReplacement, Rhs, Eigen::AliasFreeProduct> operator* (const
                                                                    Eigen::MatrixBase<Rhs>
                                                                    &x) const

MatrixReplacement (const unsigned int dim, MatrixVecProdFct fct, void *userData)

void *getUserData ()

MatrixVecProdFct getMatrixVecProdFct ()
```



### Protected Attributes

```
unsigned int m_dim
void *m_userData
MatrixVecProdFct m_matrixVecProdFct
    matrix vector product callback
```

### Class MicropolarModel\_Bender2017

- Defined in file\_SPlisHSPlasH\_Vorticity\_MicropolarModel\_Bender2017.h

### Inheritance Relationships

#### Base Type

- public SPH::VorticityBase (*Class VorticityBase*)

### Class Documentation

```
class SPH::MicropolarModel_Bender2017 : public SPH::VorticityBase
    This class implements the micropolar material model introduced by Bender et al. [9].
```

### Public Functions

```
MicropolarModel_Bender2017 (FluidModel *model)
~MicropolarModel_Bender2017 (void)
void step ()
void reset ()
void performNeighborhoodSearchSort ()
FORCE_INLINE const Vector3r & getAngularAcceleration (const unsigned int i) const
FORCE_INLINE Vector3r & getAngularAcceleration (const unsigned int i)
FORCE_INLINE void setAngularAcceleration (const unsigned int i, const Vector3r &val)
FORCE_INLINE const Vector3r & getAngularVelocity (const unsigned int i) const
FORCE_INLINE Vector3r & getAngularVelocity (const unsigned int i)
FORCE_INLINE void setAngularVelocity (const unsigned int i, const Vector3r &val)
```

### Public Static Attributes

```
int VISCOSITY_OMEGA = -1  
int INERTIA_INVERSE = -1
```

### Protected Functions

```
void initParameters ()
```

### Protected Attributes

```
std::vector<Vector3r> m_angularAcceleration  
std::vector<Vector3r> m_omega  
Real m_viscosityOmega  
Real m_inertiaInverse
```

## Class NonPressureForceBase

- Defined in file\_SPlisHSPlasH\_NonPressureForceBase.h

## Inheritance Relationships

### Base Type

- public ParameterObject

### Derived Types

- public SPH::DragBase (*Class DragBase*)
- public SPH::ElasticityBase (*Class ElasticityBase*)
- public SPH::SurfaceTensionBase (*Class SurfaceTensionBase*)
- public SPH::ViscosityBase (*Class ViscosityBase*)
- public SPH::VorticityBase (*Class VorticityBase*)

## Class Documentation

```
class SPH::NonPressureForceBase : public ParameterObject
```

Base class for all non-pressure force methods.

Subclassed by *SPH::DragBase*, *SPH::ElasticityBase*, *SPH::SurfaceTensionBase*, *SPH::ViscosityBase*, *SPH::VorticityBase*

## Public Functions

```

NonPressureForceBase (FluidModel *model)
NonPressureForceBase (const NonPressureForceBase&) = delete
NonPressureForceBase &operator= (const NonPressureForceBase&) = delete
~NonPressureForceBase (void)
void step () = 0
void reset ()
void performNeighborhoodSearchSort ()
void emittedParticles (const unsigned int startIndex)
void saveState (BinaryFileWriter &binWriter)
void loadState (BinaryFileReader &binReader)
FluidModel *getModel ()
void init ()

```

## Protected Attributes

```

FluidModel *m_model

```

## Class PoissonDiskSampling

- Defined in file\_SPlisHSPlasH\_Uilities\_PoissonDiskSampling.h

## Nested Relationships

### Nested Types

- *Struct PoissonDiskSampling::CellPosHasher*
- *Struct PoissonDiskSampling::HashEntry*
- *Struct PoissonDiskSampling::InitialPointInfo*

## Class Documentation

```

class SPH::PoissonDiskSampling

```

This class implements a Poisson disk sampling for the surface of 3D models.

## Public Functions

### PoissonDiskSampling()

```
void sampleMesh (const unsigned int numVertices, const Vector3r *vertices, const unsigned int  
                numFaces, const unsigned int *faces, const Real minRadius, const unsigned  
                int numTrials, unsigned int distanceNorm, std::vector<Vector3r> &samples)
```

Performs the poisson sampling with the respective parameters. Compare [http://graphics.cs.umass.edu/pubs/sa\\_2010.pdf](http://graphics.cs.umass.edu/pubs/sa_2010.pdf)

### Parameters

- *mesh*: mesh data of sampled body
- *vertices*: vertex data of sampled data
- *sampledVertices*: sampled vertices that will be returned
- *minRadius*: minimal distance of sampled vertices
- *numTestpointsPerFace*: # of generated test points per face of body
- *distanceNorm*: 0: euclidean norm, 1: approx geodesic distance
- *numTrials*: # of iterations used to find samples

## Public Static Functions

```
FORCE_INLINE int floor (const Real v)
```

```
struct HashEntry
```

Struct to store the hash entry (spatial hashing)

## Public Functions

```
HashEntry ()
```

## Public Members

```
std::vector<unsigned int> samples
```

```
unsigned int startIndex
```

```
struct InitialPointInfo
```

Struct to store the information of the initial points.

## Public Members

```
CellPos cP
```

```
Vector3r pos
```

```
unsigned int ID
```

## Class Poly6Kernel

- Defined in file\_SPlisHSPlasH\_SPHKernels.h

## Class Documentation

**class** SPH::Poly6Kernel

Poly6 kernel.

### Public Static Functions

*Real* **getRadius** ()

void **setRadius** (*Real* val)

*Real* **W**(const *Real* r)

$W(r,h) = (315/(64 \pi h^9))(h^2-|r|^2)^3 = (315/(64 \pi h^9))(h^2-r*r)^3$

*Real* **W**(const *Vector3r* &r)

*Vector3r* **gradW**(const *Vector3r* &r)

$\text{grad}(W(r,h)) = r(-945/(32 \pi h^9))(h^2-|r|^2)^2 = r(-945/(32 \pi h^9))(h^2-r*r)^2$

*Real* **laplacianW**(const *Vector3r* &r)

$\text{laplacian}(W(r,h)) = (-945/(32 \pi h^9))(h^2-|r|^2)(-7|r|^2+3h^2) = (-945/(32 \pi h^9))(h^2-r*r)(3 h^2-7 r*r)$

*Real* **W\_zero** ()

### Protected Static Attributes

*Real* **m\_radius**

*Real* **m\_k**

*Real* **m\_l**

*Real* **m\_m**

*Real* **m\_W\_zero**

## Template Class PrecomputedKernel

- Defined in file\_SPlisHSPlasH\_SPHKernels.h

## Class Documentation

template<typename **KernelType**, unsigned int **resolution** = 10000u>

**class** SPH::PrecomputedKernel

Precomputed kernel which is based on a lookup table as described by Bender and Koschier [5], [6].

The lookup tables can be used in combination with any kernel.

### Public Static Functions

```
Real getRadius ()  
void setRadius (Real val)  
Real W (const Vector3r &r)  
Real W (const Real r)  
Vector3r gradW (const Vector3r &r)  
Real W_zero ()
```

### Protected Static Attributes

```
Real m_W[resolution]  
Real m_gradW[resolution + 1]  
Real m_radius  
Real m_radius2  
Real m_invStepSize  
Real m_W_zero
```

### Class RegularSampling2D

- Defined in file `_SPlisHSPlasH_Uutilities_RegularSampling2D.h`

### Class Documentation

**class** `SPH::RegularSampling2D`

This class implements a per-triangle regular sampling for the surface of 3D models.

### Public Functions

```
RegularSampling2D ()
```

### Public Static Functions

```
void sampleMesh (const Matrix3r &rotation, const Vector3r &translation, const unsigned  
                numVertices, const Vector3r *vertices, const unsigned int numFaces, const  
                unsigned int *faces, const Real maxDistance, std::vector<Vector3r> &samples)  
Performs the poisson sampling with the respective parameters. Compare http://graphics.cs.umass.edu/pubs/sa\_2010.pdf
```

### Parameters

- `rotation`: rotation of the mesh
- `translation`: translation of the mesh
- `numVertices`: number of mesh vertices

- `vertices`: vertex data of sampled data
- `numFaces`: number of faces in the mesh
- `faces`: face data of sampled mesh
- `maxDistance`: maximal distance of sampled vertices
- `samples`: vector to store the samples

### Class RegularTriangleSampling

- Defined in file `_SPlisHSPlasH_Uutilities_RegularTriangleSampling.h`

### Class Documentation

#### **class** `SPH::RegularTriangleSampling`

This class implements a per-triangle regular sampling for the surface of 3D models.

#### Public Functions

**RegularTriangleSampling** ()

#### Public Static Functions

void **sampleMesh** (**const** unsigned int *numVertices*, **const** *Vector3r* \**vertices*, **const** unsigned int *numFaces*, **const** unsigned int \**faces*, **const** *Real* *maxDistance*, **std::vector**<*Vector3r*> &*samples*)

Performs the poisson sampling with the respective parameters. Compare [http://graphics.cs.umass.edu/pubs/sa\\_2010.pdf](http://graphics.cs.umass.edu/pubs/sa_2010.pdf)

#### Parameters

- `numVertices`: number of mesh vertices
- `vertices`: vertex data of sampled data
- `numFaces`: number of faces in the mesh
- `faces`: face data of sampled mesh
- `maxDistance`: maximal distance of sampled vertices
- `samples`: vector to store the samples

### Class RigidBodyObject

- Defined in file `_SPlisHSPlasH_RigidBodyObject.h`

## Inheritance Relationships

### Derived Type

- `public SPH::StaticRigidBody (Class StaticRigidBody)`

### Class Documentation

#### **class** `SPH::RigidBodyObject`

Base class for rigid body objects.

Subclassed by *SPH::StaticRigidBody*

#### Public Functions

`~RigidBodyObject ()`

`bool isDynamic () const = 0`

*Real* `const getMass () const = 0`

*Vector3r* `const &getPosition () const = 0`

`void setPosition (const Vector3r &x) = 0`

*Vector3r* `getWorldSpacePosition () const = 0`

*Vector3r* `const &getVelocity () const = 0`

`void setVelocity (const Vector3r &v) = 0`

*Matrix3r* `const &getRotation () const = 0`

`void setRotation (const Matrix3r &r) = 0`

*Matrix3r* `getWorldSpaceRotation () const = 0`

*Vector3r* `const &getAngularVelocity () const = 0`

`void setAngularVelocity (const Vector3r &v) = 0`

`void addForce (const Vector3r &f) = 0`

`void addTorque (const Vector3r &t) = 0`

`const std::vector<Vector3r> &getVertices () const = 0`

`const std::vector<Vector3r> &getVertexNormals () const = 0`

`const std::vector<unsigned int> &getFaces () const = 0`



## Class SimpleQuadrature

- Defined in file\_SPlisHSPlasH\_Uilities\_SimpleQuadrature.h

## Class Documentation

```
class SPH::SimpleQuadrature
```

### Public Types

```
using Integrand = std::function<double (Eigen::Vector3d const&)>
```

```
using Domain = Eigen::AlignedBox3d
```

### Public Static Functions

```
void determineSamplePointsInSphere (const double radius, unsigned int p)
```

```
void determineSamplePointsInCircle (const double radius, unsigned int p)
```

```
double integrate (Integrand integrand)
```

### Public Static Attributes

```
std::vector<Eigen::Vector3d> m_samplePoints
```

```
double m_volume = 0.0
```

## Class Simulation

- Defined in file\_SPlisHSPlasH\_Simulation.h

## Inheritance Relationships

### Base Type

- public ParameterObject

## Class Documentation

```
class SPH::Simulation : public ParameterObject
```

Class to manage the current simulation time and the time step size. This class is a singleton.

## Public Types

```
typedef PrecomputedKernel<CubicKernel, 10000> PrecomputedCubicKernel
```

## Public Functions

```

Simulation ()
Simulation (const Simulation&) = delete
Simulation &operator= (const Simulation&) = delete
~Simulation ()
void init (const Real particleRadius, const bool sim2D)
void reset ()
void addFluidModel (const std::string &id, const unsigned int nFluidParticles, Vector3r *fluid-
    Particles, Vector3r *fluidVelocities, const unsigned int nMaxEmitterParticles)
FluidModel *getFluidModel (const unsigned int index)
FluidModel *getFluidModelFromPointSet (const unsigned int pointSetIndex)
const unsigned int numberOfFluidModels () const
void addBoundaryModel (BoundaryModel *bm)
BoundaryModel *getBoundaryModel (const unsigned int index)
BoundaryModel *getBoundaryModelFromPointSet (const unsigned int pointSetIndex)
const unsigned int numberOfBoundaryModels () const
void updateBoundaryVolume ()
AnimationFieldSystem *getAnimationFieldSystem ()
BoundaryHandlingMethods getBoundaryHandlingMethod () const
void setBoundaryHandlingMethod (BoundaryHandlingMethods val)
int getKernel () const
void setKernel (int val)
int getGradKernel () const
void setGradKernel (int val)
FORCE_INLINE Real W_zero () const
FORCE_INLINE Real W (const Vector3r &r) const
FORCE_INLINE Vector3r gradW (const Vector3r &r)
int getSimulationMethod () const
void setSimulationMethod (const int val)
void setSimulationMethodChangedCallback (std::function<void>
    > const &callBackFct)
TimeStep *getTimeStep ()
bool is2DSimulation ()

```

```

bool zSortEnabled ()
void setParticleRadius (Real val)
Real getParticleRadius () const
Real getSupportRadius () const
void updateTimeStepSize ()
    Update time step size depending on the chosen method.
void updateTimeStepSizeCFL ()
    Update time step size by CFL condition.
void performNeighborhoodSearch ()
    Perform the neighborhood search for all fluid particles.
void performNeighborhoodSearchSort ()
void computeNonPressureForces ()
void animateParticles ()
void emitParticles ()
void emittedParticles (FluidModel *model, const unsigned int startIndex)
NeighborhoodSearch *getNeighborhoodSearch ()
void saveState (BinaryFileWriter &binWriter)
void loadState (BinaryFileReader &binReader)

FORCE_INLINE unsigned int numberOfPointSets () const
FORCE_INLINE unsigned int numberOfNeighbors (const unsigned int pointSetIndex, const unsigned int neighborIndex)
FORCE_INLINE unsigned int getNeighbor (const unsigned int pointSetIndex, const unsigned int neighborIndex)
FORCE_INLINE const unsigned int * getNeighborList (const unsigned int pointSetIndex, const unsigned int neighborIndex)

```

### Public Static Functions

```

Simulation *getCurrent ()
void setCurrent (Simulation *tm)
bool hasCurrent ()

```

### Public Static Attributes

```

int SIM_2D = -1
int PARTICLE_RADIUS = -1
int GRAVITATION = -1
int CFL_METHOD = -1
int CFL_FACTOR = -1
int CFL_MIN_TIMESTEPSIZE = -1
int CFL_MAX_TIMESTEPSIZE = -1
int ENABLE_Z_SORT = -1

```

```
int KERNEL_METHOD = -1
int GRAD_KERNEL_METHOD = -1
int ENUM_KERNEL_CUBIC = -1
int ENUM_KERNEL_WENDLANDQUINTICC2 = -1
int ENUM_KERNEL_POLY6 = -1
int ENUM_KERNEL_SPIKY = -1
int ENUM_KERNEL_PRECOMPUTED_CUBIC = -1
int ENUM_KERNEL_CUBIC_2D = -1
int ENUM_KERNEL_WENDLANDQUINTICC2_2D = -1
int ENUM_GRADKERNEL_CUBIC = -1
int ENUM_GRADKERNEL_WENDLANDQUINTICC2 = -1
int ENUM_GRADKERNEL_POLY6 = -1
int ENUM_GRADKERNEL_SPIKY = -1
int ENUM_GRADKERNEL_PRECOMPUTED_CUBIC = -1
int ENUM_GRADKERNEL_CUBIC_2D = -1
int ENUM_GRADKERNEL_WENDLANDQUINTICC2_2D = -1
int SIMULATION_METHOD = -1
int ENUM_CFL_NONE = -1
int ENUM_CFL_STANDARD = -1
int ENUM_CFL_ITER = -1
int ENUM_SIMULATION_WCSPPH = -1
int ENUM_SIMULATION_PCISPPH = -1
int ENUM_SIMULATION_PBF = -1
int ENUM_SIMULATION_IISPPH = -1
int ENUM_SIMULATION_DFSPH = -1
int ENUM_SIMULATION_PF = -1
int BOUNDARY_HANDLING_METHOD = -1
int ENUM_AKINCI2012 = -1
int ENUM_KOSCHIER2017 = -1
int ENUM_BENDER2019 = -1
```

## Protected Functions

void **initParameters** ()

## Protected Attributes

```
std::vector<FluidModel*> m_fluidModels
std::vector<BoundaryModel*> m_boundaryModels
NeighborhoodSearch *m_neighborhoodSearch
AnimationFieldSystem *m_animationFieldSystem
int m_cflMethod
Real m_cflFactor
Real m_cflMinTimeStepSize
Real m_cflMaxTimeStepSize
int m_kernelMethod
int m_gradKernelMethod
Real m_W_zero
Real (*m_kernelFct) (const Vector3r&)
Vector3r (*m_gradKernelFct) (const Vector3r &r)
SimulationMethods m_simulationMethod
TimeStep *m_timeStep
Vector3r m_gravitation
Real m_particleRadius
Real m_supportRadius
bool m_sim2D
bool m_enableZSort
std::function<void ()> m_simulationMethodChanged
int m_boundaryHandlingMethod
```

## Class SimulationDataDFSPH

- Defined in file `_SPlisHSPlasH_DFSPH_SimulationDataDFSPH.h`

## Class Documentation

### class SPH::SimulationDataDFSPH

*Simulation* data which is required by the method Divergence-free Smoothed Particle Hydrodynamics introduced by Bender and Koschier [5], [6].

#### Public Functions

**SimulationDataDFSPH()**

**~SimulationDataDFSPH()**

void **init** ()

Initialize the arrays containing the particle data.

void **cleanup** ()

Release the arrays containing the particle data.

void **reset** ()

Reset the particle data.

void **performNeighborhoodSearchSort** ()

Important: First call `m_model->performNeighborhoodSearchSort()` to call the `z_sort` of the neighborhood search.

void **emittedParticles** (*FluidModel* \*model, const unsigned int startIndex)

**FORCE\_INLINE** const Real getFactor (const unsigned int fluidIndex, const unsigned int i)

**FORCE\_INLINE** Real & getFactor (const unsigned int fluidIndex, const unsigned int i)

**FORCE\_INLINE** void setFactor (const unsigned int fluidIndex, const unsigned int i, const

**FORCE\_INLINE** const Real getKappa (const unsigned int fluidIndex, const unsigned int i)

**FORCE\_INLINE** Real & getKappa (const unsigned int fluidIndex, const unsigned int i)

**FORCE\_INLINE** void setKappa (const unsigned int fluidIndex, const unsigned int i, const

**FORCE\_INLINE** const Real getKappaV (const unsigned int fluidIndex, const unsigned int i)

**FORCE\_INLINE** Real & getKappaV (const unsigned int fluidIndex, const unsigned int i)

**FORCE\_INLINE** void setKappaV (const unsigned int fluidIndex, const unsigned int i, const

**FORCE\_INLINE** const Real getDensityAdv (const unsigned int fluidIndex, const unsigned int i)

**FORCE\_INLINE** Real & getDensityAdv (const unsigned int fluidIndex, const unsigned int i)

**FORCE\_INLINE** void setDensityAdv (const unsigned int fluidIndex, const unsigned int i, c

#### Protected Attributes

std::vector<std::vector<*Real*>> **m\_factor**

factor  $\alpha_i$  [5]

std::vector<std::vector<*Real*>> **m\_kappa**

stores  $\kappa$  value of last time step for a warm start of the pressure solver

std::vector<std::vector<*Real*>> **m\_kappaV**

stores  $\kappa^v$  value of last time step for a warm start of the divergence solver

```
std::vector<std::vector<Real>> m_density_adv
    advected density
```

## Class SimulationDataIISPH

- Defined in file\_SPlisHSPlasH\_IISPH\_SimulationDataIISPH.h

## Class Documentation

**class** SPH::SimulationDataIISPH

*Simulation* data which is required by the method Implicit Incompressible SPH introduced by Ihmsen et al. [13].

### Public Functions

**SimulationDataIISPH** ()

**~SimulationDataIISPH** ()

void **init** ()

Initialize the arrays containing the particle data.

void **cleanup** ()

Release the arrays containing the particle data.

void **reset** ()

Reset the particle data.

void **performNeighborhoodSearchSort** ()

Important: First call `m_model->performNeighborhoodSearchSort()` to call the `z_sort` of the neighborhood search.

void **emittedParticles** (*FluidModel* \*model, const unsigned int startIndex)

**FORCE\_INLINE** const Real **getAii** (const unsigned int fluidIndex, const unsigned int i) const

**FORCE\_INLINE** Real & **getAii** (const unsigned int fluidIndex, const unsigned int i)

**FORCE\_INLINE** void **setAii** (const unsigned int fluidIndex, const unsigned int i, const Real aii)

**FORCE\_INLINE** Vector3r & **getDii** (const unsigned int fluidIndex, const unsigned int i)

**FORCE\_INLINE** const Vector3r & **getDii** (const unsigned int fluidIndex, const unsigned int i)

**FORCE\_INLINE** void **setDii** (const unsigned int fluidIndex, const unsigned int i, const Vector3r dii)

**FORCE\_INLINE** Vector3r & **getDij\_pj** (const unsigned int fluidIndex, const unsigned int i, const unsigned int j)

**FORCE\_INLINE** const Vector3r & **getDij\_pj** (const unsigned int fluidIndex, const unsigned int i, const unsigned int j)

**FORCE\_INLINE** void **setDij\_pj** (const unsigned int fluidIndex, const unsigned int i, const unsigned int j, const Vector3r dij\_pj)

**FORCE\_INLINE** const Real **getDensityAdv** (const unsigned int fluidIndex, const unsigned int i)

**FORCE\_INLINE** Real & **getDensityAdv** (const unsigned int fluidIndex, const unsigned int i)

**FORCE\_INLINE** void **setDensityAdv** (const unsigned int fluidIndex, const unsigned int i, const Real densityAdv)

**FORCE\_INLINE** const Real **getPressure** (const unsigned int fluidIndex, const unsigned int i)

**FORCE\_INLINE** Real & **getPressure** (const unsigned int fluidIndex, const unsigned int i)

**FORCE\_INLINE** void **setPressure** (const unsigned int fluidIndex, const unsigned int i, const Real pressure)

```
FORCE_INLINE const Real getLastPressure (const unsigned int fluidIndex, const unsigned int i)
FORCE_INLINE Real & getLastPressure (const unsigned int fluidIndex, const unsigned int i)
FORCE_INLINE void setLastPressure (const unsigned int fluidIndex, const unsigned int i, const Real p)
FORCE_INLINE Vector3r & getPressureAccel (const unsigned int fluidIndex, const unsigned int i)
FORCE_INLINE const Vector3r & getPressureAccel (const unsigned int fluidIndex, const unsigned int i)
FORCE_INLINE void setPressureAccel (const unsigned int fluidIndex, const unsigned int i, const Vector3r a)
```

### Protected Attributes

```
std::vector<std::vector<Real>>> m_aei
std::vector<std::vector<Vector3r>>> m_dii
std::vector<std::vector<Vector3r>>> m_dij_pj
std::vector<std::vector<Real>>> m_density_adv
std::vector<std::vector<Real>>> m_pressure
std::vector<std::vector<Real>>> m_lastPressure
std::vector<std::vector<Vector3r>>> m_pressureAccel
```

### Class SimulationDataPBF

- Defined in file `_SPlisHSPlasH_PBF_SimulationDataPBF.h`

### Class Documentation

**class** SPH::SimulationDataPBF

*Simulation* data which is required by the method Position-Based Fluids introduced by Macklin and Mueller [17], [7], [8].

### Public Functions

**SimulationDataPBF** ()

**~SimulationDataPBF** ()

void **init** ()  
Initialize the arrays containing the particle data.

void **cleanup** ()  
Release the arrays containing the particle data.

void **reset** ()  
Reset the particle data.

void **performNeighborhoodSearchSort** ()  
Important: First call `m_model->performNeighborhoodSearchSort()` to call the `z_sort` of the neighborhood search.

void **emittedParticles** (*FluidModel* \*model, const unsigned int startIndex)

**FORCE\_INLINE** const Real & getLambda (const unsigned int fluidIndex, const unsigned int i)



```

FORCE_INLINE Real & getLambda (const unsigned int fluidIndex, const unsigned int i)
FORCE_INLINE void setLambda (const unsigned int fluidIndex, const unsigned int i, const
FORCE_INLINE Vector3r & getDeltaX (const unsigned int fluidIndex, const unsigned int i)
FORCE_INLINE const Vector3r & getDeltaX (const unsigned int fluidIndex, const unsigned
FORCE_INLINE void setDeltaX (const unsigned int fluidIndex, const unsigned int i, const
FORCE_INLINE Vector3r & getLastPosition (const unsigned int fluidIndex, const unsigned
FORCE_INLINE const Vector3r & getLastPosition (const unsigned int fluidIndex, const un
FORCE_INLINE void setLastPosition (const unsigned int fluidIndex, const unsigned int i
FORCE_INLINE Vector3r & getOldPosition (const unsigned int fluidIndex, const unsigned
FORCE_INLINE const Vector3r & getOldPosition (const unsigned int fluidIndex, const uns
FORCE_INLINE void setOldPosition (const unsigned int fluidIndex, const unsigned int i,

```

### Protected Attributes

```

std::vector<std::vector<Real>>> m_lambda
std::vector<std::vector<Vector3r>>> m_deltaX
std::vector<std::vector<Vector3r>>> m_oldX
std::vector<std::vector<Vector3r>>> m_lastX

```

### Class SimulationDataPCISPH

- Defined in file\_SPlisHSPlasH\_PCISPH\_SimulationDataPCISPH.h

### Class Documentation

**class** SPH::SimulationDataPCISPH

*Simulation* data which is required by the method Predictive-corrective Incompressible SPH introduced by Solenthaler and Pajarola [23].

### Public Functions

**SimulationDataPCISPH** ()

**~SimulationDataPCISPH** ()

void **init** ()  
Initialize the arrays containing the particle data.

void **cleanup** ()  
Release the arrays containing the particle data.

void **reset** ()  
Reset the particle data.

void **performNeighborhoodSearchSort** ()  
Important: First call m\_model->*performNeighborhoodSearchSort()* to call the z\_sort of the neighborhood search.

