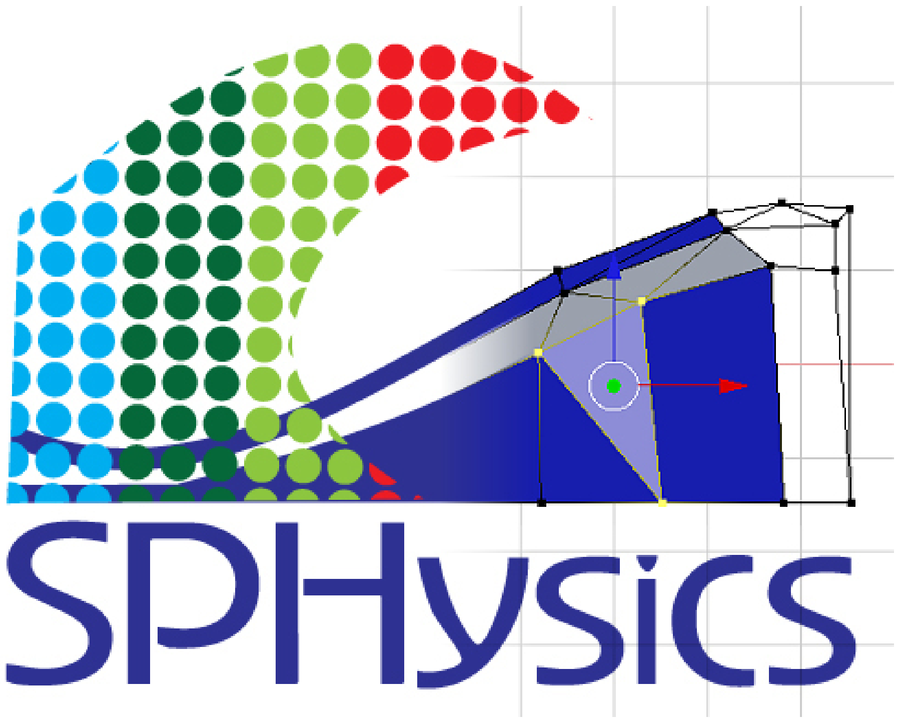
Combining Blender with SPHysics, an Introduction



July 2010

Arno Mayrhofer (`arnom@amconception.de`)
Moncho Gomez-Gesteira (`mggesteira@uvigo.es`)
Alejandro J.C. Crespo (`alexbexe@uvigo.es`)
Benedict D. Rogers (`benedict.rogers@manchester.ac.uk`)

Abstract

This paper gives an introduction on how to use Blender with SPHysics. For this purpose a step-by-step tutorial will show how to implement a typical dam break scenario. Further information will be given on the various new options implemented in the Blender SPHysics combination. Finally several advanced topics will be described in the Appendix.

Acknowledgements

The authors wish to acknowledge the support of the following projects.

- PGIDIT06PXIB383285PR
- ”Grupos de Referencia Competitiva” Xunta de Galicia (2006/136)
- School of Mechanical, Aerospace and Civil Engineering Course on Smoothed Particle Hydrodynamics (SPH) at the University of Manchester, UK

Contents

1 Introduction

The main goal of this paper is to show how it is possible to combine Blender (`http://www.blender.org`) and SPHysics (`http://www.sphysics.org`).

For a general introduction to SPHysics cf. to [1]. The pre-processing routines presented in this paper are in detail explained in [2]. Please cite the last paper when using this software (see also Appendix D).

Blender is a very powerful 3D tool which is released under the open source GPL license. The main features