```
Real getPCISPH_ScalingFactor (const unsigned int fluidIndex)
void emittedParticles (FluidModel *model, const unsigned int startIndex)
FORCE_INLINE Vector3r & getLastPosition (const unsigned int fluidIndex, const unsigned int i)
FORCE_INLINE const Vector3r & getLastPosition (const unsigned int fluidIndex, const unsigned int i)
FORCE_INLINE void setLastPosition (const unsigned int fluidIndex, const unsigned int i, const Vector3r & pos)
FORCE_INLINE Vector3r & getLastVelocity (const unsigned int fluidIndex, const unsigned int i)
FORCE_INLINE const Vector3r & getLastVelocity (const unsigned int fluidIndex, const unsigned int i)
FORCE_INLINE void setLastVelocity (const unsigned int fluidIndex, const unsigned int i, const Vector3r & vel)
FORCE_INLINE const Real & getDensityAdv (const unsigned int fluidIndex, const unsigned int i)
FORCE_INLINE Real & getDensityAdv (const unsigned int fluidIndex, const unsigned int i)
FORCE_INLINE void setDensityAdv (const unsigned int fluidIndex, const unsigned int i, const Real & adv)
FORCE_INLINE const Real & getPressure (const unsigned int fluidIndex, const unsigned int i)
FORCE_INLINE Real & getPressure (const unsigned int fluidIndex, const unsigned int i)
FORCE_INLINE void setPressure (const unsigned int fluidIndex, const unsigned int i, const Real & pres)
FORCE_INLINE Vector3r & getPressureAccel (const unsigned int fluidIndex, const unsigned int i)
FORCE_INLINE const Vector3r & getPressureAccel (const unsigned int fluidIndex, const unsigned int i)
FORCE_INLINE void setPressureAccel (const unsigned int fluidIndex, const unsigned int i, const Vector3r & acc)
```

### Protected Attributes

```
std::vector<Real> m_pcisph_factor
std::vector<std::vector<Vector3r>> m_lastX
std::vector<std::vector<Vector3r>> m_lastV
std::vector<std::vector<Real>> m_densityAdv
std::vector<std::vector<Real>> m_pressure
std::vector<std::vector<Vector3r>> m_pressureAccel
```

### Class SimulationDataPF

- Defined in file `_SPlisHSPlasH_PF_SimulationDataPF.h`

### Class Documentation

**class** SPH::SimulationDataPF

*Simulation* data which is required by the method Projective Fluids introduced by Weiler, Koschier and Bender [25].

## Public Functions

**SimulationDataPF()**

**~SimulationDataPF()**

void **init** ()

Initialize the arrays containing the particle data.

void **cleanup** ()

Release the arrays containing the particle data.

void **reset** ()

Reset the particle data.

void **performNeighborhoodSearchSort** ()

Important: First call `m_model->performNeighborhoodSearchSort()` to call the `z_sort` of the neighborhood search.

void **emittedParticles** (*FluidModel \*model*, const unsigned int *startIndex*)

**FORCE\_INLINE** const Vector3r getOldPosition (const unsigned int fluidIndex, const unsigned int i)

**FORCE\_INLINE** Vector3r & getOldPosition (const unsigned int fluidIndex, const unsigned int i)

**FORCE\_INLINE** void setOldPosition (const unsigned int fluidIndex, const unsigned int i, const Vector3r &pos)

**FORCE\_INLINE** const unsigned int getNumFluidNeighbors (const unsigned int fluidIndex, const unsigned int i)

**FORCE\_INLINE** unsigned int & getNumFluidNeighbors (const unsigned int fluidIndex, const unsigned int i)

**FORCE\_INLINE** void setNumFluidNeighbors (const unsigned int fluidIndex, const unsigned int i, const unsigned int &val)

**FORCE\_INLINE** const Vector3r & getS (const unsigned int fluidIndex, const unsigned int i)

**FORCE\_INLINE** Vector3r & getS (const unsigned int fluidIndex, const unsigned int i)

**FORCE\_INLINE** void setS (const unsigned int fluidIndex, const unsigned int i, const Vector3r &pos)

**FORCE\_INLINE** const Vector3r & getDiag (const unsigned int fluidIndex, const unsigned int i)

**FORCE\_INLINE** Vector3r & getDiag (const unsigned int fluidIndex, const unsigned int i)

**FORCE\_INLINE** void setDiag (const unsigned int fluidIndex, const unsigned int i, const Vector3r &pos)

**FORCE\_INLINE** const unsigned int & getParticleOffset (const unsigned int fluidIndex) const

## Protected Attributes

std::vector<std::vector<*Vector3r*>> **m\_old\_position**

particle position from last timestep

std::vector<std::vector<unsigned int>> **m\_num\_fluid\_neighbors**

number of neighbors that are fluid particles

std::vector<std::vector<*Vector3r*>> **m\_s**

positions predicted from momentum

std::vector<std::vector<*Vector3r*>> **m\_mat\_diag**

diagonal of system matrix, used by preconditioner

std::vector<unsigned int> **m\_particleOffset**

## Class SimulationDataWCSPH

- Defined in file\_SPlisHSPlasH\_WCSPH\_SimulationDataWCSPH.h

## Class Documentation

**class** SPH::SimulationDataWCSPH

*Simulation* data which is required by the method Weakly Compressible SPH for Free Surface Flows introduced by Becker and Teschner [3].

### Public Functions

**SimulationDataWCSPH** ()

**~SimulationDataWCSPH** ()

void **init** ()

Initialize the arrays containing the particle data.

void **cleanup** ()

Release the arrays containing the particle data.

void **reset** ()

Reset the particle data.

void **performNeighborhoodSearchSort** ()

Important: First call `m_model->performNeighborhoodSearchSort()` to call the `z_sort` of the neighborhood search.

void **emittedParticles** (*FluidModel* \*model, const unsigned int startIndex)

**FORCE\_INLINE** const Real getPressure (const unsigned int fluidIndex, const unsigned int i)

**FORCE\_INLINE** Real & getPressure (const unsigned int fluidIndex, const unsigned int i)

**FORCE\_INLINE** void setPressure (const unsigned int fluidIndex, const unsigned int i, const Real p)

**FORCE\_INLINE** Vector3r & getPressureAccel (const unsigned int fluidIndex, const unsigned int i)

**FORCE\_INLINE** const Vector3r & getPressureAccel (const unsigned int fluidIndex, const unsigned int i)

**FORCE\_INLINE** void setPressureAccel (const unsigned int fluidIndex, const unsigned int i, const Vector3r a)

### Protected Attributes

std::vector<std::vector<*Real*>> m\_pressure

std::vector<std::vector<*Vector3r*>> m\_pressureAccel

## Class SpikyKernel

- Defined in file\_SPlisHSPlasH\_SPHKernels.h

## Class Documentation

**class** SPH::SpikyKernel

Spiky kernel.

### Public Static Functions

*Real* **getRadius** ()

void **setRadius** (*Real* val)

*Real* **W** (const *Real* r)

$W(r,h) = 15/(pi*h^6) * (h-r)^3$

*Real* **W** (const *Vector3r* &r)

*Vector3r* **gradW** (const *Vector3r* &r)

$grad(W(r,h)) = -r(45/(pi*h^6) * (h-r)^2)$

*Real* **W\_zero** ()

### Protected Static Attributes

*Real* **m\_radius**

*Real* **m\_k**

*Real* **m\_l**

*Real* **m\_W\_zero**

## Class StaticRigidBody

- Defined in file\_SPlisHSPlasH\_StaticRigidBody.h

## Inheritance Relationships

### Base Type

- public SPH::RigidBodyObject (*Class RigidBodyObject*)

## Class Documentation

**class** SPH::StaticRigidBody : public SPH::RigidBodyObject

This class stores the information of a static rigid body which is not part of a rigid body simulation.

### Public Functions

**StaticRigidBody** ()

bool **isDynamic** () const

*Real* const **getMass** () const

*Vector3r* const &**getPosition** () const

void **setPosition** (const *Vector3r* &x)

*Vector3r* **getWorldSpacePosition** () const

*Vector3r* const &**getVelocity** () const

void **setVelocity** (const *Vector3r* &v)

*Matrix3r* const &**getRotation** () const

void **setRotation** (const *Matrix3r* &r)

*Matrix3r* **getWorldSpaceRotation** () const

*Vector3r* const &**getAngularVelocity** () const

void **setAngularVelocity** (const *Vector3r* &v)

void **addForce** (const *Vector3r* &f)

void **addTorque** (const *Vector3r* &t)

const std::vector<*Vector3r*> &**getVertices** () const

const std::vector<*Vector3r*> &**getVertexNormals** () const

const std::vector<unsigned int> &**getFaces** () const

void **setWorldSpacePosition** (const *Vector3r* &x)

void **setWorldSpaceRotation** (const *Matrix3r* &r)

*TriangleMesh* &**getGeometry** ()

### Protected Attributes

*Vector3r* m\_x

*Vector3r* m\_x\_world

*Vector3r* m\_zero

*Matrix3r* m\_R

*Matrix3r* m\_R\_world

*TriangleMesh* m\_geometry

## Class SurfaceTension\_Akinci2013

- Defined in file\_SPlisHSPlasH\_SurfaceTension\_SurfaceTension\_Akinci2013.h

## Inheritance Relationships

### Base Type

- `public SPH::SurfaceTensionBase (Class SurfaceTensionBase)`

## Class Documentation

**class** `SPH::SurfaceTension_Akinci2013` : **public** `SPH::SurfaceTensionBase`

This class implements the surface tension method introduced by Akinci et al. [2].

### Public Functions

**SurfaceTension\_Akinci2013** (*FluidModel* \*model)

**~SurfaceTension\_Akinci2013** (void)

void **step** ()

void **reset** ()

void **computeNormals** ()

void **performNeighborhoodSearchSort** ()

**FORCE\_INLINE** `Vector3r` & **getNormal** (const unsigned int i)

**FORCE\_INLINE** const `Vector3r` & **getNormal** (const unsigned int i) const

**FORCE\_INLINE** void **setNormal** (const unsigned int i, const `Vector3r` &val)

### Protected Attributes

std::vector<*Vector3r*> **m\_normals**

## Class SurfaceTension\_Becker2007

- Defined in file\_SPlisHSPlasH\_SurfaceTension\_SurfaceTension\_Becker2007.h

## Inheritance Relationships

### Base Type

- `public SPH::SurfaceTensionBase (Class SurfaceTensionBase)`

## Class Documentation

**class** SPH::SurfaceTension\_Becker2007 : public SPH::SurfaceTensionBase

This class implements the surface tension method introduced by Becker and Teschner [3].

### Public Functions

SurfaceTension\_Becker2007 (*FluidModel* \*model)

~SurfaceTension\_Becker2007 (void)

void **step** ()

void **reset** ()

## Class SurfaceTension\_He2014

- Defined in file\_SPlisHSPlasH\_SurfaceTension\_SurfaceTension\_He2014.h

## Inheritance Relationships

### Base Type

- public SPH::SurfaceTensionBase (*Class SurfaceTensionBase*)

## Class Documentation

**class** SPH::SurfaceTension\_He2014 : public SPH::SurfaceTensionBase

This class implements the surface tension method introduced by He et al. [12].

### Public Functions

SurfaceTension\_He2014 (*FluidModel* \*model)

~SurfaceTension\_He2014 (void)

void **step** ()

void **reset** ()

void **performNeighborhoodSearchSort** ()

**FORCE\_INLINE** const Real getColor (const unsigned int i) const

**FORCE\_INLINE** Real & getColor (const unsigned int i)

**FORCE\_INLINE** void setColor (const unsigned int i, const Real p)

**FORCE\_INLINE** const Real getGradC2 (const unsigned int i) const

**FORCE\_INLINE** Real & getGradC2 (const unsigned int i)

**FORCE\_INLINE** void setGradC2 (const unsigned int i, const Real p)



### Protected Attributes

```
std::vector<Real> m_color
std::vector<Real> m_gradC2
```

### Class SurfaceTensionBase

- Defined in file\_SPlisHSPlasH\_SurfaceTension\_SurfaceTensionBase.h

### Inheritance Relationships

#### Base Type

- public SPH::NonPressureForceBase (*Class NonPressureForceBase*)

#### Derived Types

- public SPH::SurfaceTension\_Akinci2013 (*Class SurfaceTension\_Akinci2013*)
- public SPH::SurfaceTension\_Becker2007 (*Class SurfaceTension\_Becker2007*)
- public SPH::SurfaceTension\_He2014 (*Class SurfaceTension\_He2014*)

### Class Documentation

```
class SPH::SurfaceTensionBase : public SPH::NonPressureForceBase
```

Base class for all surface tension methods.

Subclassed by *SPH::SurfaceTension\_Akinci2013*, *SPH::SurfaceTension\_Becker2007*,  
*SPH::SurfaceTension\_He2014*

#### Public Functions

```
SurfaceTensionBase (FluidModel *model)
```

```
~SurfaceTensionBase (void)
```

#### Public Static Attributes

```
int SURFACE_TENSION = -1
```

```
int SURFACE_TENSION_BOUNDARY = -1
```

### Protected Functions

void **initParameters** ()

### Protected Attributes

*Real* **m\_surfaceTension**

*Real* **m\_surfaceTensionBoundary**

### Class TimeIntegration

- Defined in file\_SPlisHSPlasH\_PBF\_TimeIntegration.h

### Class Documentation

**class** SPH::TimeIntegration

Class for the position-based fluids time integration.

### Public Static Functions

void **semiImplicitEuler** (**const** *Real* *h*, **const** *Real* *mass*, *Vector3r* &*position*, *Vector3r* &*velocity*, **const** *Vector3r* &*acceleration*)

Perform an integration step for a particle using the semi-implicit Euler (symplectic Euler) method:

$$\begin{aligned}\mathbf{v}(t+h) &= \mathbf{v}(t) + \mathbf{a}(t)h \\ \mathbf{x}(t+h) &= \mathbf{x}(t) + \mathbf{v}(t+h)h\end{aligned}$$

### Parameters

- *h*: time step size
- *mass*: mass of the particle
- *position*: position of the particle
- *velocity*: velocity of the particle
- *acceleration*: acceleration of the particle

void **velocityUpdateFirstOrder** (**const** *Real* *h*, **const** *Real* *mass*, **const** *Vector3r* &*position*, **const** *Vector3r* &*oldPosition*, *Vector3r* &*velocity*)

Perform a velocity update (first order) for the linear velocity:

$$\mathbf{v}(t+h) = \frac{1}{h}(\mathbf{p}(t+h) - \mathbf{p}(t))$$

### Parameters

- *h*: time step size

- **mass**: mass of the particle
- **position**: new position  $\mathbf{p}(t + h)$  of the particle
- **oldPosition**: position  $\mathbf{p}(t)$  of the particle before the time step
- **velocity**: resulting velocity of the particle

```
void velocityUpdateSecondOrder (const Real h, const Real mass, const Vector3r &position,
                                const Vector3r &oldPosition, const Vector3r &positionOfLast-
                                Step, Vector3r &velocity)
```

## Class TimeManager

- Defined in file\_SPlisHSPlasH\_TimeManager.h

## Class Documentation

```
class SPH::TimeManager
```

Class to manage the current simulation time and the time step size. This class is a singleton.

### Public Functions

```
TimeManager ()
```

```
~TimeManager ()
```

```
Real getTime ()
```

```
void setTime (Real t)
```

```
Real getTimeStepSize ()
```

```
void setTimeStepSize (Real tss)
```

```
void saveState (BinaryFileWriter &binWriter)
```

```
void loadState (BinaryFileReader &binReader)
```

### Public Static Functions

```
TimeManager *getCurrent ()
```

```
void setCurrent (TimeManager *tm)
```

```
bool hasCurrent ()
```

## Class TimeStep

- Defined in file\_SPlisHSPlasH\_TimeStep.h

## Inheritance Relationships

### Base Type

- public ParameterObject

### Derived Types

- public SPH::TimeStepDFSPH (*Class TimeStepDFSPH*)
- public SPH::TimeStepIISPH (*Class TimeStepIISPH*)
- public SPH::TimeStepPBF (*Class TimeStepPBF*)
- public SPH::TimeStepPCISPH (*Class TimeStepPCISPH*)
- public SPH::TimeStepPF (*Class TimeStepPF*)
- public SPH::TimeStepWCSPH (*Class TimeStepWCSPH*)

## Class Documentation

**class** SPH::TimeStep : public ParameterObject

Base class for the simulation methods.

Subclassed by *SPH::TimeStepDFSPH*, *SPH::TimeStepIISPH*, *SPH::TimeStepPBF*, *SPH::TimeStepPCISPH*, *SPH::TimeStepPF*, *SPH::TimeStepWCSPH*

### Public Functions

**TimeStep** ()

**~TimeStep** (void)

void **step** () = 0

void **reset** ()

void **init** ()

void **resize** () = 0

void **emittedParticles** (*FluidModel \*model*, **const** unsigned int *startIndex*)

void **saveState** (BinaryFileWriter &*binWriter*)

void **loadState** (BinaryFileReader &*binReader*)

### Public Static Attributes

```
int SOLVER_ITERATIONS = -1
int MIN_ITERATIONS = -1
int MAX_ITERATIONS = -1
int MAX_ERROR = -1
```

### Protected Functions

```
void clearAccelerations (const unsigned int fluidModelIndex)
    Clear accelerations and add gravitation.

void computeDensities (const unsigned int fluidModelIndex)
    Determine densities of all fluid particles.

void initParameters ()

void approximateNormal (Discregrid::DiscreteGrid *map, const Eigen::Vector3d &x, Vector3r
    &n, const unsigned int dim)

void computeVolumeAndBoundaryX (const unsigned int fluidModelIndex, const unsigned int i,
    const Vector3r &xi)

void computeVolumeAndBoundaryX ()

void computeDensityAndGradient (const unsigned int fluidModelIndex, const unsigned int i,
    const Vector3r &xi)

void computeDensityAndGradient ()
```

### Protected Attributes

```
unsigned int m_iterations
Real m_maxError
unsigned int m_minIterations
unsigned int m_maxIterations
```

### Class TimeStepDFSPH

- Defined in file\_SPlisHSPlasH\_DFSPH\_TimeStepDFSPH.h

### Inheritance Relationships

#### Base Type

- public SPH::TimeStep (*Class TimeStep*)

## Class Documentation

**class** SPH::TimeStepDFSPH : public SPH::TimeStep

This class implements the Divergence-free Smoothed Particle Hydrodynamics approach introduced by Bender and Koschier [5], [6], [16].

### Public Functions

**TimeStepDFSPH** ()

**~TimeStepDFSPH** (void)

void **step** ()

void **reset** ()

void **resize** ()

### Public Static Attributes

int **SOLVER\_ITERATIONS\_V** = -1

int **MAX\_ITERATIONS\_V** = -1

int **MAX\_ERROR\_V** = -1

int **USE\_DIVERGENCE\_SOLVER** = -1

### Protected Functions

void **computeDFSPHFactor** (const unsigned int *fluidModelIndex*)

void **pressureSolve** ()

void **pressureSolveIteration** (const unsigned int *fluidModelIndex*, *Real* &*avg\_density\_err*)

void **divergenceSolve** ()

void **divergenceSolveIteration** (const unsigned int *fluidModelIndex*, *Real* &*avg\_density\_err*)

void **computeDensityAdv** (const unsigned int *fluidModelIndex*, const unsigned int *index*, const int *numParticles*, const *Real* *h*, const *Real* *density0*)

void **computeDensityChange** (const unsigned int *fluidModelIndex*, const unsigned int *index*, const *Real* *h*)

void **warmstartDivergenceSolve** (const unsigned int *fluidModelIndex*)

void **warmstartPressureSolve** (const unsigned int *fluidModelIndex*)

void **performNeighborhoodSearch** ()

Perform the neighborhood search for all fluid particles.

void **emittedParticles** (*FluidModel* \**model*, const unsigned int *startIndex*)

void **initParameters** ()

### Protected Attributes

```
SimulationDataDFSPH m_simulationData
unsigned int m_counter
const Real m_eps = static_cast<Real>(1.0e-5)
bool m_enableDivergenceSolver
unsigned int m_iterationsV
Real m_maxErrorV
unsigned int m_maxIterationsV
```

### Class TimeStepIISPH

- Defined in file\_SPlisHSPlasH\_IISPH\_TimeStepIISPH.h

### Inheritance Relationships

#### Base Type

- public SPH::TimeStep (*Class TimeStep*)

### Class Documentation

```
class SPH::TimeStepIISPH : public SPH::TimeStep
```

This class implements the Implicit Incompressible SPH approach introduced by Ihmsen et al. [13].

#### Public Functions

```
TimeStepIISPH()
~TimeStepIISPH(void)
void step()
void reset()
void resize()
const SimulationDataIISPH &getSimulationData()
```

#### Protected Functions

```
void predictAdvection(const unsigned int fluidModelIndex)
void pressureSolve()
void pressureSolveIteration(const unsigned int fluidModelIndex, Real &avg_density_err)
void integration(const unsigned int fluidModelIndex)
void computePressureAccels(const unsigned int fluidModelIndex)
    Determine the pressure accelerations when the pressure is already known.
```

```
void performNeighborhoodSearch ()
    Perform the neighborhood search for all fluid particles.

void emittedParticles (FluidModel *model, const unsigned int startIndex)
```

### Protected Attributes

```
SimulationDataISPH m_simulationData
unsigned int m_counter
```

## Class TimeStepPBF

- Defined in file `_SPlisHSPlasH_PBF_TimeStepPBF.h`

## Inheritance Relationships

### Base Type

- `public SPH::TimeStep` (*Class TimeStep*)

## Class Documentation

```
class SPH::TimeStepPBF : public SPH::TimeStep
```

This class implements the position-based fluids approach introduced by Macklin and Mueller [17], [7], [8].

### Public Functions

```
TimeStepPBF ()
    Initialize the simulation data required for this method.

~TimeStepPBF (void)

void step ()
    Perform a simulation step.

void reset ()
    Reset the simulation method.

void resize ()
```

### Public Static Attributes

```
int VELOCITY_UPDATE_METHOD = -1
int ENUM_PBF_FIRST_ORDER = -1
int ENUM_PBF_SECOND_ORDER = -1
```



## Protected Functions

```
void pressureSolve ()
    Perform a position-based correction step for the following density constraint:  $C(\mathbf{x}) = \left(\frac{\rho_i}{\rho_0} - 1\right) = 0$ 

void pressureSolveIteration (const unsigned int fluidModelIndex, Real &avg_density_err)

void performNeighborhoodSearch ()
    Perform the neighborhood search for all fluid particles.

void emittedParticles (FluidModel *model, const unsigned int startIndex)

void initParameters ()
```

## Protected Attributes

```
SimulationDataPBF m_simulationData
unsigned int m_counter
int m_velocityUpdateMethod
```

## Class TimeStepPCISPH

- Defined in file\_SPlisHSPlasH\_PCISPH\_TimeStepPCISPH.h

## Inheritance Relationships

### Base Type

- public SPH::TimeStep (*Class TimeStep*)

## Class Documentation

```
class SPH::TimeStepPCISPH : public SPH::TimeStep
```

This class implements the Predictive-corrective Incompressible SPH approach introduced by Solenthaler and Pajarola [23].

### Public Functions

```
TimeStepPCISPH ()
~TimeStepPCISPH (void)
void step ()
void reset ()
void resize ()
```

### Protected Functions

```
void pressureSolve ()
void pressureSolveIteration (const unsigned int fluidModelIndex, Real &avg_density_err)
void performNeighborhoodSearch ()
    Perform the neighborhood search for all fluid particles.
void emittedParticles (FluidModel *model, const unsigned int startIndex)
```

### Protected Attributes

```
SimulationDataPCISPH m_simulationData
unsigned int m_counter
```

### Class TimeStepPF

- Defined in file `_SPlisHSPlasH_PF_TimeStepPF.h`

### Inheritance Relationships

#### Base Type

- `public SPH::TimeStep` (*Class TimeStep*)

### Class Documentation

```
class SPH::TimeStepPF : public SPH::TimeStep
```

This class implements the Projective Fluids approach introduced by Weiler, Koschier and Bender [25].

#### Public Functions

```
TimeStepPF ()
~TimeStepPF (void)
void step () override
void reset () override
void resize () override
```

## Public Static Functions

void **matrixVecProd** (const *Real* \*vec, *Real* \*result, void \*userData)

## Public Static Attributes

int **STIFFNESS** = -1

## Protected Types

using **VectorXr** = Eigen::Matrix<*Real*, -1, 1>

using **VectorXrMap** = Eigen::Map<*VectorXr*>

using **Solver** = Eigen::ConjugateGradient<*MatrixReplacement*, Eigen::Lower | Eigen::Upper, *JacobiPreconditioner3D*>

## Protected Functions

void **preparePreconditioner** ()

void **initialGuessForPositions** (const unsigned int *fluidModelIndex*)

void **solvePDConstraints** ()

void **updatePositionsAndVelocity** (const *VectorXr* &x)

void **addAccelerationToVelocity** ()

void **matrixFreeRHS** (const *VectorXr* &x, *VectorXr* &result)

compute the right hand side of the system in a matrix-free fashion and store the result in result

void **performNeighborhoodSearch** ()

Perform the neighborhood search for all fluid particles.

void **emittedParticles** (*FluidModel* \*model, const unsigned int *startIndex*) **override**

void **initParameters** () **override**

## Protected Attributes

*SimulationDataPF* **m\_simulationData**

*Solver* **m\_solver**

*Real* **m\_stiffness**

unsigned int **m\_counter**

unsigned int **m\_numActiveParticlesTotal**

### Protected Static Functions

```
FORCE_INLINE void diagonalMatrixElement (const unsigned int row, Vector3r &result, voi
```

### Class TimeStepWCSPH

- Defined in file\_SPlisHSPlasH\_WCSPH\_TimeStepWCSPH.h

### Inheritance Relationships

#### Base Type

- public SPH::TimeStep (*Class TimeStep*)

### Class Documentation

```
class SPH::TimeStepWCSPH : public SPH::TimeStep
```

This class implements the Weakly Compressible SPH for Free Surface Flows approach introduced by Becker and Teschner [3].

#### Public Functions

```
TimeStepWCSPH ()
```

```
~TimeStepWCSPH (void)
```

```
void step ()
```

```
void reset ()
```

```
void resize ()
```

#### Public Static Attributes

```
int STIFFNESS = -1
```

```
int EXPONENT = -1
```

#### Protected Functions

```
void computePressureAccels (const unsigned int fluidModelIndex)  
    Determine the pressure accelerations when the pressure is already known.
```

```
void performNeighborhoodSearch ()  
    Perform the neighborhood search for all fluid particles.
```

```
void emittedParticles (FluidModel *model, const unsigned int startIndex)
```

```
void initParameters ()
```

## Protected Attributes

*Real* **m\_stiffness**  
*Real* **m\_exponent**  
*SimulationDataWCSPH* **m\_simulationData**  
 unsigned int **m\_counter**

## Class TriangleMesh

- Defined in file\_SPlisHSPlasH\_TriangleMesh.h

## Class Documentation

**class** SPH::TriangleMesh

Data structure for a triangle mesh with normals and vertex normals.

## Public Types

**typedef** std::vector<unsigned int> **Faces**  
**typedef** std::vector<*Vector3r*> **Normals**  
**typedef** std::vector<*Vector3r*> **Vertices**

## Public Functions

**TriangleMesh** ()  
**~TriangleMesh** ()  
 void **release** ()  
 void **initMesh** (**const** unsigned int *nPoints*, **const** unsigned int *nFaces*)  
 void **addFace** (**const** unsigned int \***const** *indices*)  
     Add a new face.  
 void **addFace** (**const** int \***const** *indices*)  
     Add a new face.  
 void **addVertex** (**const** *Vector3r* &*vertex*)  
     Add new vertex.  
**const** *Faces* &**getFaces** () **const**  
*Faces* &**getFaces** ()  
**const** *Normals* &**getFaceNormals** () **const**  
*Normals* &**getFaceNormals** ()  
**const** *Normals* &**getVertexNormals** () **const**  
*Normals* &**getVertexNormals** ()  
**const** *Vertices* &**getVertices** () **const**

```

Vertices &getVertices ()
unsigned int numVertices () const
unsigned int numFaces () const
void updateNormals ()
void updateVertexNormals ()

```

### Protected Attributes

```

Vertices m_x
Faces m_indices
Normals m_normals
Normals m_vertexNormals

```

## Class Viscosity\_Bender2017

- Defined in file\_SPlisHSPlasH\_Viscosity\_Viscosity\_Bender2017.h

### Inheritance Relationships

#### Base Type

- public SPH::ViscosityBase (*Class ViscosityBase*)

### Class Documentation

**class** SPH::Viscosity\_Bender2017 : public SPH::ViscosityBase

This class implements the implicit simulation method for viscous fluids introduced by Bender and Koschier [6].

#### Public Functions

```

Viscosity_Bender2017 (FluidModel *model)
~Viscosity_Bender2017 (void)
void step ()
void reset ()
void performNeighborhoodSearchSort ()
void computeTargetStrainRate ()
void computeViscosityFactor ()
FORCE_INLINE void viscoGradientMultTransposeRightOpt (const Eigen::Matrix< Real, 6, 3 > & strainRate,
Matrix product
FORCE_INLINE const Vector6r & getTargetStrainRate (const unsigned int i) const
FORCE_INLINE Vector6r & getTargetStrainRate (const unsigned int i)

```

```

FORCE_INLINE void setTargetStrainRate (const unsigned int i, const Vector6r &val)
FORCE_INLINE const Matrix6r & getViscosityFactor (const unsigned int i) const
FORCE_INLINE Matrix6r & getViscosityFactor (const unsigned int i)
FORCE_INLINE void setViscosityFactor (const unsigned int i, const Matrix6r &val)
FORCE_INLINE const Vector6r & getViscosityLambda (const unsigned int i) const
FORCE_INLINE Vector6r & getViscosityLambda (const unsigned int i)
FORCE_INLINE void setViscosityLambda (const unsigned int i, const Vector6r &val)

```

### Public Static Attributes

```

int ITERATIONS = -1
int MAX_ITERATIONS = -1
int MAX_ERROR = -1

```

### Protected Functions

```

void initParameters ()

```

### Protected Attributes

```

std::vector<Vector6r> m_targetStrainRate
std::vector<Matrix6r> m_viscosityFactor
std::vector<Vector6r> m_viscosityLambda
unsigned int m_iterations
unsigned int m_maxIter
Real m_maxError

```

## Class Viscosity\_Peer2015

- Defined in file\_SPlisHSPlasH\_Viscosity\_Viscosity\_Peer2015.h

## Inheritance Relationships

### Base Type

- public SPH::ViscosityBase (*Class ViscosityBase*)

## Class Documentation

**class** SPH::Viscosity\_Peer2015 : public SPH::ViscosityBase

This class implements the implicit simulation method for viscous fluids introduced by Peer et al. [20].

### Public Functions

Viscosity\_Peer2015 (*FluidModel* \*model)

~Viscosity\_Peer2015 (void)

void **step** ()

void **reset** ()

void **performNeighborhoodSearchSort** ()

**FORCE\_INLINE** const Matrix3r & getTargetNablaV (const unsigned int i) const

**FORCE\_INLINE** Matrix3r & getTargetNablaV (const unsigned int i)

**FORCE\_INLINE** void setTargetNablaV (const unsigned int i, const Matrix3r &val)

### Public Static Functions

void **matrixVecProd** (const *Real* \*vec, *Real* \*result, void \*userData)

**FORCE\_INLINE** void **diagonalMatrixElement** (const unsigned int row, *Real* &result, void \*u

### Public Static Attributes

int **ITERATIONS** = -1

int **MAX\_ITERATIONS** = -1

int **MAX\_ERROR** = -1

### Protected Types

**typedef** Eigen::ConjugateGradient<*MatrixReplacement*, Eigen::Lower | Eigen::Upper, *JacobiPreconditioner1D*> **Solver**

### Protected Functions

void **initParameters** ()

void **computeDensities** ()



### Protected Attributes

```
std::vector<Real> m_density
std::vector<Matrix3r> m_targetNablaV
Solver m_solver
unsigned int m_iterations
unsigned int m_maxIter
Real m_maxError
```

### Class Viscosity\_Peer2016

- Defined in file\_SPlisHSPlasH\_Viscosity\_Viscosity\_Peer2016.h

### Inheritance Relationships

#### Base Type

- public SPH::ViscosityBase (*Class ViscosityBase*)

### Class Documentation

**class** SPH::Viscosity\_Peer2016 : public SPH::ViscosityBase

This class implements the implicit simulation method for viscous fluids introduced by Peer and Teschner [19].

### Public Functions

```
Viscosity_Peer2016 (FluidModel *model)
~Viscosity_Peer2016 (void)
void step ()
void reset ()
void performNeighborhoodSearchSort ()
FORCE_INLINE const Matrix3r & getTargetNablaV (const unsigned int i) const
FORCE_INLINE Matrix3r & getTargetNablaV (const unsigned int i)
FORCE_INLINE void setTargetNablaV (const unsigned int i, const Matrix3r &val)
FORCE_INLINE const Vector3r & getOmega (const unsigned int i) const
FORCE_INLINE Vector3r & getOmega (const unsigned int i)
FORCE_INLINE void setOmega (const unsigned int i, const Vector3r &val)
```

### Public Static Functions

```
void matrixVecProdV (const Real *vec, Real *result, void *userData)
```

```
FORCE_INLINE void diagonalMatrixElementV (const unsigned int row, Real &result, void *
```

```
void matrixVecProdOmega (const Real *vec, Real *result, void *userData)
```

```
FORCE_INLINE void diagonalMatrixElementOmega (const unsigned int row, Real &result, vo
```

### Public Static Attributes

```
int ITERATIONS_V = -1
```

```
int ITERATIONS_OMEGA = -1
```

```
int MAX_ITERATIONS_V = -1
```

```
int MAX_ERROR_V = -1
```

```
int MAX_ITERATIONS_OMEGA = -1
```

```
int MAX_ERROR_OMEGA = -1
```

### Protected Types

```
typedef Eigen::ConjugateGradient<MatrixReplacement, Eigen::Lower | Eigen::Upper, JacobiPreconditioner1D> Solver
```

### Protected Functions

```
void initParameters ()
```

```
void computeDensities ()
```

### Protected Attributes

```
std::vector<Real> m_density
```

```
std::vector<Matrix3r> m_targetNablaV
```

```
std::vector<Vector3r> m_omega
```

```
Solver m_solverV
```

```
Solver m_solverOmega
```

```
unsigned int m_iterationsV
```

```
unsigned int m_iterationsOmega
```

```
unsigned int m_maxIterV
```

```
Real m_maxErrorV
```

```
unsigned int m_maxIterOmega
```

```
Real m_maxErrorOmega
```

## Class Viscosity\_Standard

- Defined in file\_SPlisHSPlasH\_Viscosity\_Viscosity\_Standard.h

## Inheritance Relationships

### Base Type

- public SPH::ViscosityBase (*Class ViscosityBase*)

## Class Documentation

**class** SPH::Viscosity\_Standard: public SPH::ViscosityBase

This class implements the standard method for viscosity described e.g. by Ihmsen et al. [14]. The method evaluates the term  $\nu \nabla^2 \mathbf{v}$  and uses an approximation of the kernel Laplacian to improve the stability. This approximation is given in [14].

### Public Functions

**Viscosity\_Standard** (*FluidModel \*model*)

**~Viscosity\_Standard** (void)

void **step** ()

void **reset** ()

### Public Static Attributes

int **VISCOSITY\_COEFFICIENT\_BOUNDARY** = -1

### Protected Functions

void **initParameters** ()

### Protected Attributes

*Real* **m\_boundaryViscosity**

## Class Viscosity\_Takahashi2015

- Defined in file\_SPlisHSPlasH\_Viscosity\_Viscosity\_Takahashi2015.h

## Inheritance Relationships

### Base Type

- `public SPH::ViscosityBase (Class ViscosityBase)`

### Class Documentation

**class** `SPH::Viscosity_Takahashi2015` : `public SPH::ViscosityBase`

This class implements a variant of the implicit simulation method for viscous fluids introduced by Takahashi et al. [24]. In the original work of Takahashi et al. the second-ring neighbors are required to create the matrix of the linear system. In contrast we use a meshless conjugate gradient solver which performs the required matrix-vector multiplication in two sequential loops. In this way only the one-ring neighbors are required in each loop which increases the performance significantly. Thanks to Anreas Peer who helped us with the implementation.

### Public Functions

**Viscosity\_Takahashi2015** (*FluidModel* \*model)

**~Viscosity\_Takahashi2015** (void)

void **step** ()

void **reset** ()

void **performNeighborhoodSearchSort** ()

**FORCE\_INLINE** const **Matrix3r** & **getViscousStress** (const unsigned int i) const

**FORCE\_INLINE** **Matrix3r** & **getViscousStress** (const unsigned int i)

**FORCE\_INLINE** void **setViscousStress** (const unsigned int i, const **Matrix3r** &val)

**FORCE\_INLINE** const **Vector3r** & **getAccel** (const unsigned int i) const

**FORCE\_INLINE** **Vector3r** & **getAccel** (const unsigned int i)

**FORCE\_INLINE** void **setAccel** (const unsigned int i, const **Vector3r** &val)

### Public Static Functions

void **matrixVecProd** (const *Real* \*vec, *Real* \*result, void \*userData)

**FORCE\_INLINE** void **diagonalMatrixElement** (const unsigned int row, *Real* &result, void \*u

### Public Static Attributes

int **ITERATIONS** = -1

int **MAX\_ITERATIONS** = -1

int **MAX\_ERROR** = -1

### Protected Types

```
typedef Eigen::ConjugateGradient<MatrixReplacement, Eigen::Lower | Eigen::Upper, Eigen::IdentityPreconditioner> Solve
```

### Protected Functions

```
void initParameters ()
```

### Protected Attributes

```
std::vector<Vector3r> m_accel
std::vector<Matrix3r> m_viscousStress
Solver m_solver
unsigned int m_iterations
unsigned int m_maxIter
Real m_maxError
```

### Protected Static Functions

```
void computeViscosityAcceleration (Viscosity_Takahashi2015 *visco, const Real *v)
```

## Class Viscosity\_Weiler2018

- Defined in file\_SPlisHSPlasH\_Viscosity\_Viscosity\_Weiler2018.h

### Inheritance Relationships

#### Base Type

- public SPH::ViscosityBase (*Class ViscosityBase*)

### Class Documentation

```
class SPH::Viscosity_Weiler2018 : public SPH::ViscosityBase
```

This class implements the implicit Laplace viscosity method introduced by Weiler et al. 2018 [26].

#### Public Functions

```
Viscosity_Weiler2018 (FluidModel *model)
~Viscosity_Weiler2018 (void)
void step ()
void reset ()
void performNeighborhoodSearchSort ()
```

### Public Static Functions

```
void matrixVecProd(const Real *vec, Real *result, void *userData)
```

### Public Static Attributes

```
int ITERATIONS = -1  
int MAX_ITERATIONS = -1  
int MAX_ERROR = -1  
int VISCOSITY_COEFFICIENT_BOUNDARY = -1
```

### Protected Types

```
typedef Eigen::ConjugateGradient<MatrixReplacement, Eigen::Lower | Eigen::Upper, BlockJacobiPreconditioner3D> Solver
```

### Protected Functions

```
void initParameters()
```

### Protected Attributes

```
Real m_boundaryViscosity  
unsigned int m_maxIter  
Real m_maxError  
unsigned int m_iterations  
std::vector<Vector3r> m_vDiff  
Solver m_solver
```

### Protected Static Functions

```
FORCE_INLINE void diagonalMatrixElement (const unsigned int row, Matrix3r &result, void *userData)
```

## Class Viscosity\_XSPH

- Defined in file `_SPlisHSPlasH_Viscosity_Viscosity_XSPH.h`

## Inheritance Relationships

### Base Type

- `public SPH::ViscosityBase (Class ViscosityBase)`

### Class Documentation

**class** `SPH::Viscosity_XSPH` : **public** `SPH::ViscosityBase`

This class implements the XSPH method described by Schechter and Bridson [22].

#### Public Functions

**Viscosity\_XSPH** (*FluidModel* \*model)

**~Viscosity\_XSPH** (void)

void **step** ()

void **reset** ()

#### Public Static Attributes

int **VISCOSITY\_COEFFICIENT\_BOUNDARY** = -1

#### Protected Functions

void **initParameters** ()

#### Protected Attributes

*Real* **m\_boundaryViscosity**

### Class ViscosityBase

- Defined in file `_SPlisHSPlasH_Viscosity_ViscosityBase.h`

## Inheritance Relationships

### Base Type

- `public SPH::NonPressureForceBase (Class NonPressureForceBase)`

## Derived Types

- `public SPH::Viscosity_Bender2017` (*Class Viscosity\_Bender2017*)
- `public SPH::Viscosity_Peer2015` (*Class Viscosity\_Peer2015*)
- `public SPH::Viscosity_Peer2016` (*Class Viscosity\_Peer2016*)
- `public SPH::Viscosity_Standard` (*Class Viscosity\_Standard*)
- `public SPH::Viscosity_Takahashi2015` (*Class Viscosity\_Takahashi2015*)
- `public SPH::Viscosity_Weiler2018` (*Class Viscosity\_Weiler2018*)
- `public SPH::Viscosity_XSPH` (*Class Viscosity\_XSPH*)

## Class Documentation

**class** `SPH::ViscosityBase` : `public SPH::NonPressureForceBase`

Base class for all viscosity methods.

Subclassed by `SPH::Viscosity_Bender2017`, `SPH::Viscosity_Peer2015`, `SPH::Viscosity_Peer2016`, `SPH::Viscosity_Standard`, `SPH::Viscosity_Takahashi2015`, `SPH::Viscosity_Weiler2018`, `SPH::Viscosity_XSPH`

### Public Functions

**ViscosityBase** (*FluidModel* \*model)

**~ViscosityBase** (void)

### Public Static Attributes

int **VISCOSITY\_COEFFICIENT** = -1

### Protected Functions

void **initParameters** ()

### Protected Attributes

*Real* **m\_viscosity**

## Class VorticityBase

- Defined in file `_SPlisHSPlasH_Vorticity_VorticityBase.h`



## Inheritance Relationships

### Base Type

- `public SPH::NonPressureForceBase` (*Class NonPressureForceBase*)

### Derived Types

- `public SPH::MicropolarModel_Bender2017` (*Class MicropolarModel\_Bender2017*)
- `public SPH::VorticityConfinement` (*Class VorticityConfinement*)

## Class Documentation

**class** `SPH::VorticityBase` : `public SPH::NonPressureForceBase`

Base class for all vorticity methods.

Subclassed by *SPH::MicropolarModel\_Bender2017*, *SPH::VorticityConfinement*

### Public Functions

**VorticityBase** (*FluidModel* \*model)

**~VorticityBase** (void)

### Public Static Attributes

int **VORTICITY\_COEFFICIENT** = -1

### Protected Functions

void **initParameters** ()

### Protected Attributes

*Real* **m\_vorticityCoeff**

## Class VorticityConfinement

- Defined in file `_SPlisHSPlasH_Vorticity_VorticityConfinement.h`

## Inheritance Relationships

### Base Type

- `public SPH::VorticityBase (Class VorticityBase)`

### Class Documentation

**class** `SPH::VorticityConfinement` : `public SPH::VorticityBase`

This class implements the vorticity confinement method introduced by Macklin and Mueller [17].

#### Public Functions

**VorticityConfinement** (*FluidModel* \*model)

**~VorticityConfinement** (void)

void **step** ()

void **reset** ()

void **performNeighborhoodSearchSort** ()

**FORCE\_INLINE** const Vector3r & **getAngularVelocity** (const unsigned int i) const

**FORCE\_INLINE** Vector3r & **getAngularVelocity** (const unsigned int i)

**FORCE\_INLINE** void **setAngularVelocity** (const unsigned int i, const Vector3r &val)

#### Protected Attributes

std::vector<*Vector3r*> **m\_omega**

std::vector<*Real*> **m\_normOmega**

### Class WendlandQuinticC2Kernel

- Defined in file `_SPlisHSPlasH_SPHKernels.h`

### Class Documentation

**class** `SPH::WendlandQuinticC2Kernel`

quintic Wendland C2 kernel.

### Public Static Functions

*Real* **getRadius** ()  
 void **setRadius** (*Real* val)  
*Real* **W** (const *Real* r)  
*Real* **W** (const *Vector3r* &r)  
*Vector3r* **gradW** (const *Vector3r* &r)  
*Real* **W\_zero** ()

### Protected Static Attributes

*Real* **m\_radius**  
*Real* **m\_k**  
*Real* **m\_l**  
*Real* **m\_W\_zero**

### Class WendlandQuinticC2Kernel2D

- Defined in file\_SPlisHSPlasH\_SPHKernels.h

### Class Documentation

**class** SPH::WendlandQuinticC2Kernel2D  
 Wendland Quintic C2 spline kernel (2D).

### Public Static Functions

*Real* **getRadius** ()  
 void **setRadius** (*Real* val)  
*Real* **W** (const *Real* r)  
*Real* **W** (const *Vector3r* &r)  
*Vector3r* **gradW** (const *Vector3r* &r)  
*Real* **W\_zero** ()

### Protected Static Attributes

*Real* **m\_radius**

*Real* **m\_k**

*Real* **m\_l**

*Real* **m\_W\_zero**

### Class SceneLoader

- Defined in file `_SPlisHSPlasH_Uilities_SceneLoader.h`

### Nested Relationships

#### Nested Types

- *Struct* `SceneLoader::AnimationFieldData`
- *Struct* `SceneLoader::BoundaryData`
- *Struct* `SceneLoader::Box`
- *Struct* `SceneLoader::EmitterData`
- *Struct* `SceneLoader::FluidBlock`
- *Struct* `SceneLoader::FluidData`
- *Struct* `SceneLoader::MaterialData`
- *Struct* `SceneLoader::Scene`

### Class Documentation

**class** `Utilities::SceneLoader`  
Importer of SPlisHSPlasH scene files.

#### Public Functions

void **readScene** (**const** char \**fileName*, *Scene* &*scene*)

template<typename **T**>

bool **readValue** (**const** nlohmann::json &*j*, *T* &*v*)

template<typename **T**, int **size**>

bool **readVector** (**const** nlohmann::json &*j*, Eigen::Matrix<*T*, *size*, 1, Eigen::DontAlign> &*vec*)

template<typename **T**>

bool **readValue** (**const** std::string &*section*, **const** std::string &*key*, *T* &*v*)

template<typename **T**, int **size**>

bool **readVector** (**const** std::string &*section*, **const** std::string &*key*, Eigen::Matrix<*T*, *size*, 1, Eigen::DontAlign> &*vec*)

void **readMaterialParameterObject** (**const** std::string &*key*, GenParam::ParameterObject \**paramObj*)

```

void readParameterObject (const std::string &key, GenParam::ParameterObject *paramObj)
template<>
bool readValue (const nlohmann::json &j, bool &v)
template<>
bool readValue (const nlohmann::json &j, bool &v)

```

### Protected Functions

```

void readParameterObject (nlohmann::json &config, GenParam::ParameterObject *paramObj)

```

### Protected Attributes

```

nlohmann::json m_jsonData
struct AnimationFieldData
    Struct to store an animation field object.

```

#### Public Members

```

std::string particleFieldName
std::string expression[3]
unsigned int shapeType
Vector3r x
Matrix3r rotation
Vector3r scale
Real startTime
Real endTime
struct BoundaryData
    Struct to store a boundary object.

```

#### Public Members

```

std::string samplesFile
std::string meshFile
Vector3r translation
Matrix3r rotation
Vector3r scale
Real density
bool dynamic
bool isWall
Eigen::Matrix<float, 4, 1, Eigen::DontAlign> color
void *rigidBody

```

```
std::string mapFile
bool mapInvert
Real mapThickness
Eigen::Matrix<unsigned int, 3, 1, Eigen::DontAlign> mapResolution
unsigned int samplingMode

struct Box
    Struct for an AABB.
```

### Public Members

```
Vector3r m_minX
Vector3r m_maxX

struct EmitterData
    Struct to store an emitter object.
```

### Public Members

```
std::string id
unsigned int width
unsigned int height
Vector3r x
Real velocity
Matrix3r rotation
Real emitStartTime
Real emitEndTime
unsigned int type

struct FluidBlock
    Struct to store a fluid block.
```

### Public Members

```
std::string id
Box box
unsigned char mode
Vector3r initialVelocity

struct FluidData
    Struct to store a fluid object.
```

**Public Members**

```

std::string id
std::string samplesFile
Vector3r translation
Matrix3r rotation
Vector3r scale
Vector3r initialVelocity
unsigned char mode
bool invert
std::array<unsigned int, 3> resolutionSDF

```

**struct MaterialData**

Struct to store particle coloring information.

**Public Members**

```

std::string id
std::string colorField
unsigned int colorMapType
Real minVal
Real maxVal
unsigned int maxEmitterParticles
bool emitterReuseParticles
Vector3r emitterBoxMin
Vector3r emitterBoxMax

```

**struct Scene**

Struct to store scene information.

**Public Members**

```

std::vector<BoundaryData*> boundaryModels
std::vector<FluidData*> fluidModels
std::vector<FluidBlock*> fluidBlocks
std::vector<EmitterData*> emitters
std::vector<AnimationFieldData*> animatedFields
std::vector<MaterialData*> materials
Real particleRadius
bool sim2D
Real timeStepSize

```

*Vector3r* **camPosition**

*Vector3r* **camLookat**

## Class SDFFunctions

- Defined in file `_SPlisHSPlasH_Uilities_SDFFunctions.h`

## Class Documentation

**class** `Utilities::SDFFunctions`

Functions for generating and querying an SDF.

### Public Static Functions

`Discregrid::CubicLagrangeDiscreteGrid *generateSDF (const unsigned int numVertices, const Vector3r *vertices, const unsigned int numFaces, const unsigned int *faces, const AlignedBox3r &bbox, const std::array<unsigned int, 3> &resolution, const bool invert = false)`

Generate SDF from mesh.

*AlignedBox3r* **computeBoundingBox** (const unsigned int numVertices, const *Vector3r* \*vertices)

Compute the bounding box of a mesh.

double **distance** (Discregrid::CubicLagrangeDiscreteGrid \*sdf, const *Vector3r* &x, const *Real* thickness, *Vector3r* &normal, *Vector3r* &nextSurfacePoint)

Determine distance of a point x to the surface represented by the SDF and corresponding surface normal and next point on the surface.

double **distance** (Discregrid::CubicLagrangeDiscreteGrid \*sdf, const *Vector3r* &x, const *Real* thickness)

Determine distance of a point x to the surface represented by the SDF.

## Class VolumeSampling

- Defined in file `_SPlisHSPlasH_Uilities_VolumeSampling.h`

## Class Documentation

**class** `Utilities::VolumeSampling`

This class implements a volume sampling of 3D models.



## Public Static Functions

void **sampleMesh**(const unsigned int *numVertices*, const *Vector3r* \**vertices*, const unsigned int *numFaces*, const unsigned int \**faces*, const *Real* *radius*, const *AlignedBox3r* \**region*, const std::array<unsigned int, 3> &*resolution*, const bool *invert*, const unsigned int *sampleMode*, std::vector<*Vector3r*> &*samples*)

Performs the volume sampling with the respective parameters.

### Parameters

- *numVertices*: number of vertices
- *vertices*: vertex data
- *numFaces*: number of faces
- *faces*: index list of faces
- *radius*: radius of sampled particles
- *region*: defines a subregion of the mesh to be sampled (nullptr if not used)
- *resolution*: resolution of the used SDF
- *invert*: defines if the mesh should be inverted and the outside is sampled
- *sampleMode*: 0=regular, 1=almost dense, 2=dense
- *samples*: sampled vertices that will be returned

## Class WindingNumbers

- Defined in file `_SPlisHSPlasH_Uilities_WindingNumbers.h`

## Class Documentation

**class** Utilities::WindingNumbers

### Public Static Functions

*Real* **computeGeneralizedWindingNumber**(const *Vector3r* &*p*, const *Vector3r* &*a*, const *Vector3r* &*b*, const *Vector3r* &*c*)

Determine the winding number for a point *p* and a triangle *abc*.

*Real* **computeGeneralizedWindingNumber**(const *Vector3r* &*p*, const SPH::TriangleMesh &*mesh*)

Determine the winding number of a point *p* in a triangle mesh.

## Class Vector3f8

- Defined in file\_SPlisHSPlasH\_Uilities\_AVX\_math.h

## Class Documentation

**class** Vector3f8

### Public Functions

**Vector3f8** ()

**Vector3f8** (const bool)

**Vector3f8** (const *Scalarf8* &x, const *Scalarf8* &y, const *Scalarf8* &z)

**Vector3f8** (const *Scalarf8* &x)

**Vector3f8** (const *Vector3f* &x)

**Vector3f8** (const *Vector3f* &v0, const *Vector3f* &v1, const *Vector3f* &v2, const *Vector3f* &v3, const *Vector3f* &v4, const *Vector3f* &v5, const *Vector3f* &v6, const *Vector3f* &v7)

**Vector3f8** (*Vector3f* const \*x)

void **setZero** ()

*Scalarf8* &**operator**[] (int i)

const *Scalarf8* &**operator**[] (int i) const

*Scalarf8* &**x** ()

*Scalarf8* &**y** ()

*Scalarf8* &**z** ()

const *Scalarf8* &**x** () const

const *Scalarf8* &**y** () const

const *Scalarf8* &**z** () const

*Scalarf8* **dot** (const *Vector3f8* &a) const

*Scalarf8* **operator\*** (const *Vector3f8* &a) const

void **cross** (const *Vector3f8* &a, const *Vector3f8* &b)

const *Vector3f8* **operator%** (const *Vector3f8* &a) const

*Vector3f8* &**operator\*=** (const *Scalarf8* &s)

const *Vector3f8* **operator/** (const *Scalarf8* &s) const

*Vector3f8* &**operator/=** (const *Scalarf8* &s)

*Vector3f8* &**operator--** (const *Vector3f8* &a)

const *Vector3f8* **operator-** () const

*Scalarf8* **squaredNorm** () const

*Scalarf8* **norm** () const