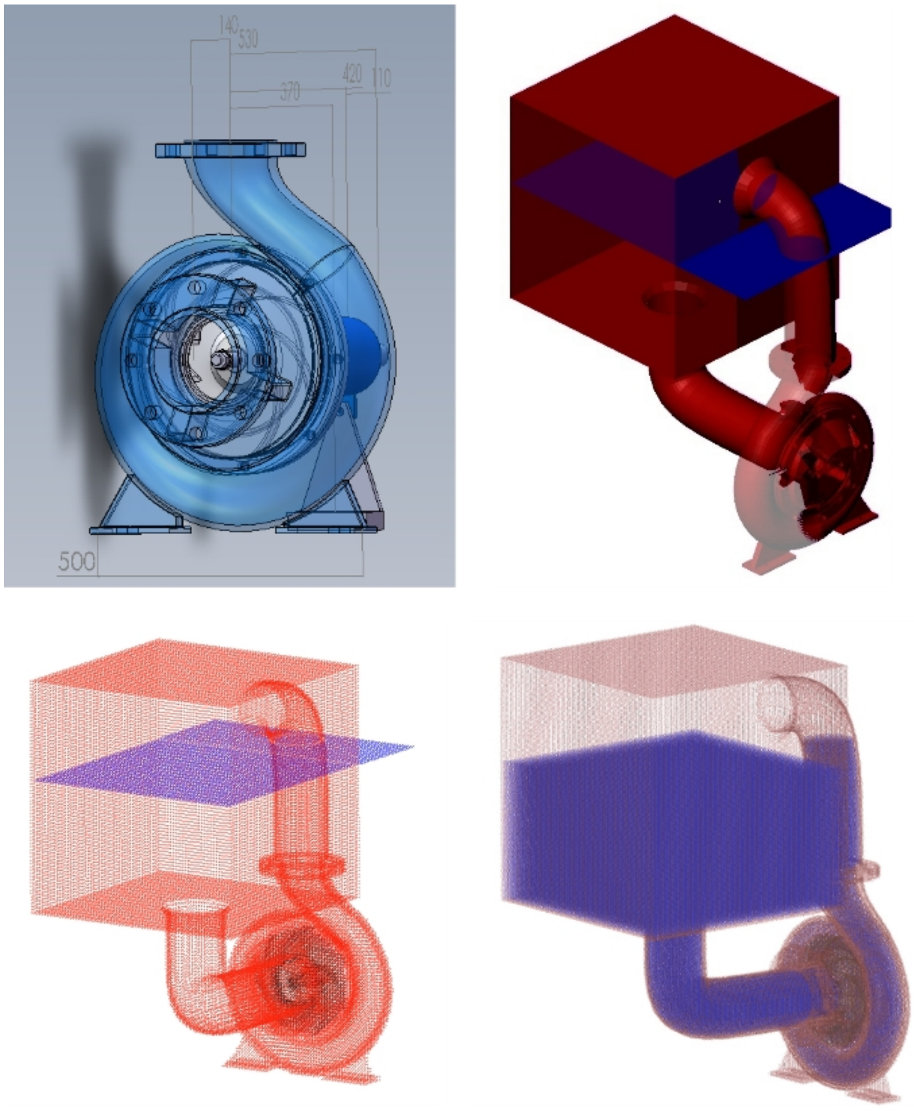


Figure 1: A little demonstration of power.

that led to the use of Blender are as follows:

- Python API
- easy editing of mesh objects
- compability with many file formats

- open source.

In Blender it is possible to create highly complex shapes which are commonly encountered in engineering, design and simulation. An example is shown in Figure 1 which shows a pump with the particles generated using the method described herein. Utilizing the Python API it is possible to create an export script that converts a general mesh object into a point cloud which can be used by SPHysicsGen to create initial data for SPHysics.
The following sections are organised according to the real workflow. In short it can be described by the following steps:

- Blender:
  - creation of a mesh
  - coloring according to particle type
  - exporting to a point cloud file
- SPHysicsGen:
  - choosing the new geometry option
  - filling the scenery
  - specifying the moving boundary behaviour

In the appendix an additional step will be described which is the creation of a mesh that describes a real bathymetry using geographic data.

2 Blender

Since the installation of Blender (Version 2.49) is not very complicated this step will be omitted and a working installation is assumed. The Python export script has been written in Python 2.6 (`http://www.python.org`) which has to be installed too. The reader is also advised to visit the Blender website (`http://www.blender.org`) and to take a look at the excellent documentation there. Furthermore a forum can be found there as well as links to other community pages that should be able to help with any problems. Please note that the authors and contributors of SPHysics are not able to give support for the use of Blender.

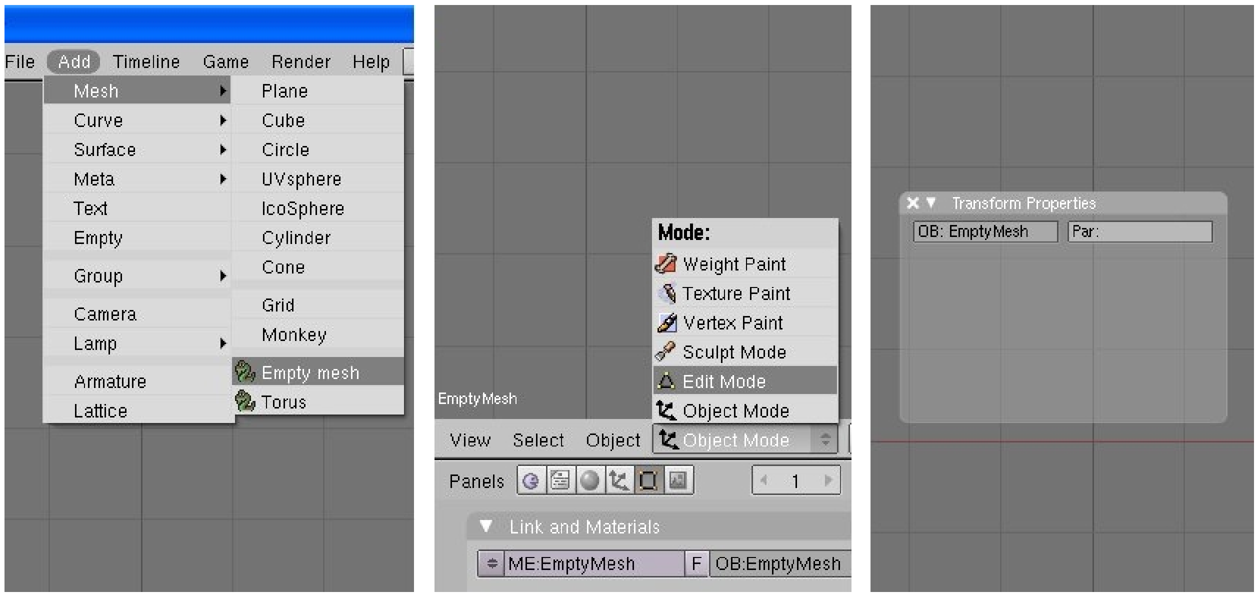
Blender 2.5/2.6: At the time of writing Blender 2.5 is in Alpha 2. Hence, the export script has not yet been ported to the new version of Blender. The Python API has changed significantly with this version and adapting the script will not be straightforward. As soon as Blender 2.6 is released porting should be considered and any help to do so is greatly appreciated.