```
void normalize ()
void store (std::vector<Vector3r> &Vf) const
void store (Vector3r *Vf) const
```

### Public Members

*Scalarf8* **v**[3]

### Public Static Functions

*Vector3f8* **blend** (*Scalarf8* **const** &c, *Vector3f8* **const** &a, *Vector3f8* **const** &b)

## 6.3.3 Enums

### Enum BoundaryHandlingMethods

- Defined in file\_SPlisHSPlasH\_Simulation.h

#### Enum Documentation

```
enum SPH::BoundaryHandlingMethods
Values:
    enumerator Akinci2012 = 0
    enumerator Koschier2017
    enumerator Bender2019
    enumerator NumSimulationMethods
```

### Enum DragMethods

- Defined in file\_SPlisHSPlasH\_FluidModel.h

#### Enum Documentation

```
enum SPH::DragMethods
Values:
    enumerator None = 0
    enumerator Macklin2014
    enumerator Gissler2017
    enumerator NumDragMethods
```

### Enum ElasticityMethods

- Defined in file\_SPlisHSPlasH\_FluidModel.h

### Enum Documentation

**enum SPH::ElasticityMethods**

*Values:*

```
enumerator None = 0
enumerator Becker2009
enumerator Peer2018
enumerator NumElasticityMethods
```

### Enum FieldType

- Defined in file\_SPlisHSPlasH\_FluidModel.h

### Enum Documentation

**enum SPH::FieldType**

*Values:*

```
enumerator Scalar = 0
enumerator Vector3
enumerator Vector6
enumerator Matrix3
enumerator Matrix6
enumerator UInt
```

### Enum ParticleState

- Defined in file\_SPlisHSPlasH\_FluidModel.h

### Enum Documentation

**enum SPH::ParticleState**

*Values:*

```
enumerator Active = 0
enumerator AnimatedByEmitter
```

## Enum SimulationMethods

- Defined in file\_SPlisHSPlasH\_Simulation.h

## Enum Documentation

```
enum SPH::SimulationMethods
```

*Values:*

```
enumerator WCSPH = 0
enumerator PCISPH
enumerator PBF
enumerator IISPH
enumerator DFSPH
enumerator PF
enumerator NumSimulationMethods
```

## Enum SurfaceSamplingMode

- Defined in file\_SPlisHSPlasH\_Uutilities\_SurfaceSampling.h

## Enum Documentation

```
enum SPH::SurfaceSamplingMode
```

*Values:*

```
enumerator PoissonDisk
enumerator RegularTriangle
enumerator Regular2D
```

## Enum SurfaceTensionMethods

- Defined in file\_SPlisHSPlasH\_FluidModel.h

## Enum Documentation

```
enum SPH::SurfaceTensionMethods
```

*Values:*

```
enumerator None = 0
enumerator Becker2007
enumerator Akinci2013
enumerator He2014
enumerator NumSurfaceTensionMethods
```

### Enum ViscosityMethods

- Defined in file\_SPlisHSPlasH\_FluidModel.h

### Enum Documentation

**enum** SPH::ViscosityMethods

*Values:*

```
enumerator None = 0
enumerator Standard
enumerator XSPH
enumerator Bender2017
enumerator Peer2015
enumerator Peer2016
enumerator Takahashi2015
enumerator Weiler2018
enumerator NumViscosityMethods
```

### Enum VorticityMethods

- Defined in file\_SPlisHSPlasH\_FluidModel.h

### Enum Documentation

**enum** SPH::VorticityMethods

*Values:*

```
enumerator None = 0
enumerator Micropolar
enumerator VorticityConfinement
enumerator NumVorticityMethods
```

## 6.3.4 Functions

### Function abs

- Defined in file\_SPlisHSPlasH\_Uutilities\_AVX\_math.h

## Function Documentation

*Scalarf8* **abs** (*Scalarf8* **const** &a)

## Function blend

- Defined in file\_SPlisHSPlasH\_Uilities\_AVX\_math.h

## Function Documentation

**Warning:** doxygenfunction: Unable to resolve multiple matches for function “blend” with arguments (Scalarf8 const&, Scalarf8 const&, Scalarf8 const&) in doxygen xml output for project “SPlisHSPlasH” from directory: ./doxyoutput/xml. Potential matches:

```
- Scalarf8 blend(Scalarf8 const &c, Scalarf8 const &a, Scalarf8 const &b)
- Vector3f8 blend(Scalarf8 const &c, Vector3f8 const &a, Vector3f8 const &b)
```

## Template Function constant8f

- Defined in file\_SPlisHSPlasH\_Uilities\_AVX\_math.h

## Function Documentation

```
template<int i0, int i1, int i2, int i3, int i4, int i5, int i6, int i7>
__m256 constant8f ()
```

## Function convert\_one

- Defined in file\_SPlisHSPlasH\_Uilities\_AVX\_math.h

## Function Documentation

*Scalarf8* **convert\_one** (**const** unsigned int \*idx, **const** *Real* \*x, **const** unsigned char count = 8u)

## Function convert\_zero(const unsigned int \*, const Real \*, const unsigned char)

- Defined in file\_SPlisHSPlasH\_Uilities\_AVX\_math.h

## Function Documentation

**Warning:** doxygenfunction: Unable to resolve multiple matches for function “convert\_zero” with arguments (const unsigned int \*, const Real \*, const unsigned char) in doxygen xml output for project “SPlisHSPlasH” from directory: ./doxyoutput/xml. Potential matches:

```
- Scalarf8 convert_zero(const Real x, const unsigned char count = 8u)
- Scalarf8 convert_zero(const unsigned int *idx, const Real *x, const unsigned char_
↳count = 8u)
```

### Function convert\_zero(const Real, const unsigned char)

- Defined in file\_SPlisHSPlasH\_Uilities\_AVX\_math.h

## Function Documentation

**Warning:** doxygenfunction: Unable to resolve multiple matches for function “convert\_zero” with arguments (const Real, const unsigned char) in doxygen xml output for project “SPlisHSPlasH” from directory: ./doxyoutput/xml. Potential matches:

```
- Scalarf8 convert_zero(const Real x, const unsigned char count = 8u)
- Scalarf8 convert_zero(const unsigned int *idx, const Real *x, const unsigned char_
↳count = 8u)
```

### Function convertVec\_zero(const unsigned int \*, const Real \*, const unsigned char)

- Defined in file\_SPlisHSPlasH\_Uilities\_AVX\_math.h

## Function Documentation

**Warning:** doxygenfunction: Unable to resolve multiple matches for function “convertVec\_zero” with arguments (const unsigned int \*, const Real \*, const unsigned char) in doxygen xml output for project “SPlisHSPlasH” from directory: ./doxyoutput/xml. Potential matches:

```
- Vector3f8 convertVec_zero(const unsigned int *idx, const Real *v, const unsigned_
↳char count = 8u)
- Vector3f8 convertVec_zero(const unsigned int *idx, const Vector3r *v, const_
↳unsigned char count = 8u)
```



**Function convertVec\_zero(const unsigned int \*, const Vector3r \*, const unsigned char)**

- Defined in file\_SPlisHSPlasH\_Uilities\_AVX\_math.h

**Function Documentation**

**Warning:** doxygenfunction: Unable to resolve multiple matches for function “convertVec\_zero” with arguments (const unsigned int \*, const Vector3r \*, const unsigned char) in doxygen xml output for project “SPlisHSPlasH” from directory: ./doxyoutput/xml. Potential matches:

```
- Vector3f8 convertVec_zero(const unsigned int *idx, const Real *v, const unsigned_
↳char count = 8u)
- Vector3f8 convertVec_zero(const unsigned int *idx, const Vector3r *v, const_
↳unsigned char count = 8u)
```

**Function dyadicProduct**

- Defined in file\_SPlisHSPlasH\_Uilities\_AVX\_math.h

**Function Documentation**

void **dyadicProduct** (const *Vector3f8* &a, const *Vector3f8* &b, *Matrix3f8* &res)

**Function getTime**

- Defined in file\_SPlisHSPlasH\_AnimationField.cpp

**Function Documentation**

*Real* SPH::TimeManager::getTime ()

**Function max**

- Defined in file\_SPlisHSPlasH\_Uilities\_AVX\_math.h

**Function Documentation**

*Scalarf8* **max** (*Scalarf8* const &a, *Scalarf8* const &b)

**Function operator!=**

- Defined in file\_SPlisHSPlasH\_Uilities\_AVX\_math.h

**Function Documentation**

*Scalarf8* **operator!=** (*Scalarf8* const &a, *Scalarf8* const &b)

**Function operator\*(Scalarf8 const&, Scalarf8 const&)**

- Defined in file\_SPlisHSPlasH\_Uilities\_AVX\_math.h

**Function Documentation**

**Warning:** doxygenfunction: Unable to resolve multiple matches for function “operator\*” with arguments (Scalarf8 const&, Scalarf8 const&) in doxygen xml output for project “SPlisHSPlasH” from directory: ./doxyout/xml. Potential matches:

```
- Matrix3f8 operator*(const Matrix3f8 &b) const
- Matrix3f8 operator*(const Scalarf8 &b) const
- Scalarf8 operator*(Scalarf8 const &a, Scalarf8 const &b)
- Scalarf8 operator*(const Vector3f8 &a) const
- Vector3f8 operator*(Vector3f8 const &a, const Scalarf8 &s)
- Vector3f8 operator*(const Vector3f8 &b) const
- const Quaternion8f operator*(const Quaternion8f &a) const
- template<typename Rhs> Eigen::Product<MatrixReplacement, Rhs, _
↳Eigen::AliasFreeProduct> operator*(const Eigen::MatrixBase<Rhs> &x) const
```

**Function operator\*(Vector3f8 const&, const Scalarf8&)**

- Defined in file\_SPlisHSPlasH\_Uilities\_AVX\_math.h

**Function Documentation**

**Warning:** doxygenfunction: Unable to resolve multiple matches for function “operator\*” with arguments (Vector3f8 const&, const Scalarf8&) in doxygen xml output for project “SPlisHSPlasH” from directory: ./doxyout/xml. Potential matches:

```
- Matrix3f8 operator*(const Matrix3f8 &b) const
- Matrix3f8 operator*(const Scalarf8 &b) const
- Scalarf8 operator*(Scalarf8 const &a, Scalarf8 const &b)
- Scalarf8 operator*(const Vector3f8 &a) const
- Vector3f8 operator*(Vector3f8 const &a, const Scalarf8 &s)
- Vector3f8 operator*(const Vector3f8 &b) const
- const Quaternion8f operator*(const Quaternion8f &a) const
- template<typename Rhs> Eigen::Product<MatrixReplacement, Rhs, _
↳Eigen::AliasFreeProduct> operator*(const Eigen::MatrixBase<Rhs> &x) const
```

## Function operator\*=-

- Defined in file\_SPlisHSPlasH\_Uilities\_AVX\_math.h

## Function Documentation

**Warning:** doxygenfunction: Unable to resolve multiple matches for function “operator\*=” with arguments (Scalarf8&, Scalarf8 const&) in doxygen xml output for project “SPlisHSPlasH” from directory: ./doxyoutput/xml. Potential matches:

```
- Scalarf8 &operator*=(Scalarf8 &a, Scalarf8 const &b)
- Vector3f8 &operator*=(const Scalarf8 &s)
```

## Function operator+(Scalarf8 const&, Scalarf8 const&)

- Defined in file\_SPlisHSPlasH\_Uilities\_AVX\_math.h

## Function Documentation

**Warning:** doxygenfunction: Unable to resolve multiple matches for function “operator+” with arguments (Scalarf8 const&, Scalarf8 const&) in doxygen xml output for project “SPlisHSPlasH” from directory: ./doxyoutput/xml. Potential matches:

```
- Scalarf8 operator+(Scalarf8 const &a, Scalarf8 const &b)
- Vector3f8 operator+(Vector3f8 const &a, Vector3f8 const &b)
```

## Function operator+(Vector3f8 const&, Vector3f8 const&)

- Defined in file\_SPlisHSPlasH\_Uilities\_AVX\_math.h

## Function Documentation

**Warning:** doxygenfunction: Unable to resolve multiple matches for function “operator+” with arguments (Vector3f8 const&, Vector3f8 const&) in doxygen xml output for project “SPlisHSPlasH” from directory: ./doxyoutput/xml. Potential matches:

```
- Scalarf8 operator+(Scalarf8 const &a, Scalarf8 const &b)
- Vector3f8 operator+(Vector3f8 const &a, Vector3f8 const &b)
```

### Function operator+=(Scalarf8&, Scalarf8 const&)

- Defined in file\_SPlisHSPlasH\_Uilities\_AVX\_math.h

#### Function Documentation

**Warning:** doxygenfunction: Unable to resolve multiple matches for function “operator+=” with arguments (Scalarf8&, Scalarf8 const&) in doxygen xml output for project “SPlisHSPlasH” from directory: ./doxyoutput/xml. Potential matches:

```
- Matrix3f8 &operator+=(const Matrix3f8 &a)
- Scalarf8 &operator+=(Scalarf8 &a, Scalarf8 const &b)
- Vector3f8 &operator+=(Vector3f8 &a, Vector3f8 const &b)
```

### Function operator+=(Vector3f8&, Vector3f8 const&)

- Defined in file\_SPlisHSPlasH\_Uilities\_AVX\_math.h

#### Function Documentation

**Warning:** doxygenfunction: Unable to resolve multiple matches for function “operator+=” with arguments (Vector3f8&, Vector3f8 const&) in doxygen xml output for project “SPlisHSPlasH” from directory: ./doxyoutput/xml. Potential matches:

```
- Matrix3f8 &operator+=(const Matrix3f8 &a)
- Scalarf8 &operator+=(Scalarf8 &a, Scalarf8 const &b)
- Vector3f8 &operator+=(Vector3f8 &a, Vector3f8 const &b)
```

### Function operator-(Scalarf8&)

- Defined in file\_SPlisHSPlasH\_Uilities\_AVX\_math.h

#### Function Documentation

**Warning:** doxygenfunction: Unable to resolve multiple matches for function “operator-” with arguments (Scalarf8&) in doxygen xml output for project “SPlisHSPlasH” from directory: ./doxyoutput/xml. Potential matches:

```
- Scalarf8 operator-(Scalarf8 &a)
- Scalarf8 operator-(Scalarf8 const &a, Scalarf8 const &b)
- Vector3f8 operator-(Vector3f8 const &a, Vector3f8 const &b)
- const Vector3f8 operator-() const
```

### Function operator-(Scalarf8 const&, Scalarf8 const&)

- Defined in file\_SPlisHSPlasH\_Uilities\_AVX\_math.h

#### Function Documentation

**Warning:** doxygenfunction: Unable to resolve multiple matches for function “operator-” with arguments (Scalarf8 const&, Scalarf8 const&) in doxygen xml output for project “SPlisHSPlasH” from directory: ./doxyoutput/xml. Potential matches:

```
- Scalarf8 operator-(Scalarf8 &a)
- Scalarf8 operator-(Scalarf8 const &a, Scalarf8 const &b)
- Vector3f8 operator-(Vector3f8 const &a, Vector3f8 const &b)
- const Vector3f8 operator-() const
```

### Function operator-(Vector3f8 const&, Vector3f8 const&)

- Defined in file\_SPlisHSPlasH\_Uilities\_AVX\_math.h

#### Function Documentation

**Warning:** doxygenfunction: Unable to resolve multiple matches for function “operator-” with arguments (Vector3f8 const&, Vector3f8 const&) in doxygen xml output for project “SPlisHSPlasH” from directory: ./doxyoutput/xml. Potential matches:

```
- Scalarf8 operator-(Scalarf8 &a)
- Scalarf8 operator-(Scalarf8 const &a, Scalarf8 const &b)
- Vector3f8 operator-(Vector3f8 const &a, Vector3f8 const &b)
- const Vector3f8 operator-() const
```

### Function operator-=

- Defined in file\_SPlisHSPlasH\_Uilities\_AVX\_math.h

#### Function Documentation

**Warning:** doxygenfunction: Unable to resolve multiple matches for function “operator-=” with arguments (Scalarf8&, Scalarf8 const&) in doxygen xml output for project “SPlisHSPlasH” from directory: ./doxyoutput/xml. Potential matches:

```
- Scalarf8 &operator-=(Scalarf8 &a, Scalarf8 const &b)
- Vector3f8 &operator-=(const Vector3f8 &a)
```

### Function operator/

- Defined in file\_SPlisHSPlasH\_Uilities\_AVX\_math.h

### Function Documentation

**Warning:** doxygenfunction: Unable to resolve multiple matches for function “operator/” with arguments (Scalarf8 const&, Scalarf8 const&) in doxygen xml output for project “SPlisHSPlasH” from directory: ./doxyoutput/xml. Potential matches:

```
- Scalarf8 operator/(Scalarf8 const &a, Scalarf8 const &b)
- const Vector3f8 operator/(const Scalarf8 &s) const
```

### Function operator/=

- Defined in file\_SPlisHSPlasH\_Uilities\_AVX\_math.h

### Function Documentation

**Warning:** doxygenfunction: Unable to resolve multiple matches for function “operator/=” with arguments (Scalarf8&, Scalarf8 const&) in doxygen xml output for project “SPlisHSPlasH” from directory: ./doxyoutput/xml. Potential matches:

```
- Scalarf8 &operator/=(Scalarf8 &a, Scalarf8 const &b)
- Vector3f8 &operator/=(const Scalarf8 &s)
```

### Function operator<

- Defined in file\_SPlisHSPlasH\_Uilities\_AVX\_math.h

### Function Documentation

*Scalarf8* operator< (*Scalarf8* const &a, *Scalarf8* const &b)

### Function operator<=

- Defined in file\_SPlisHSPlasH\_Uilities\_AVX\_math.h

### Function Documentation

*Scalarf8* **operator<=** (*Scalarf8* **const** &*a*, *Scalarf8* **const** &*b*)

### Function operator==

- Defined in file\_SPlisHSPlasH\_Uilities\_AVX\_math.h

### Function Documentation

*Scalarf8* **operator==** (*Scalarf8* **const** &*a*, *Scalarf8* **const** &*b*)

### Function operator>

- Defined in file\_SPlisHSPlasH\_Uilities\_AVX\_math.h

### Function Documentation

*Scalarf8* **operator>** (*Scalarf8* **const** &*a*, *Scalarf8* **const** &*b*)

### Function operator>=

- Defined in file\_SPlisHSPlasH\_Uilities\_AVX\_math.h

### Function Documentation

*Scalarf8* **operator>=** (*Scalarf8* **const** &*a*, *Scalarf8* **const** &*b*)

## 6.3.5 Variables

### Variable SPH::gaussian\_abscissae\_1

- Defined in file\_SPlisHSPlasH\_Uilities\_GaussQuadrature.cpp

### Variable Documentation

double **const** SPH::gaussian\_abscissae\_1[101][51]

### Variable SPH::gaussian\_n\_1

- Defined in file\_SPlisHSPlasH\_Uutilities\_GaussQuadrature.cpp

### Variable Documentation

unsigned int **const** SPH::gaussian\_n\_1[101]

### Variable SPH::gaussian\_weights\_1

- Defined in file\_SPlisHSPlasH\_Uutilities\_GaussQuadrature.cpp

### Variable Documentation

double **const** SPH::gaussian\_weights\_1[101][51]

## 6.3.6 Defines

### Define \_USE\_MATH\_DEFINES

- Defined in file\_SPlisHSPlasH\_Drag\_DragForce\_Gissler2017.cpp

### Define Documentation

**\_USE\_MATH\_DEFINES**

### Define \_USE\_MATH\_DEFINES

- Defined in file\_SPlisHSPlasH\_SPHKernels.h

### Define Documentation

**\_USE\_MATH\_DEFINES**

### Define \_USE\_MATH\_DEFINES

- Defined in file\_SPlisHSPlasH\_Uutilities\_PoissonDiskSampling.cpp



### Define Documentation

**`_USE_MATH_DEFINES`**

### Define `_USE_MATH_DEFINES`

- Defined in file\_SPlisHSPlasH\_Uutilities\_WindingNumbers.cpp

### Define Documentation

**`_USE_MATH_DEFINES`**

### Define `compute_Vj`

- Defined in file\_SPlisHSPlasH\_FluidModel.h

### Define Documentation

**`compute_Vj`** (*fm\_neighbor*)

### Define `compute_Vj_gradW`

- Defined in file\_SPlisHSPlasH\_FluidModel.h

### Define Documentation

**`compute_Vj_gradW`** ()

### Define `compute_Vj_gradW_samephase`

- Defined in file\_SPlisHSPlasH\_FluidModel.h

### Define Documentation

**`compute_Vj_gradW_samephase`** ()

### Define `compute_xj`

- Defined in file\_SPlisHSPlasH\_FluidModel.h

## Define Documentation

**compute\_xj** (*fm\_neighbor, pid*)

## Define forall\_boundary\_neighbors

- Defined in file\_SPlisHSPlasH\_Simulation.h

## Define Documentation

**forall\_boundary\_neighbors** (*code*)

Loop over the boundary neighbors of all fluid phases. Simulation \*sim and unsigned int fluidModelIndex must be defined.

## Define forall\_density\_maps

- Defined in file\_SPlisHSPlasH\_Simulation.h

## Define Documentation

**forall\_density\_maps** (*code*)

Loop over the boundary density maps. Simulation \*sim, unsigned int nBoundaries and unsigned int fluidModelIndex must be defined.

## Define forall\_fluid\_neighbors

- Defined in file\_SPlisHSPlasH\_Simulation.h

## Define Documentation

**forall\_fluid\_neighbors** (*code*)

Loop over the fluid neighbors of all fluid phases. Simulation \*sim and unsigned int fluidModelIndex must be defined.

## Define forall\_fluid\_neighbors\_in\_same\_phase

- Defined in file\_SPlisHSPlasH\_Simulation.h

### Define Documentation

#### **forall\_fluid\_neighbors\_in\_same\_phase** (*code*)

Loop over the fluid neighbors of the same fluid phase. Simulation *sim*, *unsigned int fluidModelIndex* and *FluidModel* model must be defined.

### Define forall\_volume\_maps

- Defined in file\_SPlisHSPlasH\_Simulation.h

### Define Documentation

#### **forall\_volume\_maps** (*code*)

Loop over the boundary volume maps. Simulation \**sim*, *unsigned int nBoundaries* and *unsigned int fluidModelIndex* must be defined.

### Define FORCE\_INLINE

- Defined in file\_SPlisHSPlasH\_Common.h

### Define Documentation

#### **FORCE\_INLINE**

### Define PD\_USE\_DIAGONAL\_PRECONDITIONER

- Defined in file\_SPlisHSPlasH\_PF\_TimeStepPF.h

### Define Documentation

#### **PD\_USE\_DIAGONAL\_PRECONDITIONER**

### Define REAL\_MAX

- Defined in file\_SPlisHSPlasH\_Common.h

### Define Documentation

#### **REAL\_MAX**

## **Define REAL\_MIN**

- Defined in file\_SPlisHSPlasH\_Common.h

## **Define Documentation**

**REAL\_MIN**

## **Define RealParameter**

- Defined in file\_SPlisHSPlasH\_Common.h

## **Define Documentation**

**RealParameter**

## **Define RealParameterType**

- Defined in file\_SPlisHSPlasH\_Common.h

## **Define Documentation**

**RealParameterType**

## **Define RealVectorParameter**

- Defined in file\_SPlisHSPlasH\_Common.h

## **Define Documentation**

**RealVectorParameter**

## **Define RealVectorParameterType**

- Defined in file\_SPlisHSPlasH\_Common.h

## **Define Documentation**

**RealVectorParameterType**

**Define REPORT\_MEMORY\_LEAKS**

- Defined in file\_SPlisHSPlasH\_Common.h

**Define Documentation****REPORT\_MEMORY\_LEAKS****Define USE\_BLOCKDIAGONAL\_PRECONDITIONER**

- Defined in file\_SPlisHSPlasH\_Viscosity\_Viscosity\_Weiler2018.h

**Define Documentation****USE\_BLOCKDIAGONAL\_PRECONDITIONER****Define USE\_WARMSTART**

- Defined in file\_SPlisHSPlasH\_DFSPH\_TimeStepDFSPH.h

**Define Documentation****USE\_WARMSTART****Define USE\_WARMSTART\_V**

- Defined in file\_SPlisHSPlasH\_DFSPH\_TimeStepDFSPH.h

**Define Documentation****USE\_WARMSTART\_V****Define Vec3Block**

- Defined in file\_SPlisHSPlasH\_PF\_TimeStepPF.cpp

**Define Documentation**

**Warning:** doxygendefine: Cannot find define “Vec3Block” in doxygen xml output for project “SPlisHSPlasH” from directory: ./doxyoutput/xml

**6.3.7 Typedefs**

### Typedef AlignedBox2r

- Defined in file\_SPlisHSPlasH\_Common.h

### Typedef Documentation

```
using AlignedBox2r = Eigen::AlignedBox<Real, 2>
```

### Typedef AlignedBox3r

- Defined in file\_SPlisHSPlasH\_Common.h

### Typedef Documentation

```
using AlignedBox3r = Eigen::AlignedBox<Real, 3>
```

### Typedef AngleAxisr

- Defined in file\_SPlisHSPlasH\_Common.h

### Typedef Documentation

```
using AngleAxisr = Eigen::AngleAxis<Real>
```

### Typedef AtomicRealVec

- Defined in file\_SPlisHSPlasH\_PF\_TimeStepPF.cpp

### Typedef Documentation

**Warning:** doxygentypedef: Cannot find typedef “AtomicRealVec” in doxygen xml output for project “SPlisH-SPlasH” from directory: ./doxyoutput/xml

### Typedef Matrix2r

- Defined in file\_SPlisHSPlasH\_Common.h

### Typedef Documentation

**using Matrix2r** = Eigen::Matrix<*Real*, 2, 2, Eigen::DontAlign>

### Typedef Matrix3f

- Defined in file\_SPlisHSPlasH\_Common.h

### Typedef Documentation

**using Matrix3f** = Eigen::Matrix<float, 3, 3, Eigen::DontAlign>

### Typedef Matrix3r

- Defined in file\_SPlisHSPlasH\_Common.h

### Typedef Documentation

**using Matrix3r** = Eigen::Matrix<*Real*, 3, 3, Eigen::DontAlign>

### Typedef Matrix4r

- Defined in file\_SPlisHSPlasH\_Common.h

### Typedef Documentation

**using Matrix4r** = Eigen::Matrix<*Real*, 4, 4, Eigen::DontAlign>

### Typedef Matrix5r

- Defined in file\_SPlisHSPlasH\_Common.h

### Typedef Documentation

**using Matrix5r** = Eigen::Matrix<*Real*, 5, 5, Eigen::DontAlign>

### Typedef Matrix6r

- Defined in file\_SPlisHSPlasH\_Common.h

### Typedef Documentation

**using** **Matrix6r** = Eigen::Matrix<*Real*, 6, 6, Eigen::DontAlign>

### Typedef NeighborhoodSearch

- Defined in file\_SPlisHSPlasH\_NeighborhoodSearch.h

### Typedef Documentation

**typedef** CompactNSearch::NeighborhoodSearch **NeighborhoodSearch**

### Typedef Quaternionr

- Defined in file\_SPlisHSPlasH\_Common.h

### Typedef Documentation

**using** **Quaternionr** = Eigen::Quaternion<*Real*, Eigen::DontAlign>

### Typedef Real

- Defined in file\_SPlisHSPlasH\_Common.h

### Typedef Documentation

**typedef** float **Real**

### Typedef SystemMatrixType

- Defined in file\_SPlisHSPlasH\_Utilities\_MatrixFreeSolver.h

### Typedef Documentation

**using** **SystemMatrixType** = Eigen::SparseMatrix<*Real*>

### Typedef Vector2i

- Defined in file\_SPlisHSPlasH\_Common.h



### Typedef Documentation

**using Vector2i** = Eigen::Matrix<int, 2, 1, Eigen::DontAlign>

### Typedef Vector2r

- Defined in file\_SPlisHSPlasH\_Common.h

### Typedef Documentation

**using Vector2r** = Eigen::Matrix<*Real*, 2, 1, Eigen::DontAlign>

### Typedef Vector3f

- Defined in file\_SPlisHSPlasH\_Common.h

### Typedef Documentation

**using Vector3f** = Eigen::Matrix<float, 3, 1, Eigen::DontAlign>

### Typedef Vector3r

- Defined in file\_SPlisHSPlasH\_Common.h

### Typedef Documentation

**using Vector3r** = Eigen::Matrix<*Real*, 3, 1, Eigen::DontAlign>

### Typedef Vector4f

- Defined in file\_SPlisHSPlasH\_Common.h

### Typedef Documentation

**using Vector4f** = Eigen::Matrix<float, 4, 1, Eigen::DontAlign>

### Typedef Vector4r

- Defined in file\_SPlisHSPlasH\_Common.h

### Typedef Documentation

**using** **Vector4r** = Eigen::Matrix<*Real*, 4, 1, Eigen::DontAlign>

### Typedef Vector5r

- Defined in file\_SPlisHSPlasH\_Common.h

### Typedef Documentation

**using** **Vector5r** = Eigen::Matrix<*Real*, 5, 1, Eigen::DontAlign>

### Typedef Vector6r

- Defined in file\_SPlisHSPlasH\_Common.h

### Typedef Documentation

**using** **Vector6r** = Eigen::Matrix<*Real*, 6, 1, Eigen::DontAlign>

### Typedef VectorXr

- Defined in file\_SPlisHSPlasH\_Common.h

### Typedef Documentation

**using** **SPH::TimeStepPF::VectorXr** = Eigen::Matrix<*Real*, -1, 1>

## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`



## Symbols

`_USE_MATH_DEFINES` (*C macro*), 140, 141

## A

`abs` (*C++ function*), 131  
`AlignedBox2r` (*C++ type*), 146  
`AlignedBox3r` (*C++ type*), 146  
`AngleAxisr` (*C++ type*), 146

## C

`compute_Vj` (*C macro*), 141  
`compute_Vj_gradW` (*C macro*), 141  
`compute_Vj_gradW_samephase` (*C macro*), 141  
`compute_xj` (*C macro*), 142  
`constant8f` (*C++ function*), 131  
`convert_one` (*C++ function*), 131

## D

`dyadicProduct` (*C++ function*), 133

## E

`Eigen::internal::generic_product_impl<MatrixReplacement,  
 Rhs, SparseShape, DenseShape,  
 GemvProduct>` (*C++ struct*), 30  
`Eigen::internal::generic_product_impl<MatrixReplacement,  
 Rhs, SparseShape, DenseShape,  
 GemvProduct>::Scalar` (*C++ type*), 30  
`Eigen::internal::generic_product_impl<MatrixReplacement,  
 Rhs, SparseShape, DenseShape,  
 GemvProduct>::scaleAndAddTo` (*C++  
 function*), 30  
`Eigen::internal::traits<SPH::MatrixReplacement>`  
 (*C++ struct*), 30

## F

`forall_boundary_neighbors` (*C macro*), 142  
`forall_density_maps` (*C macro*), 142  
`forall_fluid_neighbors` (*C macro*), 142  
`forall_fluid_neighbors_in_same_phase` (*C  
 macro*), 143  
`forall_volume_maps` (*C macro*), 143

`FORCE_INLINE` (*C macro*), 143

## M

`Matrix2r` (*C++ type*), 147  
`Matrix3f` (*C++ type*), 147  
`Matrix3f8` (*C++ class*), 38  
`Matrix3f8::determinant` (*C++ function*), 38  
`Matrix3f8::m` (*C++ member*), 38  
`Matrix3f8::Matrix3f8` (*C++ function*), 38  
`Matrix3f8::operator()` (*C++ function*), 38  
`Matrix3f8::operator*` (*C++ function*), 38  
`Matrix3f8::operator+=` (*C++ function*), 38  
`Matrix3f8::reduce` (*C++ function*), 38  
`Matrix3f8::setCol` (*C++ function*), 38  
`Matrix3f8::setZero` (*C++ function*), 38  
`Matrix3f8::store` (*C++ function*), 38  
`Matrix3f8::transpose` (*C++ function*), 38  
`Matrix3r` (*C++ type*), 147  
`Matrix4r` (*C++ type*), 147  
`Matrix5r` (*C++ type*), 147  
`Matrix6r` (*C++ type*), 148  
`max` (*C++ function*), 133

## N

`NeighborhoodSearch` (*C++ type*), 148

## O

`operator!=` (*C++ function*), 134  
`operator==` (*C++ function*), 139  
`operator>` (*C++ function*), 139  
`operator>=` (*C++ function*), 139  
`operator<` (*C++ function*), 138  
`operator<=` (*C++ function*), 139

## P

`PD_USE_DIAGONAL_PRECONDITIONER` (*C macro*),  
 143

## Q

`Quaternion8f` (*C++ class*), 38  
`Quaternion8f::operator*` (*C++ function*), 39  
`Quaternion8f::operator[]` (*C++ function*), 39

Quaternion8f::q (C++ *member*), 39  
 Quaternion8f::Quaternion8f (C++ *function*), 39  
 Quaternion8f::set (C++ *function*), 39  
 Quaternion8f::store (C++ *function*), 39  
 Quaternion8f::toRotationMatrix (C++ *function*), 39  
 Quaternion8f::w (C++ *function*), 39  
 Quaternion8f::x (C++ *function*), 39  
 Quaternion8f::y (C++ *function*), 39  
 Quaternion8f::z (C++ *function*), 39  
 Quaternionr (C++ *type*), 148

## R

Real (C++ *type*), 148  
 REAL\_MAX (C *macro*), 143  
 REAL\_MIN (C *macro*), 144  
 RealParameter (C *macro*), 144  
 RealParameterType (C *macro*), 144  
 RealVectorParameter (C *macro*), 144  
 RealVectorParameterType (C *macro*), 144  
 REPORT\_MEMORY\_LEAKS (C *macro*), 145

## S

Scalarf8 (C++ *class*), 39  
 Scalarf8::load (C++ *function*), 40  
 Scalarf8::operator= (C++ *function*), 40  
 Scalarf8::reduce (C++ *function*), 40  
 Scalarf8::rsqrt (C++ *function*), 40  
 Scalarf8::Scalarf8 (C++ *function*), 40  
 Scalarf8::sqrt (C++ *function*), 40  
 Scalarf8::store (C++ *function*), 40  
 Scalarf8::v (C++ *member*), 40  
 SPH::AdhesionKernel (C++ *class*), 40  
 SPH::AdhesionKernel::getRadius (C++ *function*), 40  
 SPH::AdhesionKernel::m\_k (C++ *member*), 41  
 SPH::AdhesionKernel::m\_radius (C++ *member*), 41  
 SPH::AdhesionKernel::m\_W\_zero (C++ *member*), 41  
 SPH::AdhesionKernel::setRadius (C++ *function*), 40  
 SPH::AdhesionKernel::W (C++ *function*), 40  
 SPH::AdhesionKernel::W\_zero (C++ *function*), 40  
 SPH::AnimationField (C++ *class*), 41  
 SPH::AnimationField::~~AnimationField (C++ *function*), 41  
 SPH::AnimationField::AnimationField (C++ *function*), 41  
 SPH::AnimationField::m\_endTime (C++ *member*), 41

SPH::AnimationField::m\_expression (C++ *member*), 41  
 SPH::AnimationField::m\_particleFieldName (C++ *member*), 41  
 SPH::AnimationField::m\_rotation (C++ *member*), 41  
 SPH::AnimationField::m\_scale (C++ *member*), 41  
 SPH::AnimationField::m\_startTime (C++ *member*), 41  
 SPH::AnimationField::m\_type (C++ *member*), 41  
 SPH::AnimationField::m\_x (C++ *member*), 41  
 SPH::AnimationField::reset (C++ *function*), 41  
 SPH::AnimationField::setEndTime (C++ *function*), 41  
 SPH::AnimationField::setStartTime (C++ *function*), 41  
 SPH::AnimationField::step (C++ *function*), 41  
 SPH::AnimationFieldSystem (C++ *class*), 42  
 SPH::AnimationFieldSystem::~~AnimationFieldSystem (C++ *function*), 42  
 SPH::AnimationFieldSystem::addAnimationField (C++ *function*), 42  
 SPH::AnimationFieldSystem::AnimationFieldSystem (C++ *function*), 42  
 SPH::AnimationFieldSystem::getAnimationFields (C++ *function*), 42  
 SPH::AnimationFieldSystem::m\_fields (C++ *member*), 42  
 SPH::AnimationFieldSystem::numAnimationFields (C++ *function*), 42  
 SPH::AnimationFieldSystem::reset (C++ *function*), 42  
 SPH::AnimationFieldSystem::step (C++ *function*), 42  
 SPH::BlockJacobiPreconditioner3D (C++ *class*), 42  
 SPH::BlockJacobiPreconditioner3D::\_solve\_impl (C++ *function*), 43  
 SPH::BlockJacobiPreconditioner3D::analyzePattern (C++ *function*), 43  
 SPH::BlockJacobiPreconditioner3D::BlockJacobiPreconditioner3D (C++ *function*), 43  
 SPH::BlockJacobiPreconditioner3D::cols (C++ *function*), 43  
 SPH::BlockJacobiPreconditioner3D::compute (C++ *function*), 43  
 SPH::BlockJacobiPreconditioner3D::DiagonalMatrixElement (C++ *type*), 42  
 SPH::BlockJacobiPreconditioner3D::factorize (C++ *function*), 43  
 SPH::BlockJacobiPreconditioner3D::info

(C++ function), 43

SPH::BlockJacobiPreconditioner3D::init (C++ function), 43

SPH::BlockJacobiPreconditioner3D::m\_diag (C++ member), 43

SPH::BlockJacobiPreconditioner3D::m\_dim (C++ member), 43

SPH::BlockJacobiPreconditioner3D::m\_invDiag (C++ member), 43

SPH::BlockJacobiPreconditioner3D::m\_userDiag (C++ member), 43

SPH::BlockJacobiPreconditioner3D::rows (C++ function), 43

SPH::BlockJacobiPreconditioner3D::solve (C++ function), 43

SPH::BlockJacobiPreconditioner3D::StorageType (C++ type), 42

SPH::BlockJacobiPreconditioner3D::[anonymous] (C++ enum), 42

SPH::BlockJacobiPreconditioner3D::[anonymous] (C++ enumerator), 42

SPH::BlockJacobiPreconditioner3D::[anonymous] (C++ enumerator), 42

SPH::BoundaryHandlingMethods (C++ enum), 127

SPH::BoundaryHandlingMethods::Akinci2012 (C++ enumerator), 127

SPH::BoundaryHandlingMethods::Bender2019 (C++ enumerator), 127

SPH::BoundaryHandlingMethods::Koschier2017 (C++ enumerator), 127

SPH::BoundaryHandlingMethods::NumSimulations (C++ enumerator), 127

SPH::BoundaryModel (C++ class), 44

SPH::BoundaryModel::~~BoundaryModel (C++ function), 44

SPH::BoundaryModel::BoundaryModel (C++ function), 44

SPH::BoundaryModel::clearForceAndTorque (C++ function), 44

SPH::BoundaryModel::getForceAndTorque (C++ function), 44

SPH::BoundaryModel::getRigidBodyObject (C++ function), 44

SPH::BoundaryModel::loadState (C++ function), 44

SPH::BoundaryModel::m\_forcePerThread (C++ member), 44

SPH::BoundaryModel::m\_rigidBody (C++ member), 44

SPH::BoundaryModel::m\_torquePerThread (C++ member), 44

SPH::BoundaryModel::performNeighborhoodSearch (C++ function), 44

SPH::BoundaryModel::reset (C++ function), 44

SPH::BoundaryModel::saveState (C++ function), 44

SPH::BoundaryModel\_Akinci2012 (C++ class), 45

SPH::BoundaryModel\_Akinci2012::~~BoundaryModel\_Akinci2012 (C++ function), 45

SPH::BoundaryModel\_Akinci2012::BoundaryModel\_Akinci2012 (C++ function), 45

SPH::BoundaryModel\_Akinci2012::computeBoundaryVolume (C++ function), 45

SPH::BoundaryModel\_Akinci2012::getPointSetIndex (C++ function), 45

SPH::BoundaryModel\_Akinci2012::initModel (C++ function), 45

SPH::BoundaryModel\_Akinci2012::loadState (C++ function), 45

SPH::BoundaryModel\_Akinci2012::m\_pointSetIndex (C++ member), 46

SPH::BoundaryModel\_Akinci2012::m\_sorted (C++ member), 46

SPH::BoundaryModel\_Akinci2012::m\_V (C++ member), 46

SPH::BoundaryModel\_Akinci2012::m\_v (C++ member), 46

SPH::BoundaryModel\_Akinci2012::m\_x (C++ member), 46

SPH::BoundaryModel\_Akinci2012::m\_x0 (C++ member), 46

SPH::BoundaryModel\_Akinci2012::numberOfParticles (C++ function), 45

SPH::BoundaryModel\_Akinci2012::performNeighborhoodSearch (C++ function), 45

SPH::BoundaryModel\_Akinci2012::reset (C++ function), 45

SPH::BoundaryModel\_Akinci2012::resize (C++ function), 45

SPH::BoundaryModel\_Akinci2012::saveState (C++ function), 45

SPH::BoundaryModel\_Bender2019 (C++ class), 46

SPH::BoundaryModel\_Bender2019::~~BoundaryModel\_Bender2019 (C++ function), 46

SPH::BoundaryModel\_Bender2019::BoundaryModel\_Bender2019 (C++ function), 46

SPH::BoundaryModel\_Bender2019::getMap (C++ function), 46

SPH::BoundaryModel\_Bender2019::getMaxDist (C++ function), 46

SPH::BoundaryModel\_Bender2019::getMaxVel (C++ function), 46

SPH::BoundaryModel\_Bender2019::initModel (C++ function), 46

SPH::BoundaryModel\_Bender2019::m\_boundaryVolume

(C++ member), 47

SPH::BoundaryModel\_Bender2019::m\_boundaryXj (C++ member), 47

SPH::BoundaryModel\_Bender2019::m\_map (C++ member), 47

SPH::BoundaryModel\_Bender2019::m\_maxDist (C++ member), 47

SPH::BoundaryModel\_Bender2019::m\_maxVel (C++ member), 47

SPH::BoundaryModel\_Bender2019::reset (C++ function), 46

SPH::BoundaryModel\_Bender2019::setMap (C++ function), 46

SPH::BoundaryModel\_Bender2019::setMaxDist (C++ function), 46

SPH::BoundaryModel\_Bender2019::setMaxVel (C++ function), 46

SPH::BoundaryModel\_Koschier2017 (C++ class), 47

SPH::BoundaryModel\_Koschier2017::~~BoundaryModel\_Koschier2017 (C++ function), 47

SPH::BoundaryModel\_Koschier2017::BoundaryModel\_Koschier2017 (C++ function), 47

SPH::BoundaryModel\_Koschier2017::getMap (C++ function), 47

SPH::BoundaryModel\_Koschier2017::getMaxDist (C++ function), 47

SPH::BoundaryModel\_Koschier2017::getMaxVel (C++ function), 47

SPH::BoundaryModel\_Koschier2017::initModel (C++ function), 47

SPH::BoundaryModel\_Koschier2017::m\_boundaryDensity (C++ member), 48

SPH::BoundaryModel\_Koschier2017::m\_boundaryDensityGradient (C++ member), 48

SPH::BoundaryModel\_Koschier2017::m\_boundaryXj (C++ member), 48

SPH::BoundaryModel\_Koschier2017::m\_map (C++ member), 48

SPH::BoundaryModel\_Koschier2017::m\_maxDist (C++ member), 48

SPH::BoundaryModel\_Koschier2017::m\_maxVel (C++ member), 48

SPH::BoundaryModel\_Koschier2017::reset (C++ function), 47

SPH::BoundaryModel\_Koschier2017::setMap (C++ function), 47

SPH::BoundaryModel\_Koschier2017::setMaxDist (C++ function), 47

SPH::BoundaryModel\_Koschier2017::setMaxVel (C++ function), 47

SPH::CohesionKernel (C++ class), 48

SPH::CohesionKernel::getRadius (C++ function), 48

SPH::CohesionKernel::m\_c (C++ member), 49

SPH::CohesionKernel::m\_k (C++ member), 49

SPH::CohesionKernel::m\_radius (C++ member), 49

SPH::CohesionKernel::m\_W\_zero (C++ member), 49

SPH::CohesionKernel::setRadius (C++ function), 48

SPH::CohesionKernel::W (C++ function), 48

SPH::CohesionKernel::W\_zero (C++ function), 48

SPH::CubicKernel (C++ class), 49

SPH::CubicKernel2D (C++ class), 50

SPH::CubicKernel2D::getRadius (C++ function), 50

SPH::CubicKernel2D::gradW (C++ function), 50

SPH::CubicKernel2D::m\_k (C++ member), 50

SPH::CubicKernel2D::m\_l (C++ member), 50

SPH::CubicKernel2D::m\_radius (C++ member), 50

SPH::CubicKernel2D::m\_W\_zero (C++ member), 50

SPH::CubicKernel2D::setRadius (C++ function), 50

SPH::CubicKernel2D::W (C++ function), 50

SPH::CubicKernel2D::W\_zero (C++ function), 50

SPH::CubicKernel::getRadius (C++ function), 49

SPH::CubicKernel::gradW (C++ function), 49

SPH::CubicKernel::m\_k (C++ member), 49

SPH::CubicKernel::m\_l (C++ member), 49

SPH::CubicKernel::m\_radius (C++ member), 49

SPH::CubicKernel::m\_W\_zero (C++ member), 49

SPH::CubicKernel::setRadius (C++ function), 49

SPH::CubicKernel::W (C++ function), 49

SPH::CubicKernel::W\_zero (C++ function), 49

SPH::DragBase (C++ class), 51

SPH::DragBase::~~DragBase (C++ function), 51

SPH::DragBase::DRAG\_COEFFICIENT (C++ member), 51

SPH::DragBase::DragBase (C++ function), 51

SPH::DragBase::initParameters (C++ function), 51

SPH::DragBase::m\_dragCoefficient (C++ member), 51

SPH::DragForce\_Gissler2017 (C++ class), 51

SPH::DragForce\_Gissler2017::~~DragForce\_Gissler2017 (C++ function), 52

SPH::DragForce\_Gissler2017::C\_b (C++ member), 52



SPH::DragForce\_Gissler2017::C\_d (C++ member), 52

SPH::DragForce\_Gissler2017::C\_F (C++ member), 52

SPH::DragForce\_Gissler2017::C\_k (C++ member), 52

SPH::DragForce\_Gissler2017::DragForce\_Gissler2017 (C++ function), 52

SPH::DragForce\_Gissler2017::mu\_a (C++ member), 52

SPH::DragForce\_Gissler2017::mu\_l (C++ member), 52

SPH::DragForce\_Gissler2017::reset (C++ function), 52

SPH::DragForce\_Gissler2017::rho\_a (C++ member), 52

SPH::DragForce\_Gissler2017::sigma (C++ member), 52

SPH::DragForce\_Gissler2017::step (C++ function), 52

SPH::DragForce\_Macklin2014 (C++ class), 52

SPH::DragForce\_Macklin2014::~~DragForce\_Macklin2014 (C++ function), 52

SPH::DragForce\_Macklin2014::DragForce\_Macklin2014 (C++ function), 52

SPH::DragForce\_Macklin2014::reset (C++ function), 52

SPH::DragForce\_Macklin2014::step (C++ function), 52

SPH::DragMethods (C++ enum), 127

SPH::DragMethods::Gissler2017 (C++ enumerator), 127

SPH::DragMethods::Macklin2014 (C++ enumerator), 127

SPH::DragMethods::None (C++ enumerator), 127

SPH::DragMethods::NumDragMethods (C++ enumerator), 127

SPH::Elasticity\_Becker2009 (C++ class), 53

SPH::Elasticity\_Becker2009::~~Elasticity\_Becker2009 (C++ function), 53

SPH::Elasticity\_Becker2009::ALPHA (C++ member), 53

SPH::Elasticity\_Becker2009::computeForces (C++ function), 53

SPH::Elasticity\_Becker2009::computeRotations (C++ function), 53

SPH::Elasticity\_Becker2009::computeStress (C++ function), 53

SPH::Elasticity\_Becker2009::Elasticity\_Becker2009 (C++ function), 53

SPH::Elasticity\_Becker2009::initParameters (C++ function), 53

SPH::Elasticity\_Becker2009::initValues (C++ function), 53

SPH::Elasticity\_Becker2009::loadState (C++ function), 53

SPH::Elasticity\_Becker2009::m\_alpha (C++ member), 54

SPH::Elasticity\_Becker2009::m\_current\_to\_initial\_in (C++ member), 54

SPH::Elasticity\_Becker2009::m\_F (C++ member), 54

SPH::Elasticity\_Becker2009::m\_initial\_to\_current\_in (C++ member), 54

SPH::Elasticity\_Becker2009::m\_initialNeighbors (C++ member), 54

SPH::Elasticity\_Becker2009::m\_restVolumes (C++ member), 54

SPH::Elasticity\_Becker2009::m\_rotations (C++ member), 54

SPH::Elasticity\_Becker2009::m\_stress (C++ member), 54

SPH::Elasticity\_Becker2009::performNeighborhoodSearch (C++ function), 53

SPH::Elasticity\_Becker2009::reset (C++ function), 53

SPH::Elasticity\_Becker2009::saveState (C++ function), 53

SPH::Elasticity\_Becker2009::step (C++ function), 53

SPH::Elasticity\_Peer2018 (C++ class), 54

SPH::Elasticity\_Peer2018::~~Elasticity\_Peer2018 (C++ function), 54

SPH::Elasticity\_Peer2018::ALPHA (C++ member), 55

SPH::Elasticity\_Peer2018::computeMatrixL (C++ function), 55

SPH::Elasticity\_Peer2018::computeRHS (C++ function), 55

SPH::Elasticity\_Peer2018::computeRotations (C++ function), 55

SPH::Elasticity\_Peer2018::Elasticity\_Peer2018 (C++ function), 54

SPH::Elasticity\_Peer2018::initParameters (C++ function), 55

SPH::Elasticity\_Peer2018::initValues (C++ function), 55

SPH::Elasticity\_Peer2018::ITERATIONS (C++ member), 55

SPH::Elasticity\_Peer2018::loadState (C++ function), 54

SPH::Elasticity\_Peer2018::m\_alpha (C++ member), 55

SPH::Elasticity\_Peer2018::m\_current\_to\_initial\_inde (C++ member), 55

SPH::Elasticity\_Peer2018::m\_F (C++ member), 55

SPH::Elasticity\_Peer2018::m\_initial\_to\_current\_inde

(C++ member), 55

SPH::Elasticity\_Peer2018::m\_initialNeighbors (C++ member), 55

SPH::Elasticity\_Peer2018::m\_iterations (C++ member), 55

SPH::Elasticity\_Peer2018::m\_L (C++ member), 55

SPH::Elasticity\_Peer2018::m\_maxError (C++ member), 55

SPH::Elasticity\_Peer2018::m\_maxIter (C++ member), 55

SPH::Elasticity\_Peer2018::m\_restVolumes (C++ member), 55

SPH::Elasticity\_Peer2018::m\_RL (C++ member), 55

SPH::Elasticity\_Peer2018::m\_rotations (C++ member), 55

SPH::Elasticity\_Peer2018::m\_solver (C++ member), 55

SPH::Elasticity\_Peer2018::m\_stress (C++ member), 55

SPH::Elasticity\_Peer2018::matrixVecProd (C++ function), 55

SPH::Elasticity\_Peer2018::MAX\_ERROR (C++ member), 55

SPH::Elasticity\_Peer2018::MAX\_ITERATIONS (C++ member), 55

SPH::Elasticity\_Peer2018::performNeighborhoodSearch (C++ function), 54

SPH::Elasticity\_Peer2018::reset (C++ function), 54

SPH::Elasticity\_Peer2018::saveState (C++ function), 54

SPH::Elasticity\_Peer2018::Solver (C++ type), 55

SPH::Elasticity\_Peer2018::step (C++ function), 54

SPH::ElasticityBase (C++ class), 56

SPH::ElasticityBase::~ElasticityBase (C++ function), 56

SPH::ElasticityBase::ElasticityBase (C++ function), 56

SPH::ElasticityBase::initParameters (C++ function), 56

SPH::ElasticityBase::m\_poissonRatio (C++ member), 56

SPH::ElasticityBase::m\_youngsModulus (C++ member), 56

SPH::ElasticityBase::POISSON\_RATIO (C++ member), 56

SPH::ElasticityBase::YOUNGS\_MODULUS (C++ member), 56