2.1 Creating a Mesh

In the following sections we will give a tutorial on how to setup a typical 3-D dam break text case (Case 9 of the standard SPHysics distribution). In **run_directory/Case9** you will be able to find all required files. However it is advisable to create the Blender file yourself for a better understanding.

After Blender is started a simple cube should be visible from top view. Pressing **Del** and clicking on **Erase selected object(s)** will get rid of it since it is not of any use for the mesh. To create a new (empty) mesh object go to **Add→Mesh→Empty mesh** and switch to Edit Mode as displayed in Figure 2(b) (or by pressing **TAB**). It also is advisable to press **N**[1] to make the 'Transform Properties' window visible.
A general mesh consists of vertices. They can be connected in two ways:

[1]This only refers to the letter **N** and not the combination **SHIFT + N**



(a) Creating a new mesh (b) Selecting mode (c) Property dialogue box for editing

Figure 2: Setting up an empty mesh and switching to edit mode

- 2 vertices can be connected via an edge
- 3 or 4 vertices can be connected via a face.

2.1.1 Creating Vertices

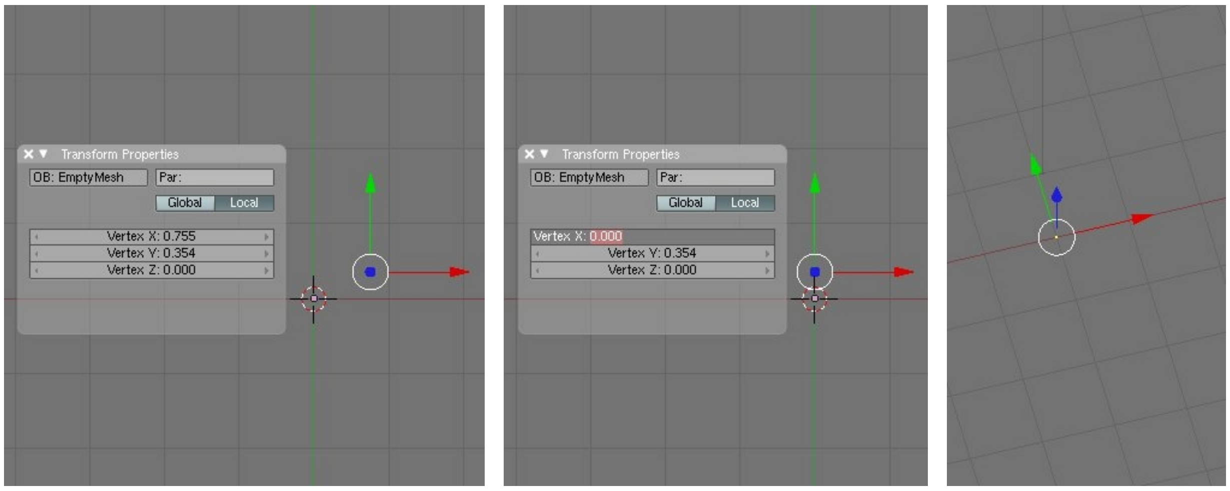
To create the first vertex press **Ctrl** and **left-mouse** at the desired position. Note that the 'Transform Properties' window displays the current position of the vertex. Clicking on 'Vertex X: *Value*' toggles user input to enable accurate positioning of the vertex (Clicking on the arrows on either side decreases/increases the value). Confirm the input by pressing **Return** and the vertex will be moved to the new coordinates (see Figure 3(a) & 3(b)). Move this vertex to $(4, 0, 0)$.