SPH::ElasticityMethods (C++ enum), 128

SPH::ElasticityMethods::Becker2009 (C++ enumerator), 128

SPH::ElasticityMethods::None (C++ enumerator), 128

SPH::ElasticityMethods::NumElasticityMethods (C++ enumerator), 128

SPH::ElasticityMethods::Peer2018 (C++ enumerator), 128

SPH::Emitter (C++ class), 57

SPH::Emitter::~~Emitter (C++ function), 57

SPH::Emitter::emitParticles (C++ function), 57

SPH::Emitter::emitParticlesCircle (C++ function), 57

SPH::Emitter::Emitter (C++ function), 57

SPH::Emitter::getNextEmitTime (C++ function), 57

SPH::Emitter::getSize (C++ function), 57

SPH::Emitter::loadState (C++ function), 57

SPH::Emitter::m\_emitCounter (C++ member), 58

SPH::Emitter::m\_emitEndTime (C++ member), 58

SPH::Emitter::m\_emitStartTime (C++ member), 58

SPH::Emitter::m\_height (C++ member), 58

SPH::Emitter::m\_model (C++ member), 58

SPH::Emitter::m\_nextEmitTime (C++ member), 58

SPH::Emitter::m\_rotation (C++ member), 58

SPH::Emitter::m\_type (C++ member), 58

SPH::Emitter::m\_velocity (C++ member), 58

SPH::Emitter::m\_width (C++ member), 58

SPH::Emitter::m\_x (C++ member), 58

SPH::Emitter::reset (C++ function), 57

SPH::Emitter::saveState (C++ function), 57

SPH::Emitter::setEmitEndTime (C++ function), 57

SPH::Emitter::setEmitStartTime (C++ function), 57

SPH::Emitter::setNextEmitTime (C++ function), 57

SPH::Emitter::step (C++ function), 57

SPH::EmitterSystem (C++ class), 58

SPH::EmitterSystem::~~EmitterSystem (C++ function), 58

SPH::EmitterSystem::addEmitter (C++ function), 58

SPH::EmitterSystem::disableReuseParticles (C++ function), 58

SPH::EmitterSystem::EmitterSystem (C++ function), 58

SPH::EmitterSystem::enableReuseParticles (C++ function), 58

SPH::EmitterSystem::getEmitters (C++

function), 58  
 SPH::EmitterSystem::loadState (C++ function), 58  
 SPH::EmitterSystem::m\_boxMax (C++ member), 59  
 SPH::EmitterSystem::m\_boxMin (C++ member), 59  
 SPH::EmitterSystem::m\_emitters (C++ member), 59  
 SPH::EmitterSystem::m\_maxParticlesToReuse (C++ member), 59  
 SPH::EmitterSystem::m\_model (C++ member), 59  
 SPH::EmitterSystem::m\_numberOfEmittedParticles (C++ member), 59  
 SPH::EmitterSystem::m\_numReusedParticles (C++ member), 59  
 SPH::EmitterSystem::m\_reusedParticles (C++ member), 59  
 SPH::EmitterSystem::m\_reuseParticles (C++ member), 59  
 SPH::EmitterSystem::numEmittedParticles (C++ function), 58  
 SPH::EmitterSystem::numEmitters (C++ function), 58  
 SPH::EmitterSystem::numReusedParticles (C++ function), 58  
 SPH::EmitterSystem::reset (C++ function), 58  
 SPH::EmitterSystem::reuseParticles (C++ function), 59  
 SPH::EmitterSystem::saveState (C++ function), 58  
 SPH::EmitterSystem::step (C++ function), 58  
 SPH::FieldDescription (C++ struct), 30  
 SPH::FieldDescription::FieldDescription (C++ function), 31  
 SPH::FieldDescription::getFct (C++ member), 31  
 SPH::FieldDescription::name (C++ member), 31  
 SPH::FieldDescription::storeData (C++ member), 31  
 SPH::FieldDescription::type (C++ member), 31  
 SPH::FieldType (C++ enum), 128  
 SPH::FieldType::Matrix3 (C++ enumerator), 128  
 SPH::FieldType::Matrix6 (C++ enumerator), 128  
 SPH::FieldType::Scalar (C++ enumerator), 128  
 SPH::FieldType::UInt (C++ enumerator), 128  
 SPH::FieldType::Vector3 (C++ enumerator), 128  
 SPH::FieldType::Vector6 (C++ enumerator), 128  
 SPH::FluidModel (C++ class), 59  
 SPH::FluidModel::~~FluidModel (C++ function), 59  
 SPH::FluidModel::addField (C++ function), 60  
 SPH::FluidModel::computeDragForce (C++ function), 61  
 SPH::FluidModel::computeElasticity (C++ function), 61  
 SPH::FluidModel::computeSurfaceTension (C++ function), 61  
 SPH::FluidModel::computeViscosity (C++ function), 61  
 SPH::FluidModel::computeVorticity (C++ function), 61  
 SPH::FluidModel::DENSITY0 (C++ member), 62  
 SPH::FluidModel::DRAG\_METHOD (C++ member), 62  
 SPH::FluidModel::ELASTICITY\_METHOD (C++ member), 62  
 SPH::FluidModel::emittedParticles (C++ function), 60  
 SPH::FluidModel::ENUM\_DRAG\_GISSLER2017 (C++ member), 62  
 SPH::FluidModel::ENUM\_DRAG\_MACKLIN2014 (C++ member), 62  
 SPH::FluidModel::ENUM\_DRAG\_NONE (C++ member), 62  
 SPH::FluidModel::ENUM\_ELASTICITY\_BECKER2009 (C++ member), 62  
 SPH::FluidModel::ENUM\_ELASTICITY\_NONE (C++ member), 62  
 SPH::FluidModel::ENUM\_ELASTICITY\_PEER2018 (C++ member), 63  
 SPH::FluidModel::ENUM\_SURFACETENSION\_AKINCI2013 (C++ member), 62  
 SPH::FluidModel::ENUM\_SURFACETENSION\_BECKER2007 (C++ member), 62  
 SPH::FluidModel::ENUM\_SURFACETENSION\_HE2014 (C++ member), 62  
 SPH::FluidModel::ENUM\_SURFACETENSION\_NONE (C++ member), 62  
 SPH::FluidModel::ENUM\_VISCOSITY\_BENDER2017 (C++ member), 62  
 SPH::FluidModel::ENUM\_VISCOSITY\_NONE (C++ member), 62  
 SPH::FluidModel::ENUM\_VISCOSITY\_PEER2015 (C++ member), 62  
 SPH::FluidModel::ENUM\_VISCOSITY\_PEER2016 (C++ member), 62  
 SPH::FluidModel::ENUM\_VISCOSITY\_STANDARD (C++ member), 62  
 SPH::FluidModel::ENUM\_VISCOSITY\_TAKAHASHI2015 (C++ member), 62

SPH::FluidModel::ENUM\_VISCOSITY\_WEILER20\$PH::FluidModel::m\_density0 (C++ member),  
(C++ member), 62 64

SPH::FluidModel::ENUM\_VISCOSITY\_XSPH SPH::FluidModel::m\_drag (C++ member), 63  
(C++ member), 62 SPH::FluidModel::m\_dragMethod (C++ mem-  
SPH::FluidModel::ENUM\_VORTICITY\_MICROPOLAR ber), 63  
(C++ member), 62 SPH::FluidModel::m\_dragMethodChanged  
SPH::FluidModel::ENUM\_VORTICITY\_NONE (C++ member), 63  
(C++ member), 62 SPH::FluidModel::m\_elasticity (C++ mem-  
SPH::FluidModel::ENUM\_VORTICITY\_VC (C++ member), 63  
member), 62 SPH::FluidModel::m\_elasticityMethod  
SPH::FluidModel::FluidModel (C++ function), (C++ member), 63  
59 SPH::FluidModel::m\_elasticityMethodChanged  
SPH::FluidModel::getDragBase (C++ func- (C++ member), 64  
tion), 60 SPH::FluidModel::m\_emitterSystem (C++  
SPH::FluidModel::getDragMethod (C++ func- member), 63  
tion), 60 SPH::FluidModel::m\_fields (C++ member), 63  
SPH::FluidModel::getElasticityBase (C++ SPH::FluidModel::m\_id (C++ member), 63  
function), 60 SPH::FluidModel::m\_masses (C++ member), 63  
SPH::FluidModel::getElasticityMethod SPH::FluidModel::m\_numActiveParticles  
(C++ function), 60 (C++ member), 64  
SPH::FluidModel::getEmitterSystem (C++ SPH::FluidModel::m\_numActiveParticles0  
function), 60 (C++ member), 64  
SPH::FluidModel::getField (C++ function), 60 SPH::FluidModel::m\_particleId (C++ mem-  
SPH::FluidModel::getFields (C++ function), ber), 63  
60 SPH::FluidModel::m\_particleState (C++  
SPH::FluidModel::getId (C++ function), 59 member), 63  
SPH::FluidModel::getNumActiveParticles0 SPH::FluidModel::m\_pointSetIndex (C++  
(C++ function), 60 member), 64  
SPH::FluidModel::getPointSetIndex (C++ SPH::FluidModel::m\_surfaceTension (C++  
function), 60 member), 63  
SPH::FluidModel::getSurfaceTensionBase SPH::FluidModel::m\_surfaceTensionMethod  
(C++ function), 60 (C++ member), 63  
SPH::FluidModel::getSurfaceTensionMethod SPH::FluidModel::m\_surfaceTensionMethodChanged  
(C++ function), 60 (C++ member), 63  
SPH::FluidModel::getViscosityBase (C++ SPH::FluidModel::m\_V (C++ member), 63  
function), 60 SPH::FluidModel::m\_v (C++ member), 63  
SPH::FluidModel::getViscosityMethod SPH::FluidModel::m\_v0 (C++ member), 63  
(C++ function), 60 SPH::FluidModel::m\_viscosity (C++ mem-  
SPH::FluidModel::getVorticityBase (C++ ber), 63  
function), 60 SPH::FluidModel::m\_viscosityMethod (C++  
SPH::FluidModel::getVorticityMethod member), 63  
(C++ function), 60 SPH::FluidModel::m\_viscosityMethodChanged  
SPH::FluidModel::init (C++ function), 59 (C++ member), 64  
SPH::FluidModel::initMasses (C++ function), SPH::FluidModel::m\_vorticity (C++ mem-  
63 ber), 63  
SPH::FluidModel::initModel (C++ function), SPH::FluidModel::m\_vorticityMethod (C++  
60 member), 63  
SPH::FluidModel::initParameters (C++ SPH::FluidModel::m\_vorticityMethodChanged  
function), 63 (C++ member), 64  
SPH::FluidModel::loadState (C++ function), SPH::FluidModel::m\_x (C++ member), 63  
61 SPH::FluidModel::m\_x0 (C++ member), 63  
SPH::FluidModel::m\_a (C++ member), 63 SPH::FluidModel::NUM\_PARTICLES (C++ mem-  
SPH::FluidModel::m\_density (C++ member), ber), 62  
63 SPH::FluidModel::NUM\_REUSED\_PARTICLES

(C++ member), 62

SPH::FluidModel::numActiveParticles (C++ function), 60

SPH::FluidModel::numberOfFields (C++ function), 60

SPH::FluidModel::numberOfParticles (C++ function), 60

SPH::FluidModel::numParticles (C++ function), 60

SPH::FluidModel::operator= (C++ function), 59

SPH::FluidModel::performNeighborhoodSearchSort (C++ function), 60

SPH::FluidModel::releaseFluidParticles (C++ function), 63

SPH::FluidModel::removeFieldByName (C++ function), 60

SPH::FluidModel::reset (C++ function), 60

SPH::FluidModel::resizeFluidParticles (C++ function), 63

SPH::FluidModel::saveState (C++ function), 61

SPH::FluidModel::setDensity0 (C++ function), 60

SPH::FluidModel::setDragMethod (C++ function), 60

SPH::FluidModel::setDragMethodChangedCallback (C++ function), 60

SPH::FluidModel::setElasticityMethod (C++ function), 60

SPH::FluidModel::setElasticityMethodChangedCallback (C++ function), 61

SPH::FluidModel::setNumActiveParticles (C++ function), 60

SPH::FluidModel::setNumActiveParticles0 (C++ function), 60

SPH::FluidModel::setSurfaceMethodChangedCallback (C++ function), 61

SPH::FluidModel::setSurfaceTensionMethod (C++ function), 60

SPH::FluidModel::setViscosityMethod (C++ function), 60

SPH::FluidModel::setViscosityMethodChangedCallback (C++ function), 61

SPH::FluidModel::setVorticityMethod (C++ function), 60

SPH::FluidModel::setVorticityMethodChangedCallback (C++ function), 61

SPH::FluidModel::SURFACE\_TENSION\_METHOD (C++ member), 62

SPH::FluidModel::VISCOSITY\_METHOD (C++ member), 62

SPH::FluidModel::VORTICITY\_METHOD (C++ member), 62

SPH::gaussian\_abscissae\_1 (C++ member), 139

SPH::gaussian\_n\_1 (C++ member), 140

SPH::gaussian\_weights\_1 (C++ member), 140

SPH::GaussQuadrature (C++ class), 64

SPH::GaussQuadrature::Domain (C++ type), 64

SPH::GaussQuadrature::exportSamples (C++ function), 64

SPH::GaussQuadrature::Integrand (C++ type), 64

SPH::GaussQuadrature::integrate (C++ function), 64

SPH::JacobiPreconditioner1D (C++ class), 64

SPH::JacobiPreconditioner1D::\_solve\_impl (C++ function), 65

SPH::JacobiPreconditioner1D::analyzePattern (C++ function), 65

SPH::JacobiPreconditioner1D::cols (C++ function), 65

SPH::JacobiPreconditioner1D::compute (C++ function), 65

SPH::JacobiPreconditioner1D::DiagonalMatrixElementFct (C++ type), 65

SPH::JacobiPreconditioner1D::factorize (C++ function), 65

SPH::JacobiPreconditioner1D::info (C++ function), 65

SPH::JacobiPreconditioner1D::init (C++ function), 65

SPH::JacobiPreconditioner1D::JacobiPreconditioner1D (C++ function), 65

SPH::JacobiPreconditioner1D::m\_diagonalElementFct (C++ member), 65

SPH::JacobiPreconditioner1D::m\_dim (C++ member), 65

SPH::JacobiPreconditioner1D::m\_invDiag (C++ member), 65

SPH::JacobiPreconditioner1D::m\_userData (C++ member), 65

SPH::JacobiPreconditioner1D::rows (C++ function), 65

SPH::JacobiPreconditioner1D::solve (C++ function), 65

SPH::JacobiPreconditioner1D::StorageIndex (C++ type), 65

SPH::JacobiPreconditioner1D::[anonymous] (C++ enum), 65

SPH::JacobiPreconditioner1D::[anonymous]::ColsAtCor (C++ enumerator), 65

SPH::JacobiPreconditioner1D::[anonymous]::MaxColsAtCor (C++ enumerator), 65

SPH::JacobiPreconditioner3D (C++ class), 66

SPH::JacobiPreconditioner3D::\_solve\_impl (C++ function), 66



SPH::JacobiPreconditioner3D::analyzePattern (C++ function), 66

SPH::JacobiPreconditioner3D::cols (C++ function), 66

SPH::JacobiPreconditioner3D::compute (C++ function), 66

SPH::JacobiPreconditioner3D::DiagonalMatrix (C++ type), 66

SPH::JacobiPreconditioner3D::factorize (C++ function), 66

SPH::JacobiPreconditioner3D::info (C++ function), 66

SPH::JacobiPreconditioner3D::init (C++ function), 66

SPH::JacobiPreconditioner3D::JacobiPreconditioner3D (C++ function), 66

SPH::JacobiPreconditioner3D::m\_diagonalElement (C++ member), 67

SPH::JacobiPreconditioner3D::m\_dim (C++ member), 67

SPH::JacobiPreconditioner3D::m\_invDiag (C++ member), 67

SPH::JacobiPreconditioner3D::m\_userData (C++ member), 67

SPH::JacobiPreconditioner3D::rows (C++ function), 66

SPH::JacobiPreconditioner3D::solve (C++ function), 66

SPH::JacobiPreconditioner3D::StorageIndex (C++ type), 66

SPH::JacobiPreconditioner3D::[anonymous] (C++ enum), 66

SPH::JacobiPreconditioner3D::[anonymous] (C++ enumerator), 66

SPH::JacobiPreconditioner3D::[anonymous] (C++ enumerator), 66

SPH::MathFunctions (C++ class), 67

SPH::MathFunctions::eigenDecomposition (C++ function), 67

SPH::MathFunctions::extractRotation (C++ function), 67

SPH::MathFunctions::getOrthogonalVectors (C++ function), 67

SPH::MathFunctions::jacobiRotate (C++ function), 67

SPH::MathFunctions::pseudoInverse (C++ function), 67

SPH::MathFunctions::svdWithInversionHandling (C++ function), 67

SPH::MatrixReplacement (C++ class), 68

SPH::MatrixReplacement::cols (C++ function), 68

SPH::MatrixReplacement::getMatrixVecProd (C++ function), 68

SPH::MatrixReplacement::getUserData (C++ function), 68

SPH::MatrixReplacement::m\_dim (C++ member), 69

SPH::MatrixReplacement::m\_matrixVecProdFct (C++ member), 69

SPH::MatrixReplacement::m\_userData (C++ member), 69

SPH::MatrixReplacement::MatrixReplacement (C++ function), 68

SPH::MatrixReplacement::MatrixVecProdFct (C++ type), 68

SPH::MatrixReplacement::operator\* (C++ function), 68

SPH::MatrixReplacement::RealScalar (C++ type), 68

SPH::MatrixReplacement::rows (C++ function), 68

SPH::MatrixReplacement::Scalar (C++ type), 68

SPH::MatrixReplacement::StorageIndex (C++ type), 68

SPH::MatrixReplacement::[anonymous] (C++ enum), 68

SPH::MatrixReplacement::[anonymous]::ColsAtCompileTime (C++ enumerator), 68

SPH::MatrixReplacement::[anonymous]::IsRowMajor (C++ enumerator), 68

SPH::MatrixReplacement::[anonymous]::MaxColsAtCompileTime (C++ enumerator), 68

SPH::MicropolarModel\_Bender2017 (C++ class), 69

SPH::MicropolarModel\_Bender2017::~MicropolarModel\_Bender2017 (C++ function), 69

SPH::MicropolarModel\_Bender2017::INERTIA\_INVERSE (C++ member), 70

SPH::MicropolarModel\_Bender2017::initParameters (C++ function), 70

SPH::MicropolarModel\_Bender2017::m\_angularAcceleration (C++ member), 70

SPH::MicropolarModel\_Bender2017::m\_inertiaInverse (C++ member), 70

SPH::MicropolarModel\_Bender2017::m\_omega (C++ member), 70

SPH::MicropolarModel\_Bender2017::m\_viscosityOmega (C++ member), 70

SPH::MicropolarModel\_Bender2017::MicropolarModel\_Bender2017 (C++ function), 69

SPH::MicropolarModel\_Bender2017::performNeighborhood (C++ function), 69

SPH::MicropolarModel\_Bender2017::reset (C++ function), 69

SPH::MicropolarModel\_Bender2017::step (C++ function), 69

SPH::MicropolarModel\_Bender2017::VISCOSITY\_OMEGA (C++ function), 72  
 (C++ member), 70  
 SPH::NonPressureForceBase (C++ class), 70  
 (C++ function), 72  
 SPH::NonPressureForceBase::~NonPressureForceBase6Kernel (C++ class), 73  
 (C++ function), 71  
 SPH::NonPressureForceBase::getRadius (C++ function),  
 73  
 SPH::NonPressureForceBase::emittedParticles (C++ function), 71  
 SPH::NonPressureForceBase::gradW (C++ function), 73  
 SPH::NonPressureForceBase::getModel (C++ function), 71  
 SPH::NonPressureForceBase::laplacianW (C++ function), 73  
 SPH::NonPressureForceBase::init (C++ function), 71  
 SPH::NonPressureForceBase::m\_k (C++ member), 73  
 SPH::NonPressureForceBase::loadState (C++ function), 71  
 SPH::NonPressureForceBase::m\_l (C++ member), 73  
 SPH::NonPressureForceBase::m\_model (C++ member), 73  
 SPH::NonPressureForceBase::m\_m (C++ member), 73  
 SPH::NonPressureForceBase::m\_radius (C++ member),  
 73  
 SPH::NonPressureForceBase::m\_W\_zero (C++ member),  
 73  
 SPH::NonPressureForceBase::NonPressureForceBase (C++ function), 71  
 SPH::NonPressureForceBase::setRadius (C++ function),  
 73  
 SPH::NonPressureForceBase::operator= (C++ function), 71  
 SPH::NonPressureForceBase::W (C++ function), 73  
 SPH::NonPressureForceBase::performNeighborSearch (C++ function), 73  
 SPH::NonPressureForceBase::W\_zero (C++ function), 73  
 SPH::NonPressureForceBase::reset (C++ function), 71  
 SPH::NonPressureForceBase::PrecomputedKernel (C++ class), 73  
 SPH::NonPressureForceBase::PrecomputedKernel::getRadius (C++  
 function), 74  
 SPH::NonPressureForceBase::PrecomputedKernel::gradW (C++ func-  
 tion), 74  
 SPH::NonPressureForceBase::PrecomputedKernel::m\_gradW (C++  
 member), 74  
 SPH::NonPressureForceBase::PrecomputedKernel::m\_invStepSize  
 (C++ member), 74  
 SPH::NonPressureForceBase::PrecomputedKernel::m\_radius (C++  
 member), 74  
 SPH::NonPressureForceBase::PrecomputedKernel::m\_radius2 (C++  
 member), 74  
 SPH::NonPressureForceBase::PrecomputedKernel::m\_W (C++ member),  
 74  
 SPH::NonPressureForceBase::PrecomputedKernel::m\_W\_zero (C++  
 member), 74  
 SPH::NonPressureForceBase::PrecomputedKernel::setRadius (C++  
 function), 74  
 SPH::NonPressureForceBase::PrecomputedKernel::W (C++ function), 74  
 SPH::NonPressureForceBase::PrecomputedKernel::W\_zero (C++ func-  
 tion), 74  
 SPH::NonPressureForceBase::PrecomputedKernel::RegularSampling2D (C++ class), 74  
 SPH::NonPressureForceBase::PrecomputedKernel::RegularSampling2D::RegularSampling2D  
 (C++ function), 74  
 SPH::NonPressureForceBase::PrecomputedKernel::RegularSampling2D::sampleMesh (C++  
 function), 74  
 SPH::NonPressureForceBase::PrecomputedKernel::RegularSampling2D::RegularSampling2D  
 (C++ member), 74  
 SPH::NonPressureForceBase::PrecomputedKernel::RegularTriangleSampling (C++ class),  
 75  
 SPH::NonPressureForceBase::PrecomputedKernel::RegularTriangleSampling::RegularTriangleSampl-  
 ing (C++ function), 75  
 SPH::NonPressureForceBase::PrecomputedKernel::RegularTriangleSampling::sampleMesh  
 (C++ function), 75  
 SPH::NonPressureForceBase::PrecomputedKernel::RigidBodyObject (C++ class), 76

SPH::RigidBodyObject::~~RigidBodyObject (C++ function), 76	SPH::Simulation::addBoundaryModel (C++ function), 78
SPH::RigidBodyObject::addForce (C++ function), 76	SPH::Simulation::addFluidModel (C++ function), 78
SPH::RigidBodyObject::addTorque (C++ function), 76	SPH::Simulation::animateParticles (C++ function), 79
SPH::RigidBodyObject::getAngularVelocity (C++ function), 76	SPH::Simulation::BOUNDARY_HANDLING_METHOD (C++ member), 80
SPH::RigidBodyObject::getFaces (C++ function), 76	SPH::Simulation::CFL_FACTOR (C++ member), 79
SPH::RigidBodyObject::getMass (C++ function), 76	SPH::Simulation::CFL_MAX_TIMESTEPSIZE (C++ member), 79
SPH::RigidBodyObject::getPosition (C++ function), 76	SPH::Simulation::CFL_METHOD (C++ member), 79
SPH::RigidBodyObject::getRotation (C++ function), 76	SPH::Simulation::CFL_MIN_TIMESTEPSIZE (C++ member), 79
SPH::RigidBodyObject::getVelocity (C++ function), 76	SPH::Simulation::computeNonPressureForces (C++ function), 79
SPH::RigidBodyObject::getVertexNormals (C++ function), 76	SPH::Simulation::emitParticles (C++ function), 79
SPH::RigidBodyObject::getVertices (C++ function), 76	SPH::Simulation::emittedParticles (C++ function), 79
SPH::RigidBodyObject::getWorldSpacePosition (C++ function), 76	SPH::Simulation::ENABLE_Z_SORT (C++ member), 79
SPH::RigidBodyObject::getWorldSpaceRotation (C++ function), 76	SPH::Simulation::ENUM_AKINCI2012 (C++ member), 80
SPH::RigidBodyObject::isDynamic (C++ function), 76	SPH::Simulation::ENUM_BENDER2019 (C++ member), 80
SPH::RigidBodyObject::setAngularVelocity (C++ function), 76	SPH::Simulation::ENUM_CFL_ITER (C++ member), 80
SPH::RigidBodyObject::setPosition (C++ function), 76	SPH::Simulation::ENUM_CFL_NONE (C++ member), 80
SPH::RigidBodyObject::setRotation (C++ function), 76	SPH::Simulation::ENUM_CFL_STANDARD (C++ member), 80
SPH::RigidBodyObject::setVelocity (C++ function), 76	SPH::Simulation::ENUM_GRADKERNEL_CUBIC (C++ member), 80
SPH::SimpleQuadrature (C++ class), 77	SPH::Simulation::ENUM_GRADKERNEL_CUBIC_2D (C++ member), 80
SPH::SimpleQuadrature::determineSamplePointsInCell (C++ function), 77	SPH::Simulation::ENUM_GRADKERNEL_POLY6 (C++ member), 80
SPH::SimpleQuadrature::determineSamplePointsInSphere (C++ function), 77	SPH::Simulation::ENUM_GRADKERNEL_PRECOMPUTED_CUBIC (C++ member), 80
SPH::SimpleQuadrature::Domain (C++ type), 77	SPH::Simulation::ENUM_GRADKERNEL_SPIKY (C++ member), 80
SPH::SimpleQuadrature::Integrand (C++ type), 77	SPH::Simulation::ENUM_GRADKERNEL_WENDLANDQUINTICC2 (C++ member), 80
SPH::SimpleQuadrature::integrate (C++ function), 77	SPH::Simulation::ENUM_GRADKERNEL_WENDLANDQUINTICC2_2D (C++ member), 80
SPH::SimpleQuadrature::m_samplePoints (C++ member), 77	SPH::Simulation::ENUM_KERNEL_CUBIC (C++ member), 80
SPH::SimpleQuadrature::m_volume (C++ member), 77	SPH::Simulation::ENUM_KERNEL_CUBIC_2D (C++ member), 80
SPH::Simulation (C++ class), 77	SPH::Simulation::ENUM_KERNEL_POLY6 (C++ member), 80
SPH::Simulation::~~Simulation (C++ function), 78	



SPH::Simulation::ENUM\_KERNEL\_PRECOMPUTEDSPHISimulation::hasCurrent (C++ function),  
(C++ member), 80 79

SPH::Simulation::ENUM\_KERNEL\_SPIKY (C++ member), 80 SPH::Simulation::init (C++ function), 78

SPH::Simulation::ENUM\_KERNEL\_WENDLANDQUINTICC2 function), 81 SPH::Simulation::initParameters (C++  
(C++ member), 80 SPH::Simulation::is2DSimulation (C++  
SPH::Simulation::ENUM\_KERNEL\_WENDLANDQUINTICC2 function), 78 SPH::Simulation::KERNEL\_METHOD (C++ mem-  
(C++ member), 80 SPH::Simulation::loadState (C++ function),  
SPH::Simulation::ENUM\_KOSCHIER2017 (C++ member), 79  
SPH::Simulation::ENUM\_SIMULATION\_DFSPH (C++ member), 80 SPH::Simulation::m\_animationFieldSystem  
SPH::Simulation::ENUM\_SIMULATION\_IISPH (C++ member), 81 (C++ member), 81  
SPH::Simulation::ENUM\_SIMULATION\_PBF (C++ member), 81 SPH::Simulation::m\_boundaryHandlingMethod  
SPH::Simulation::ENUM\_SIMULATION\_PCISPH (C++ member), 81 (C++ member), 81  
SPH::Simulation::ENUM\_SIMULATION\_PF (C++ member), 81 SPH::Simulation::m\_boundaryModels (C++  
SPH::Simulation::ENUM\_SIMULATION\_WCSPH (C++ member), 81 member), 81  
SPH::Simulation::getAnimationFieldSystem (C++ function), 78 SPH::Simulation::m\_cflFactor (C++ mem-  
SPH::Simulation::getBoundaryHandlingMethod (C++ function), 78 ber), 81  
SPH::Simulation::getBoundaryModel (C++ function), 78 SPH::Simulation::m\_cflMaxTimeStepSize  
SPH::Simulation::getBoundaryModelFromPointSet (C++ function), 78 (C++ member), 81  
SPH::Simulation::getCurrent (C++ function), 79 SPH::Simulation::m\_cflMethod (C++ mem-  
SPH::Simulation::getFluidModel (C++ function), 78 ber), 81  
SPH::Simulation::getFluidModelFromPointSet (C++ function), 78 SPH::Simulation::m\_cflMinTimeStepSize  
SPH::Simulation::getGradKernel (C++ function), 78 (C++ member), 81  
SPH::Simulation::getKernel (C++ function), 78 SPH::Simulation::m\_enableZSort (C++ mem-  
SPH::Simulation::getNeighborhoodSearch (C++ function), 79 ber), 81  
SPH::Simulation::getParticleRadius (C++ function), 79 SPH::Simulation::m\_fluidModels (C++ mem-  
SPH::Simulation::getSimulationMethod (C++ function), 78 ber), 81  
SPH::Simulation::getSupportRadius (C++ function), 79 SPH::Simulation::m\_gradKernelFct (C++  
SPH::Simulation::getTimeStep (C++ function), 78 member), 81  
SPH::Simulation::GRAD\_KERNEL\_METHOD (C++ member), 80 SPH::Simulation::m\_gradKernelMethod  
SPH::Simulation::GRAVITATION (C++ member), 79 (C++ member), 81  
SPH::Simulation::m\_gravitation (C++ mem-  
SPH::Simulation::m\_kernelFct (C++ mem-  
SPH::Simulation::m\_kernelMethod (C++ member), 81  
SPH::Simulation::m\_neighborhoodSearch  
SPH::Simulation::m\_particleRadius (C++ member), 81  
SPH::Simulation::m\_sim2D (C++ member), 81  
SPH::Simulation::m\_simulationMethod  
SPH::Simulation::m\_simulationMethodChanged  
SPH::Simulation::m\_supportRadius (C++ member), 81  
SPH::Simulation::m\_timeStep (C++ member), 81  
SPH::Simulation::m\_W\_zero (C++ member), 81  
SPH::Simulation::numberOfBoundaryModels

(C++ function), 78

SPH::Simulation::numberOfFluidModels (C++ function), 78

SPH::Simulation::operator= (C++ function), 78

SPH::Simulation::PARTICLE\_RADIUS (C++ member), 79

SPH::Simulation::performNeighborhoodSearch (C++ function), 79

SPH::Simulation::performNeighborhoodSearchSort (C++ function), 79

SPH::Simulation::PrecomputedCubicKernel (C++ type), 78

SPH::Simulation::reset (C++ function), 78

SPH::Simulation::saveState (C++ function), 79

SPH::Simulation::setBoundaryHandlingMethod (C++ function), 78

SPH::Simulation::setCurrent (C++ function), 79

SPH::Simulation::setGradKernel (C++ function), 78

SPH::Simulation::setKernel (C++ function), 78

SPH::Simulation::setParticleRadius (C++ function), 79

SPH::Simulation::setSimulationMethod (C++ function), 78

SPH::Simulation::setSimulationMethodChangedCallback (C++ function), 78

SPH::Simulation::SIM\_2D (C++ member), 79

SPH::Simulation::Simulation (C++ function), 78

SPH::Simulation::SIMULATION\_METHOD (C++ member), 80

SPH::Simulation::updateBoundaryVolume (C++ function), 78

SPH::Simulation::updateTimeStepSize (C++ function), 79

SPH::Simulation::updateTimeStepSizeCFL (C++ function), 79

SPH::Simulation::zSortEnabled (C++ function), 78

SPH::SimulationDataDFSPH (C++ class), 82

SPH::SimulationDataDFSPH::~~SimulationDataDFSPH (C++ function), 82

SPH::SimulationDataDFSPH::cleanup (C++ function), 82

SPH::SimulationDataDFSPH::emittedParticles (C++ function), 82

SPH::SimulationDataDFSPH::init (C++ function), 82

SPH::SimulationDataDFSPH::m\_density\_adv (C++ member), 82

SPH::SimulationDataDFSPH::m\_factor (C++ member), 82

SPH::SimulationDataDFSPH::m\_kappa (C++ member), 82

SPH::SimulationDataDFSPH::m\_kappaV (C++ member), 82

SPH::SimulationDataDFSPH::performNeighborhoodSearch (C++ function), 82

SPH::SimulationDataDFSPH::reset (C++ function), 82

SPH::SimulationDataDFSPH::SimulationDataDFSPH (C++ function), 82

SPH::SimulationDataIISPH (C++ class), 83

SPH::SimulationDataIISPH::~~SimulationDataIISPH (C++ function), 83

SPH::SimulationDataIISPH::cleanup (C++ function), 83

SPH::SimulationDataIISPH::emittedParticles (C++ function), 83

SPH::SimulationDataIISPH::init (C++ function), 83

SPH::SimulationDataIISPH::m\_aid (C++ member), 84

SPH::SimulationDataIISPH::m\_density\_adv (C++ member), 84

SPH::SimulationDataIISPH::m\_did (C++ member), 84

SPH::SimulationDataIISPH::m\_dij\_pj (C++ member), 84

SPH::SimulationDataIISPH::m\_lastPressure (C++ member), 84

SPH::SimulationDataIISPH::m\_pressure (C++ member), 84

SPH::SimulationDataIISPH::m\_pressureAccel (C++ member), 84

SPH::SimulationDataIISPH::performNeighborhoodSearch (C++ function), 83

SPH::SimulationDataIISPH::reset (C++ function), 83

SPH::SimulationDataIISPH::SimulationDataIISPH (C++ function), 83

SPH::SimulationDataPBF (C++ class), 84

SPH::SimulationDataPBF::~~SimulationDataPBF (C++ function), 84

SPH::SimulationDataPBF::cleanup (C++ function), 84

SPH::SimulationDataPBF::emittedParticles (C++ function), 84

SPH::SimulationDataPBF::init (C++ function), 84

SPH::SimulationDataPBF::m\_deltaX (C++ member), 85

SPH::SimulationDataPBF::m\_lambda (C++ member), 85

SPH::SimulationDataPBF::m\_lastX (C++ member), 85

SPH::SimulationDataPBF::m\_oldX (C++ member), 85

SPH::SimulationDataPBF::performNeighborhoodSearch (C++ function), 84

SPH::SimulationDataPBF::reset (C++ function), 84

SPH::SimulationDataPBF::SimulationDataPBF (C++ function), 84

SPH::SimulationDataPCISPH (C++ class), 85

SPH::SimulationDataPCISPH::~SimulationDataPCISPH (C++ function), 85

SPH::SimulationDataPCISPH::cleanup (C++ function), 85

SPH::SimulationDataPCISPH::emittedParticles (C++ function), 86

SPH::SimulationDataPCISPH::getPCISPH\_ScaleFactor (C++ function), 85

SPH::SimulationDataPCISPH::init (C++ function), 85

SPH::SimulationDataPCISPH::m\_densityAdv (C++ member), 86

SPH::SimulationDataPCISPH::m\_lastV (C++ member), 86

SPH::SimulationDataPCISPH::m\_lastX (C++ member), 86

SPH::SimulationDataPCISPH::m\_pcisph\_factor (C++ member), 86

SPH::SimulationDataPCISPH::m\_pressure (C++ member), 86

SPH::SimulationDataPCISPH::m\_pressureAccel (C++ member), 86

SPH::SimulationDataPCISPH::performNeighborhoodSearch (C++ function), 85

SPH::SimulationDataPCISPH::reset (C++ function), 85

SPH::SimulationDataPCISPH::SimulationDataPCISPH (C++ function), 85

SPH::SimulationDataPF (C++ class), 86

SPH::SimulationDataPF::~SimulationDataPF (C++ function), 87

SPH::SimulationDataPF::cleanup (C++ function), 87

SPH::SimulationDataPF::emittedParticles (C++ function), 87

SPH::SimulationDataPF::init (C++ function), 87

SPH::SimulationDataPF::m\_mat\_diag (C++ member), 87

SPH::SimulationDataPF::m\_num\_fluid\_neighbors (C++ member), 87

SPH::SimulationDataPF::m\_old\_position (C++ member), 87

SPH::SimulationDataPF::m\_particleOffset (C++ member), 87

SPH::SimulationDataPF::m\_s (C++ member), 87

SPH::SimulationDataPF::performNeighborhoodSearch (C++ function), 87

SPH::SimulationDataPF::reset (C++ function), 87

SPH::SimulationDataPF::SimulationDataPF (C++ function), 87

SPH::SimulationDataWCSPH (C++ class), 88

SPH::SimulationDataWCSPH::~SimulationDataWCSPH (C++ function), 88

SPH::SimulationDataWCSPH::cleanup (C++ function), 88

SPH::SimulationDataWCSPH::emittedParticles (C++ function), 88

SPH::SimulationDataWCSPH::init (C++ function), 88

SPH::SimulationDataWCSPH::m\_pressure (C++ member), 88

SPH::SimulationDataWCSPH::m\_pressureAccel (C++ member), 88

SPH::SimulationDataWCSPH::performNeighborhoodSearch (C++ function), 88

SPH::SimulationDataWCSPH::reset (C++ function), 88

SPH::SimulationDataWCSPH::SimulationDataWCSPH (C++ function), 88

SPH::SimulationMethods (C++ enum), 129

SPH::SimulationMethods::DFSPH (C++ enumerator), 129

SPH::SimulationMethods::IISPH (C++ enumerator), 129

SPH::SimulationMethods::NumSimulationMethods (C++ enumerator), 129

SPH::SimulationMethods::PBF (C++ enumerator), 129

SPH::SimulationMethods::PCISPH (C++ enumerator), 129

SPH::SimulationMethods::PF (C++ enumerator), 129

SPH::SimulationMethods::WCSPH (C++ enumerator), 129

SPH::SpikyKernel (C++ class), 89

SPH::SpikyKernel::getRadius (C++ function), 89

SPH::SpikyKernel::gradW (C++ function), 89

SPH::SpikyKernel::m\_k (C++ member), 89

SPH::SpikyKernel::m\_l (C++ member), 89

SPH::SpikyKernel::m\_radius (C++ member), 89

SPH::SpikyKernel::m\_W\_zero (C++ member), 89

SPH::SpikyKernel::setRadius (C++ *function*), 89

SPH::SpikyKernel::W (C++ *function*), 89

SPH::SpikyKernel::W\_zero (C++ *function*), 89

SPH::StaticRigidBody (C++ *class*), 90

SPH::StaticRigidBody::addForce (C++ *function*), 90

SPH::StaticRigidBody::addTorque (C++ *function*), 90

SPH::StaticRigidBody::getAngularVelocity (C++ *function*), 90

SPH::StaticRigidBody::getFaces (C++ *function*), 90

SPH::StaticRigidBody::getGeometry (C++ *function*), 90

SPH::StaticRigidBody::getMass (C++ *function*), 90

SPH::StaticRigidBody::getPosition (C++ *function*), 90

SPH::StaticRigidBody::getRotation (C++ *function*), 90

SPH::StaticRigidBody::getVelocity (C++ *function*), 90

SPH::StaticRigidBody::getVertexNormals (C++ *function*), 90

SPH::StaticRigidBody::getVertices (C++ *function*), 90

SPH::StaticRigidBody::getWorldSpacePosition (C++ *function*), 90

SPH::StaticRigidBody::getWorldSpaceRotation (C++ *function*), 90

SPH::StaticRigidBody::isDynamic (C++ *function*), 90

SPH::StaticRigidBody::m\_geometry (C++ *member*), 90

SPH::StaticRigidBody::m\_R (C++ *member*), 90

SPH::StaticRigidBody::m\_R\_world (C++ *member*), 90

SPH::StaticRigidBody::m\_x (C++ *member*), 90

SPH::StaticRigidBody::m\_x\_world (C++ *member*), 90

SPH::StaticRigidBody::m\_zero (C++ *member*), 90

SPH::StaticRigidBody::setAngularVelocity (C++ *function*), 90

SPH::StaticRigidBody::setPosition (C++ *function*), 90

SPH::StaticRigidBody::setRotation (C++ *function*), 90

SPH::StaticRigidBody::setVelocity (C++ *function*), 90

SPH::StaticRigidBody::setWorldSpacePosition (C++ *function*), 90

SPH::StaticRigidBody::setWorldSpaceRotation (C++ *function*), 90

SPH::StaticRigidBody::StaticRigidBody (C++ *function*), 90

SPH::SurfaceSamplingMode (C++ *enum*), 129

SPH::SurfaceSamplingMode::PoissonDisk (C++ *enumerator*), 129

SPH::SurfaceSamplingMode::Regular2D (C++ *enumerator*), 129

SPH::SurfaceSamplingMode::RegularTriangle (C++ *enumerator*), 129

SPH::SurfaceTension\_Akinci2013 (C++ *class*), 91

SPH::SurfaceTension\_Akinci2013::~~SurfaceTension\_Akinci2013 (C++ *function*), 91

SPH::SurfaceTension\_Akinci2013::computeNormals (C++ *function*), 91

SPH::SurfaceTension\_Akinci2013::m\_normals (C++ *member*), 91

SPH::SurfaceTension\_Akinci2013::performNeighborhoodSearch (C++ *function*), 91

SPH::SurfaceTension\_Akinci2013::reset (C++ *function*), 91

SPH::SurfaceTension\_Akinci2013::step (C++ *function*), 91

SPH::SurfaceTension\_Akinci2013::SurfaceTension\_Akinci2013 (C++ *function*), 91

SPH::SurfaceTension\_Becker2007 (C++ *class*), 92

SPH::SurfaceTension\_Becker2007::~~SurfaceTension\_Becker2007 (C++ *function*), 92

SPH::SurfaceTension\_Becker2007::reset (C++ *function*), 92

SPH::SurfaceTension\_Becker2007::step (C++ *function*), 92

SPH::SurfaceTension\_Becker2007::SurfaceTension\_Becker2007 (C++ *function*), 92

SPH::SurfaceTension\_He2014 (C++ *class*), 92

SPH::SurfaceTension\_He2014::~~SurfaceTension\_He2014 (C++ *function*), 92

SPH::SurfaceTension\_He2014::m\_color (C++ *member*), 93

SPH::SurfaceTension\_He2014::m\_gradC2 (C++ *member*), 93

SPH::SurfaceTension\_He2014::performNeighborhoodSearch (C++ *function*), 92

SPH::SurfaceTension\_He2014::reset (C++ *function*), 92

SPH::SurfaceTension\_He2014::step (C++ *function*), 92

SPH::SurfaceTension\_He2014::SurfaceTension\_He2014 (C++ *function*), 92

SPH::SurfaceTensionBase (C++ *class*), 93

SPH::SurfaceTensionBase::~~SurfaceTensionBase (C++ *function*), 93

SPH::SurfaceTensionBase::initParameters (C++ function), 94  
 SPH::SurfaceTensionBase::m\_surfaceTension (C++ member), 94  
 SPH::SurfaceTensionBase::m\_surfaceTensionBoundary (C++ member), 94  
 SPH::SurfaceTensionBase::SURFACE\_TENSIONS (C++ member), 93  
 SPH::SurfaceTensionBase::SURFACE\_TENSIONS\_BOUNDARY (C++ member), 93  
 SPH::SurfaceTensionBase::SurfaceTensionBase (C++ function), 93  
 SPH::SurfaceTensionMethods (C++ enum), 129  
 SPH::SurfaceTensionMethods::Akinci2013 (C++ enumerator), 129  
 SPH::SurfaceTensionMethods::Becker2007 (C++ enumerator), 129  
 SPH::SurfaceTensionMethods::He2014 (C++ enumerator), 129  
 SPH::SurfaceTensionMethods::None (C++ enumerator), 129  
 SPH::SurfaceTensionMethods::NumSurfaceTensionMethods (C++ enumerator), 129  
 SPH::TimeIntegration (C++ class), 94  
 SPH::TimeIntegration::semiImplicitEuler (C++ function), 94  
 SPH::TimeIntegration::velocityUpdateFirstOrder (C++ function), 94  
 SPH::TimeIntegration::velocityUpdateSecondOrder (C++ function), 95  
 SPH::TimeManager (C++ class), 95  
 SPH::TimeManager::~~TimeManager (C++ function), 95  
 SPH::TimeManager::getCurrent (C++ function), 95  
 SPH::TimeManager::getTime (C++ function), 95, 133  
 SPH::TimeManager::getTimeStepSize (C++ function), 95  
 SPH::TimeManager::hasCurrent (C++ function), 95  
 SPH::TimeManager::loadState (C++ function), 95  
 SPH::TimeManager::saveState (C++ function), 95  
 SPH::TimeManager::setCurrent (C++ function), 95  
 SPH::TimeManager::setTime (C++ function), 95  
 SPH::TimeManager::setTimeStepSize (C++ function), 95  
 SPH::TimeManager::TimeManager (C++ function), 95  
 SPH::TimeStep (C++ class), 96  
 SPH::TimeStep::~~TimeStep (C++ function), 96  
 SPH::TimeStep::approximateNormal (C++ function), 97  
 SPH::TimeStep::clearAccelerations (C++ function), 97  
 SPH::TimeStep::computeDensities (C++ function), 97  
 SPH::TimeStep::computeDensityAndGradient (C++ function), 97  
 SPH::TimeStep::computeVolumeAndBoundaryX (C++ function), 97  
 SPH::TimeStep::emittedParticles (C++ function), 96  
 SPH::TimeStep::init (C++ function), 96  
 SPH::TimeStep::initParameters (C++ function), 97  
 SPH::TimeStep::loadState (C++ function), 96  
 SPH::TimeStep::m\_iterations (C++ member), 97  
 SPH::TimeStep::m\_maxError (C++ member), 97  
 SPH::TimeStep::m\_maxIterations (C++ member), 97  
 SPH::TimeStep::m\_minIterations (C++ member), 97  
 SPH::TimeStep::MAX\_ERROR (C++ member), 97  
 SPH::TimeStep::MAX\_ITERATIONS (C++ member), 97  
 SPH::TimeStep::MIN\_ITERATIONS (C++ member), 97  
 SPH::TimeStep::reset (C++ function), 96  
 SPH::TimeStep::resize (C++ function), 96  
 SPH::TimeStep::saveState (C++ function), 96  
 SPH::TimeStep::SOLVER\_ITERATIONS (C++ member), 97  
 SPH::TimeStep::step (C++ function), 96  
 SPH::TimeStep::TimeStep (C++ function), 96  
 SPH::TimeStepDFSPH (C++ class), 98  
 SPH::TimeStepDFSPH::~~TimeStepDFSPH (C++ function), 98  
 SPH::TimeStepDFSPH::computeDensityAdv (C++ function), 98  
 SPH::TimeStepDFSPH::computeDensityChange (C++ function), 98  
 SPH::TimeStepDFSPH::computeDFSPHFactor (C++ function), 98  
 SPH::TimeStepDFSPH::divergenceSolve (C++ function), 98  
 SPH::TimeStepDFSPH::divergenceSolveIteration (C++ function), 98  
 SPH::TimeStepDFSPH::emittedParticles (C++ function), 98  
 SPH::TimeStepDFSPH::initParameters (C++ function), 98  
 SPH::TimeStepDFSPH::m\_counter (C++ member), 99



SPH::TimeStepDFSPH::m\_enableDivergenceSolver (C++ member), 99

SPH::TimeStepDFSPH::m\_eps (C++ member), 99

SPH::TimeStepDFSPH::m\_iterationsV (C++ member), 99

SPH::TimeStepDFSPH::m\_maxErrorV (C++ member), 99

SPH::TimeStepDFSPH::m\_maxIterationsV (C++ member), 99

SPH::TimeStepDFSPH::m\_simulationData (C++ member), 99

SPH::TimeStepDFSPH::MAX\_ERROR\_V (C++ member), 98

SPH::TimeStepDFSPH::MAX\_ITERATIONS\_V (C++ member), 98

SPH::TimeStepDFSPH::performNeighborhoodSearch (C++ function), 98

SPH::TimeStepDFSPH::pressureSolve (C++ function), 98

SPH::TimeStepDFSPH::pressureSolveIteration (C++ function), 98

SPH::TimeStepDFSPH::reset (C++ function), 98

SPH::TimeStepDFSPH::resize (C++ function), 98

SPH::TimeStepDFSPH::SOLVER\_ITERATIONS\_V (C++ member), 98

SPH::TimeStepDFSPH::step (C++ function), 98

SPH::TimeStepDFSPH::TimeStepDFSPH (C++ function), 98

SPH::TimeStepDFSPH::USE\_DIVERGENCE\_SOLVER (C++ member), 98

SPH::TimeStepDFSPH::warmstartDivergenceSolver (C++ function), 98

SPH::TimeStepDFSPH::warmstartPressureSolve (C++ function), 98

SPH::TimeStepIISPH (C++ class), 99

SPH::TimeStepIISPH::~~TimeStepIISPH (C++ function), 99

SPH::TimeStepIISPH::computePressureAcceleration (C++ function), 99

SPH::TimeStepIISPH::emittedParticles (C++ function), 100

SPH::TimeStepIISPH::getSimulationData (C++ function), 99

SPH::TimeStepIISPH::integration (C++ function), 99

SPH::TimeStepIISPH::m\_counter (C++ member), 100

SPH::TimeStepIISPH::m\_simulationData (C++ member), 100

SPH::TimeStepIISPH::performNeighborhoodSearch (C++ function), 99

SPH::TimeStepIISPH::predictAdvection (C++ function), 99

SPH::TimeStepIISPH::pressureSolve (C++ function), 99

SPH::TimeStepIISPH::pressureSolveIteration (C++ function), 99

SPH::TimeStepIISPH::reset (C++ function), 99

SPH::TimeStepIISPH::resize (C++ function), 99

SPH::TimeStepIISPH::step (C++ function), 99

SPH::TimeStepIISPH::TimeStepIISPH (C++ function), 99

SPH::TimeStepPBF (C++ class), 100

SPH::TimeStepPBF::~~TimeStepPBF (C++ function), 100

SPH::TimeStepPBF::emittedParticles (C++ function), 101

SPH::TimeStepPBF::ENUM\_PBF\_FIRST\_ORDER (C++ member), 100

SPH::TimeStepPBF::ENUM\_PBF\_SECOND\_ORDER (C++ member), 100

SPH::TimeStepPBF::initParameters (C++ function), 101

SPH::TimeStepPBF::m\_counter (C++ member), 101

SPH::TimeStepPBF::m\_simulationData (C++ member), 101

SPH::TimeStepPBF::m\_velocityUpdateMethod (C++ member), 101

SPH::TimeStepPBF::performNeighborhoodSearch (C++ function), 101

SPH::TimeStepPBF::pressureSolve (C++ function), 101

SPH::TimeStepPBF::pressureSolveIteration (C++ function), 101

SPH::TimeStepPBF::reset (C++ function), 100

SPH::TimeStepPBF::resize (C++ function), 100

SPH::TimeStepPBF::step (C++ function), 100

SPH::TimeStepPBF::TimeStepPBF (C++ function), 100

SPH::TimeStepPBF::VELOCITY\_UPDATE\_METHOD (C++ member), 100

SPH::TimeStepPCISPH (C++ class), 101

SPH::TimeStepPCISPH::~~TimeStepPCISPH (C++ function), 101

SPH::TimeStepPCISPH::emittedParticles (C++ function), 102

SPH::TimeStepPCISPH::m\_counter (C++ member), 102

SPH::TimeStepPCISPH::m\_simulationData (C++ member), 102

SPH::TimeStepPCISPH::performNeighborhoodSearch (C++ function), 102

SPH::TimeStepPCISPH::pressureSolve (C++ function), 102

SPH::TimeStepPCISPH::pressureSolveIteration

(C++ function), 102

SPH::TimeStepPCISPH::reset (C++ function), 101

SPH::TimeStepPCISPH::resize (C++ function), 101

SPH::TimeStepPCISPH::step (C++ function), 101

SPH::TimeStepPCISPH::TimeStepPCISPH (C++ function), 101

SPH::TimeStepPF (C++ class), 102

SPH::TimeStepPF::~~TimeStepPF (C++ function), 102

SPH::TimeStepPF::addAccellerationToVelocity (C++ function), 103

SPH::TimeStepPF::emittedParticles (C++ function), 103

SPH::TimeStepPF::initialGuessForPositions (C++ function), 103

SPH::TimeStepPF::initParameters (C++ function), 103

SPH::TimeStepPF::m\_counter (C++ member), 103

SPH::TimeStepPF::m\_numActiveParticlesTotal (C++ member), 103

SPH::TimeStepPF::m\_simulationData (C++ member), 103

SPH::TimeStepPF::m\_solver (C++ member), 103

SPH::TimeStepPF::m\_stiffness (C++ member), 103

SPH::TimeStepPF::matrixFreeRHS (C++ function), 103

SPH::TimeStepPF::matrixVecProd (C++ function), 103

SPH::TimeStepPF::performNeighborhoodSearch (C++ function), 103

SPH::TimeStepPF::preparePreconditioner (C++ function), 103

SPH::TimeStepPF::reset (C++ function), 102

SPH::TimeStepPF::resize (C++ function), 102

SPH::TimeStepPF::solvePDConstraints (C++ function), 103

SPH::TimeStepPF::Solver (C++ type), 103

SPH::TimeStepPF::step (C++ function), 102

SPH::TimeStepPF::STIFFNESS (C++ member), 103

SPH::TimeStepPF::TimeStepPF (C++ function), 102

SPH::TimeStepPF::updatePositionsAndVelocities (C++ function), 103

SPH::TimeStepPF::VectorXr (C++ type), 103

SPH::TimeStepPF::VectorXrMap (C++ type), 103

SPH::TimeStepWCSPH (C++ class), 104

SPH::TimeStepWCSPH::~~TimeStepWCSPH (C++ function), 104

SPH::TimeStepWCSPH::computePressureAccels (C++ function), 104

SPH::TimeStepWCSPH::emittedParticles (C++ function), 104

SPH::TimeStepWCSPH::EXPONENT (C++ member), 104

SPH::TimeStepWCSPH::initParameters (C++ function), 104

SPH::TimeStepWCSPH::m\_counter (C++ member), 105

SPH::TimeStepWCSPH::m\_exponent (C++ member), 105

SPH::TimeStepWCSPH::m\_simulationData (C++ member), 105

SPH::TimeStepWCSPH::m\_stiffness (C++ member), 105

SPH::TimeStepWCSPH::performNeighborhoodSearch (C++ function), 104

SPH::TimeStepWCSPH::reset (C++ function), 104

SPH::TimeStepWCSPH::resize (C++ function), 104

SPH::TimeStepWCSPH::step (C++ function), 104

SPH::TimeStepWCSPH::STIFFNESS (C++ member), 104

SPH::TimeStepWCSPH::TimeStepWCSPH (C++ function), 104

SPH::TriangleMesh (C++ class), 105

SPH::TriangleMesh::~~TriangleMesh (C++ function), 105

SPH::TriangleMesh::addFace (C++ function), 105

SPH::TriangleMesh::addVertex (C++ function), 105

SPH::TriangleMesh::Faces (C++ type), 105

SPH::TriangleMesh::getFaceNormals (C++ function), 105

SPH::TriangleMesh::getFaces (C++ function), 105

SPH::TriangleMesh::getVertexNormals (C++ function), 105

SPH::TriangleMesh::getVertices (C++ function), 105

SPH::TriangleMesh::initMesh (C++ function), 105

SPH::TriangleMesh::m\_indices (C++ member), 106

SPH::TriangleMesh::m\_normals (C++ member), 106

SPH::TriangleMesh::m\_vertexNormals (C++ member), 106

SPH::TriangleMesh::m\_x (C++ member), 106  
 SPH::TriangleMesh::Normals (C++ type), 105  
 SPH::TriangleMesh::numFaces (C++ function), 106  
 SPH::TriangleMesh::numVertices (C++ function), 106  
 SPH::TriangleMesh::release (C++ function), 105  
 SPH::TriangleMesh::TriangleMesh (C++ function), 105  
 SPH::TriangleMesh::updateNormals (C++ function), 106  
 SPH::TriangleMesh::updateVertexNormals (C++ function), 106  
 SPH::TriangleMesh::Vertices (C++ type), 105  
 SPH::Viscosity\_Bender2017 (C++ class), 106  
 SPH::Viscosity\_Bender2017::~~Viscosity\_Bender2017 (C++ member), 107  
 SPH::Viscosity\_Bender2017::computeTargetStrainRate (C++ function), 106  
 SPH::Viscosity\_Bender2017::computeViscosityFactor (C++ function), 106  
 SPH::Viscosity\_Bender2017::initParameters (C++ function), 107  
 SPH::Viscosity\_Bender2017::ITERATIONS (C++ member), 107  
 SPH::Viscosity\_Bender2017::m\_iterations (C++ member), 107  
 SPH::Viscosity\_Bender2017::m\_maxError (C++ member), 107  
 SPH::Viscosity\_Bender2017::m\_maxIter (C++ member), 107  
 SPH::Viscosity\_Bender2017::m\_targetStrainRate (C++ member), 107  
 SPH::Viscosity\_Bender2017::m\_viscosityFactor (C++ member), 107  
 SPH::Viscosity\_Bender2017::m\_viscosityLambda (C++ member), 107  
 SPH::Viscosity\_Bender2017::MAX\_ERROR (C++ member), 107  
 SPH::Viscosity\_Bender2017::MAX\_ITERATIONS (C++ member), 107  
 SPH::Viscosity\_Bender2017::performNeighborhoodSearch (C++ function), 106  
 SPH::Viscosity\_Bender2017::reset (C++ function), 106  
 SPH::Viscosity\_Bender2017::step (C++ function), 106  
 SPH::Viscosity\_Bender2017::Viscosity\_Bender2017 (C++ function), 106  
 SPH::Viscosity\_Peer2015 (C++ class), 108  
 SPH::Viscosity\_Peer2015::~~Viscosity\_Peer2015 (C++ member), 108  
 SPH::Viscosity\_Peer2015::computeDensities (C++ member), 108  
 SPH::Viscosity\_Peer2015::initParameters (C++ function), 108  
 SPH::Viscosity\_Peer2015::ITERATIONS (C++ member), 108  
 SPH::Viscosity\_Peer2015::m\_density (C++ member), 109  
 SPH::Viscosity\_Peer2015::m\_iterations (C++ member), 109  
 SPH::Viscosity\_Peer2015::m\_maxError (C++ member), 109  
 SPH::Viscosity\_Peer2015::m\_maxIter (C++ member), 109  
 SPH::Viscosity\_Peer2015::m\_solver (C++ member), 109  
 SPH::Viscosity\_Peer2015::m\_targetNablaV (C++ member), 109  
 SPH::Viscosity\_Peer2015::matrixVecProd (C++ function), 108  
 SPH::Viscosity\_Peer2015::MAX\_ERROR (C++ member), 108  
 SPH::Viscosity\_Peer2015::MAX\_ITERATIONS (C++ member), 108  
 SPH::Viscosity\_Peer2015::performNeighborhoodSearch (C++ function), 108  
 SPH::Viscosity\_Peer2015::reset (C++ function), 108  
 SPH::Viscosity\_Peer2015::Solver (C++ type), 108  
 SPH::Viscosity\_Peer2015::step (C++ function), 108  
 SPH::Viscosity\_Peer2015::Viscosity\_Peer2015 (C++ function), 108  
 SPH::Viscosity\_Peer2016 (C++ class), 109  
 SPH::Viscosity\_Peer2016::~~Viscosity\_Peer2016 (C++ member), 109  
 SPH::Viscosity\_Peer2016::computeDensities (C++ function), 110  
 SPH::Viscosity\_Peer2016::initParameters (C++ function), 110  
 SPH::Viscosity\_Peer2016::ITERATIONS\_OMEGA (C++ member), 110  
 SPH::Viscosity\_Peer2016::ITERATIONS\_V (C++ member), 110  
 SPH::Viscosity\_Peer2016::m\_density (C++ member), 110  
 SPH::Viscosity\_Peer2016::m\_iterationsOmega (C++ member), 110  
 SPH::Viscosity\_Peer2016::m\_iterationsV (C++ member), 110  
 SPH::Viscosity\_Peer2016::m\_maxErrorOmega (C++ member), 110  
 SPH::Viscosity\_Peer2016::m\_maxErrorV (C++ member), 110



SPH::Viscosity\_Peer2016::m\_maxIterOmega (C++ function), 113  
 (C++ member), 110  
 SPH::Viscosity\_Peer2016::m\_maxIterV (C++ member), 110  
 SPH::Viscosity\_Peer2016::m\_omega (C++ member), 110  
 SPH::Viscosity\_Peer2016::m\_solverOmega (C++ member), 110  
 SPH::Viscosity\_Peer2016::m\_solverV (C++ member), 110  
 SPH::Viscosity\_Peer2016::m\_targetNablaV (C++ member), 110  
 SPH::Viscosity\_Peer2016::matrixVecProdOmega (C++ function), 110  
 (C++ member), 110  
 SPH::Viscosity\_Peer2016::matrixVecProdV (C++ function), 110  
 (C++ member), 110  
 SPH::Viscosity\_Peer2016::MAX\_ERROR\_OMEGA (C++ member), 110  
 (C++ member), 110  
 SPH::Viscosity\_Peer2016::MAX\_ERROR\_V (C++ member), 110  
 (C++ member), 110  
 SPH::Viscosity\_Peer2016::MAX\_ITERATIONS\_OMEGA (C++ member), 112  
 (C++ member), 110  
 SPH::Viscosity\_Peer2016::MAX\_ITERATIONS\_V (C++ member), 112  
 (C++ member), 110  
 SPH::Viscosity\_Peer2016::performNeighborhoodSearch (C++ function), 112  
 (C++ function), 109  
 SPH::Viscosity\_Peer2016::reset (C++ function), 109  
 (C++ function), 109  
 SPH::Viscosity\_Peer2016::Solver (C++ type), 110  
 (C++ type), 110  
 SPH::Viscosity\_Peer2016::step (C++ function), 109  
 (C++ function), 109  
 SPH::Viscosity\_Peer2016::Viscosity\_Peer2016 (C++ function), 112  
 (C++ function), 109  
 SPH::Viscosity\_Standard (C++ class), 111  
 SPH::Viscosity\_Standard::~~Viscosity\_Standard (C++ function), 113  
 (C++ function), 111  
 SPH::Viscosity\_Standard::initParameters (C++ function), 111  
 (C++ function), 111  
 SPH::Viscosity\_Standard::m\_boundaryViscosity (C++ member), 114  
 (C++ member), 111  
 SPH::Viscosity\_Standard::reset (C++ function), 111  
 (C++ function), 111  
 SPH::Viscosity\_Standard::step (C++ function), 111  
 (C++ function), 111  
 SPH::Viscosity\_Standard::VISCOSITY\_COEFFICIENT (C++ member), 114  
 (C++ member), 111  
 SPH::Viscosity\_Standard::Viscosity\_Standard (C++ member), 114  
 (C++ function), 111  
 SPH::Viscosity\_Takahashi2015 (C++ class), 112  
 SPH::Viscosity\_Takahashi2015::~~Viscosity\_Takahashi2015 (C++ function), 112  
 (C++ function), 112  
 SPH::Viscosity\_Takahashi2015::computeViscosity (C++ function), 114  
 (C++ function), 114  
 SPH::Viscosity\_Takahashi2015::initParameters (C++ function), 113  
 (C++ function), 113  
 SPH::Viscosity\_Takahashi2015::ITERATIONS (C++ member), 112  
 (C++ member), 112  
 SPH::Viscosity\_Takahashi2015::m\_accel (C++ member), 113  
 (C++ member), 113  
 SPH::Viscosity\_Takahashi2015::m\_iterations (C++ member), 113  
 (C++ member), 113  
 SPH::Viscosity\_Takahashi2015::m\_maxError (C++ member), 113  
 (C++ member), 113  
 SPH::Viscosity\_Takahashi2015::m\_maxIter (C++ member), 113  
 (C++ member), 113  
 SPH::Viscosity\_Takahashi2015::m\_solver (C++ member), 113  
 (C++ member), 113  
 SPH::Viscosity\_Takahashi2015::m\_viscousStress (C++ member), 113  
 (C++ member), 113  
 SPH::Viscosity\_Takahashi2015::matrixVecProd (C++ function), 112  
 (C++ function), 112  
 SPH::Viscosity\_Takahashi2015::MAX\_ERROR (C++ member), 112  
 (C++ member), 112  
 SPH::Viscosity\_Takahashi2015::MAX\_ITERATIONS (C++ member), 112  
 (C++ member), 112  
 SPH::Viscosity\_Takahashi2015::performNeighborhoodSearch (C++ function), 112  
 (C++ function), 112  
 SPH::Viscosity\_Takahashi2015::reset (C++ function), 112  
 (C++ function), 112  
 SPH::Viscosity\_Takahashi2015::Solver (C++ type), 113  
 (C++ type), 113  
 SPH::Viscosity\_Takahashi2015::step (C++ function), 112  
 (C++ function), 112  
 SPH::Viscosity\_Takahashi2015::Viscosity\_Takahashi2015 (C++ function), 112  
 (C++ function), 112  
 SPH::Viscosity\_Weiler2018 (C++ class), 113  
 SPH::Viscosity\_Weiler2018::~~Viscosity\_Weiler2018 (C++ function), 113  
 (C++ function), 113  
 SPH::Viscosity\_Weiler2018::initParameters (C++ function), 114  
 (C++ function), 114  
 SPH::Viscosity\_Weiler2018::ITERATIONS (C++ member), 114  
 (C++ member), 114  
 SPH::Viscosity\_Weiler2018::m\_boundaryViscosity (C++ member), 114  
 (C++ member), 114  
 SPH::Viscosity\_Weiler2018::m\_iterations (C++ member), 114  
 (C++ member), 114  
 SPH::Viscosity\_Weiler2018::m\_maxError (C++ member), 114  
 (C++ member), 114  
 SPH::Viscosity\_Weiler2018::m\_maxIter (C++ member), 114  
 (C++ member), 114  
 SPH::Viscosity\_Weiler2018::m\_solver (C++ member), 114  
 (C++ member), 114  
 SPH::Viscosity\_Weiler2018::m\_vDiff (C++ member), 114  
 (C++ member), 114  
 SPH::Viscosity\_Weiler2018::matrixVecProd (C++ function), 114  
 (C++ function), 114

SPH::Viscosity\_Weiler2018::MAX\_ERROR (C++ member), 114

SPH::Viscosity\_Weiler2018::MAX\_ITERATIONS (C++ member), 114

SPH::Viscosity\_Weiler2018::performNeighborhoodSearch (C++ function), 113

SPH::Viscosity\_Weiler2018::reset (C++ function), 113

SPH::Viscosity\_Weiler2018::Solver (C++ type), 114

SPH::Viscosity\_Weiler2018::step (C++ function), 113

SPH::Viscosity\_Weiler2018::VISCOSITY\_COEFFICIENT\_BOUNDARY (C++ member), 114

SPH::Viscosity\_Weiler2018::Viscosity\_Weiler2018 (C++ function), 113

SPH::Viscosity\_XSPH (C++ class), 115

SPH::Viscosity\_XSPH::~~Viscosity\_XSPH (C++ function), 115

SPH::Viscosity\_XSPH::initParameters (C++ function), 115

SPH::Viscosity\_XSPH::m\_boundaryViscosity (C++ member), 115

SPH::Viscosity\_XSPH::reset (C++ function), 115

SPH::Viscosity\_XSPH::step (C++ function), 115

SPH::Viscosity\_XSPH::VISCOSITY\_COEFFICIENT\_BOUNDARY (C++ member), 115

SPH::Viscosity\_XSPH::Viscosity\_XSPH (C++ function), 115

SPH::ViscosityBase (C++ class), 116

SPH::ViscosityBase::~~ViscosityBase (C++ function), 116

SPH::ViscosityBase::initParameters (C++ function), 116

SPH::ViscosityBase::m\_viscosity (C++ member), 116

SPH::ViscosityBase::VISCOSITY\_COEFFICIENT\_BOUNDARY (C++ member), 116

SPH::ViscosityBase::ViscosityBase (C++ function), 116

SPH::ViscosityMethods (C++ enum), 130

SPH::ViscosityMethods::Bender2017 (C++ enumerator), 130

SPH::ViscosityMethods::None (C++ enumerator), 130

SPH::ViscosityMethods::NumViscosityMethods (C++ enumerator), 130

SPH::ViscosityMethods::Peer2015 (C++ enumerator), 130

SPH::ViscosityMethods::Peer2016 (C++ enumerator), 130

SPH::ViscosityMethods::Standard (C++ enumerator), 130

SPH::ViscosityMethods::Takahashi2015 (C++ enumerator), 130

SPH::ViscosityMethods::Weiler2018 (C++ enumerator), 130

SPH::ViscosityMethods::XSPH (C++ enumerator), 130

SPH::VorticityBase (C++ class), 117

SPH::VorticityBase::~~VorticityBase (C++ function), 117

SPH::VorticityBase::initParameters (C++ function), 117

SPH::VorticityBase::m\_vorticityCoeff (C++ member), 117

SPH::VorticityBase::VORTICITY\_COEFFICIENT (C++ member), 117

SPH::VorticityBase::VorticityBase (C++ function), 117

SPH::VorticityConfinement (C++ class), 118

SPH::VorticityConfinement::~~VorticityConfinement (C++ function), 118

SPH::VorticityConfinement::m\_normOmega (C++ member), 118

SPH::VorticityConfinement::m\_omega (C++ member), 118

SPH::VorticityConfinement::performNeighborhoodSearch (C++ function), 118

SPH::VorticityConfinement::reset (C++ function), 118

SPH::VorticityConfinement::step (C++ function), 118

SPH::VorticityConfinement::VorticityConfinement (C++ function), 118

SPH::VorticityMethods (C++ enum), 130

SPH::VorticityMethods::Micropolar (C++ enumerator), 130

SPH::VorticityMethods::None (C++ enumerator), 130

SPH::VorticityMethods::NumVorticityMethods (C++ enumerator), 130

SPH::VorticityMethods::VorticityConfinement (C++ enumerator), 130

SPH::WendlandQuinticC2Kernel (C++ class), 118

SPH::WendlandQuinticC2Kernel2D (C++ class), 119

SPH::WendlandQuinticC2Kernel2D::getRadius (C++ function), 119

SPH::WendlandQuinticC2Kernel2D::gradW (C++ function), 119

SPH::WendlandQuinticC2Kernel2D::m\_k (C++ member), 120

SPH::WendlandQuinticC2Kernel2D::m\_l (C++ member), 120

SPH::WendlandQuinticC2Kernel2D::m\_radius Utilities::SceneLoader::BoundaryData  
(C++ member), 120 (C++ struct), 33, 121

SPH::WendlandQuinticC2Kernel2D::m\_W\_zero Utilities::SceneLoader::BoundaryData::color  
(C++ member), 120 (C++ member), 34, 121

SPH::WendlandQuinticC2Kernel2D::setRadius Utilities::SceneLoader::BoundaryData::density  
(C++ function), 119 (C++ member), 33, 121

SPH::WendlandQuinticC2Kernel2D::W (C++ Utilities::SceneLoader::BoundaryData::dynamic  
function), 119 (C++ member), 33, 121

SPH::WendlandQuinticC2Kernel2D::W\_zero Utilities::SceneLoader::BoundaryData::isWall  
(C++ function), 119 (C++ member), 33, 121

SPH::WendlandQuinticC2Kernel::getRadius Utilities::SceneLoader::BoundaryData::mapFile  
(C++ function), 119 (C++ member), 34, 121

SPH::WendlandQuinticC2Kernel::gradW Utilities::SceneLoader::BoundaryData::mapInvert  
(C++ function), 119 (C++ member), 34, 122

SPH::WendlandQuinticC2Kernel::m\_k (C++ Utilities::SceneLoader::BoundaryData::mapResolution  
member), 119 (C++ member), 34, 122

SPH::WendlandQuinticC2Kernel::m\_l (C++ Utilities::SceneLoader::BoundaryData::mapThickness  
member), 119 (C++ member), 34, 122

SPH::WendlandQuinticC2Kernel::m\_radius Utilities::SceneLoader::BoundaryData::meshFile  
(C++ member), 119 (C++ member), 33, 121

SPH::WendlandQuinticC2Kernel::m\_W\_zero Utilities::SceneLoader::BoundaryData::rigidBody  
(C++ member), 119 (C++ member), 34, 121

SPH::WendlandQuinticC2Kernel::setRadius Utilities::SceneLoader::BoundaryData::rotation  
(C++ function), 119 (C++ member), 33, 121

SPH::WendlandQuinticC2Kernel::W (C++ Utilities::SceneLoader::BoundaryData::samplesFile  
function), 119 (C++ member), 33, 121

SPH::WendlandQuinticC2Kernel::W\_zero Utilities::SceneLoader::BoundaryData::samplingMode  
(C++ function), 119 (C++ member), 34, 122

SystemMatrixType (C++ type), 148 Utilities::SceneLoader::BoundaryData::scale  
(C++ member), 33, 121

## U

Utilities::SceneLoader::BoundaryData::translation  
(C++ member), 33, 121

USE\_BLOCKDIAGONAL\_PRECONDITIONER (C Utilities::SceneLoader::Box (C++ struct),  
macro), 145 34, 122

USE\_WARMSTART (C macro), 145 Utilities::SceneLoader::Box::m\_maxX  
(C++ member), 34, 122

USE\_WARMSTART\_V (C macro), 145 Utilities::SceneLoader::Box::m\_minX  
(C++ member), 34, 122

Utilities::SceneLoader (C++ class), 120 Utilities::SceneLoader::EmitterData  
(C++ struct), 33, 121 (C++ struct), 35, 122

Utilities::SceneLoader::AnimationFieldData Utilities::SceneLoader::EmitterData::emitEndTime  
(C++ member), 33, 121 (C++ member), 35, 122

Utilities::SceneLoader::AnimationFieldData::endTime Utilities::SceneLoader::EmitterData::emitStartTime  
(C++ member), 33, 121 (C++ member), 35, 122

Utilities::SceneLoader::AnimationFieldData::particleFrequency Utilities::SceneLoader::EmitterData::height  
(C++ member), 33, 121 (C++ member), 35, 122

Utilities::SceneLoader::AnimationFieldData::rotation Utilities::SceneLoader::EmitterData::id  
(C++ member), 33, 121 (C++ member), 35, 122

Utilities::SceneLoader::AnimationFieldData::scale Utilities::SceneLoader::EmitterData::rotation  
(C++ member), 33, 121 (C++ member), 35, 122

Utilities::SceneLoader::AnimationFieldData::shapeType Utilities::SceneLoader::EmitterData::type  
(C++ member), 33, 121 (C++ member), 35, 122

Utilities::SceneLoader::AnimationFieldData::startTime Utilities::SceneLoader::EmitterData::velocity  
(C++ member), 33, 121 (C++ member), 35, 122

Utilities::SceneLoader::AnimationFieldData::x Utilities::SceneLoader::EmitterData::velocity  
(C++ member), 33, 121 (C++ member), 35, 122

Utilities::SceneLoader::EmitterData::width (C++ member), 35, 122  
 Utilities::SceneLoader::EmitterData::x (C++ member), 35, 122  
 Utilities::SceneLoader::FluidBlock (C++ struct), 35, 122  
 Utilities::SceneLoader::FluidBlock::box (C++ member), 35, 122  
 Utilities::SceneLoader::FluidBlock::id (C++ member), 35, 122  
 Utilities::SceneLoader::FluidBlock::initValVelocity (C++ member), 35, 122  
 Utilities::SceneLoader::FluidBlock::mode (C++ member), 35, 122  
 Utilities::SceneLoader::FluidData (C++ struct), 36, 122  
 Utilities::SceneLoader::FluidData::id (C++ member), 36, 123  
 Utilities::SceneLoader::FluidData::initialVelocity (C++ member), 36, 123  
 Utilities::SceneLoader::FluidData::invert (C++ member), 36, 123  
 Utilities::SceneLoader::FluidData::mode (C++ member), 36, 123  
 Utilities::SceneLoader::FluidData::resolution (C++ member), 36, 123  
 Utilities::SceneLoader::FluidData::rotation (C++ member), 36, 123  
 Utilities::SceneLoader::FluidData::samples (C++ member), 36, 123  
 Utilities::SceneLoader::FluidData::scale (C++ member), 36, 123  
 Utilities::SceneLoader::FluidData::translation (C++ member), 36, 123  
 Utilities::SceneLoader::m\_jsonData (C++ member), 121  
 Utilities::SceneLoader::MaterialData (C++ struct), 36, 123  
 Utilities::SceneLoader::MaterialData::colorField (C++ member), 37, 123  
 Utilities::SceneLoader::MaterialData::colorMapType (C++ member), 37, 123  
 Utilities::SceneLoader::MaterialData::emitterBoxMax (C++ member), 37, 123  
 Utilities::SceneLoader::MaterialData::emitterBoxMin (C++ member), 37, 123  
 Utilities::SceneLoader::MaterialData::emitterReuseParameters (C++ member), 37, 123  
 Utilities::SceneLoader::MaterialData::id (C++ member), 37, 123  
 Utilities::SceneLoader::MaterialData::maxEmitterParticles (C++ member), 37, 123  
 Utilities::SceneLoader::MaterialData::maxVal (C++ member), 37, 123  
 Utilities::SceneLoader::MaterialData::minVal (C++ member), 37, 123  
 Utilities::SceneLoader::readMaterialParameterObject (C++ function), 120  
 Utilities::SceneLoader::readParameterObject (C++ function), 120, 121  
 Utilities::SceneLoader::readScene (C++ function), 120  
 Utilities::SceneLoader::readValue (C++ function), 120, 121  
 Utilities::SceneLoader::readVector (C++ function), 120  
 Utilities::SceneLoader::Scene (C++ struct), 37, 123  
 Utilities::SceneLoader::Scene::animatedFields (C++ member), 37, 123  
 Utilities::SceneLoader::Scene::boundaryModels (C++ member), 37, 123  
 Utilities::SceneLoader::Scene::camLookat (C++ member), 37, 124  
 Utilities::SceneLoader::Scene::camPosition (C++ member), 37, 123  
 Utilities::SceneLoader::Scene::emitters (C++ member), 37, 123  
 Utilities::SceneLoader::Scene::fluidBlocks (C++ member), 37, 123  
 Utilities::SceneLoader::Scene::fluidModels (C++ member), 37, 123  
 Utilities::SceneLoader::Scene::materials (C++ member), 37, 123  
 Utilities::SceneLoader::Scene::particleRadius (C++ member), 37, 123  
 Utilities::SceneLoader::Scene::sim2D (C++ member), 37, 123  
 Utilities::SceneLoader::Scene::timeStepSize (C++ member), 37, 123  
 Utilities::SDFFunctions (C++ class), 124  
 Utilities::SDFFunctions::computeBoundingBox (C++ function), 124  
 Utilities::SDFFunctions::distance (C++ function), 124  
 Utilities::SDFFunctions::generateSDF (C++ function), 124  
 Utilities::VolumeSampling (C++ class), 124  
 Utilities::VolumeSampling::sampleMesh (C++ function), 125  
 Utilities::WindingNumbers (C++ class), 125  
 Utilities::WindingNumbers::computeGeneralizedWindingNumbers (C++ function), 125  
 Vector2i (C++ type), 149  
 Vector2r (C++ type), 149  
 Vector3f (C++ type), 149

Vector3f8 (C++ *class*), 126  
Vector3f8::blend (C++ *function*), 127  
Vector3f8::cross (C++ *function*), 126  
Vector3f8::dot (C++ *function*), 126  
Vector3f8::norm (C++ *function*), 126  
Vector3f8::normalize (C++ *function*), 126  
Vector3f8::operator\* (C++ *function*), 126  
Vector3f8::operator\*= (C++ *function*), 126  
Vector3f8::operator/ (C++ *function*), 126  
Vector3f8::operator/= (C++ *function*), 126  
Vector3f8::operator% (C++ *function*), 126  
Vector3f8::operator- (C++ *function*), 126  
Vector3f8::operator-= (C++ *function*), 126  
Vector3f8::operator[] (C++ *function*), 126  
Vector3f8::setZero (C++ *function*), 126  
Vector3f8::squaredNorm (C++ *function*), 126  
Vector3f8::store (C++ *function*), 127  
Vector3f8::v (C++ *member*), 127  
Vector3f8::Vector3f8 (C++ *function*), 126  
Vector3f8::x (C++ *function*), 126  
Vector3f8::y (C++ *function*), 126  
Vector3f8::z (C++ *function*), 126  
Vector3r (C++ *type*), 149  
Vector4f (C++ *type*), 149  
Vector4r (C++ *type*), 150  
Vector5r (C++ *type*), 150  
Vector6r (C++ *type*), 150