Before creating the next vertex, a short word on changing the viewpoint in the 3D window. All movements necessary can be achieved with the numeric keyboard (numpad) or by the **View** menu. Press **num 2**[2] and **num 4** (To rotate down and left). Now for the first time it is possible to actually see the vertex itself, it is the small yellow dot inside the white circle at the centre of the red to blue-green arrows (Figure 3(c)).

The general keys for movement of the viewpoint are:

- **num 2** - down
- **num 4** - left
- **num 6** - right
- **num 8** - up
- **num +** - zoom in
- **num -** - zoom out

[2]**num 2** refers to the number two on the numeric keyboard


Transform Properties
OB: EmptyMesh Par:
Global Local
Vertex X: 0.755
Vertex Y: 0.354
Vertex Z: 0.000

Transform Properties
OB: EmptyMesh Par:
Global Local
Vertex X: 0.000
Vertex Y: 0.354
Vertex Z: 0.000

(a) Creating a vertex (b) Moving the vertex (c) Vertex visible as yellow dot

Figure 3: Creating and moving the first vertex and rotating the viewpoint

When additionally pressing **Ctrl** the viewpoint is moved instead of rotated.
Also helpful is the **num-5** key which toggles between orthographic and perspective view. For the purpose here the authors recommend the use of the orthographic view.
Now create a second vertex by again pressing **Ctrl + left-mouse**. It can be seen that the second vertex is already connected to the first one by an edge. This happened because the first vertex was still selected. Selected elements are displayed in yellow colour and at the current stage the second vertex should be selected whereas the first one is not (pink colour). The goal is to create a simple rectangular box with a gate inside to simulate a dam break, so move the second vertex to (4,4,0). Now create two more vertices at (0,4,0) and (0,0,0), the result is displayed in Figure 4(a).

2.1.2 Creating Edges & Faces

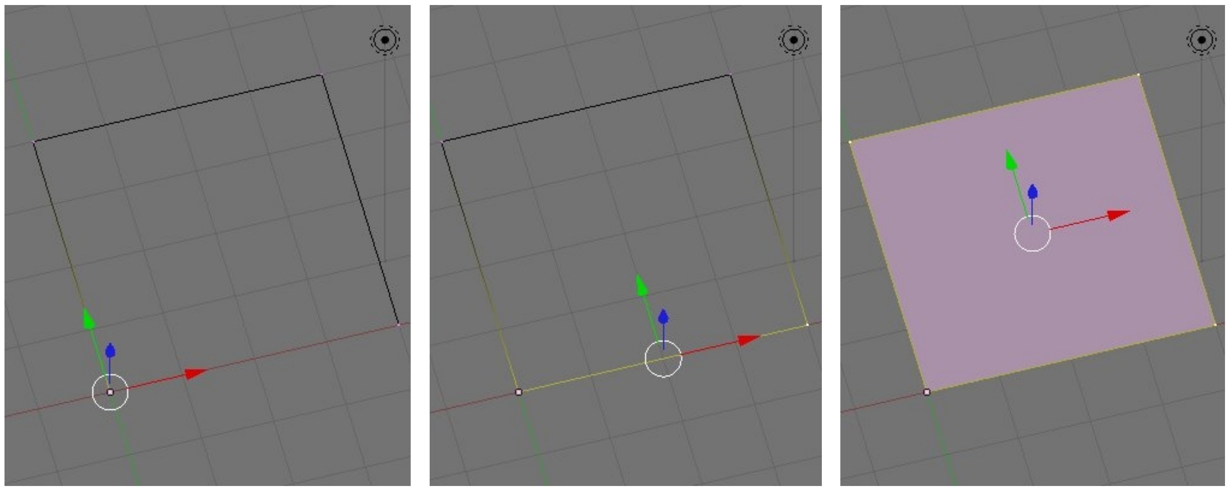

(a) Initial situation (b) Adding the 4th edge (c) Creating the face

Figure 4: Connecting two vertices via an edge and four vertices via a face

To create an edge between the vertices at (0,0,0) and (4,0,0) it is necessary to select them both. The vertex at (0,0,0) is already selected (Figure 4(a)) and to select the one at (4,0,0) hold **Shift** and click with

the **right-mouse** onto the vertex. To create an edge between them press **F** (see Figure 4(b)). After that add the two other vertices to the selection and press **F** again to create a face. The result can be seen in Figure 4(c).
To create the upper border of the box press **A** to deselect all (**A** toggles between select all and deselect all). Create four more vertices at (0,0,2),(4,0,2),(4,4,2) and (0,4,2) and connect the last two via an edge, but do not create a face and deselect all (see Figure 5(a)). Select (0,0,0) and (0,0,2) and create an edge between them, similarly for the remaining three corners see Figure 5(b).

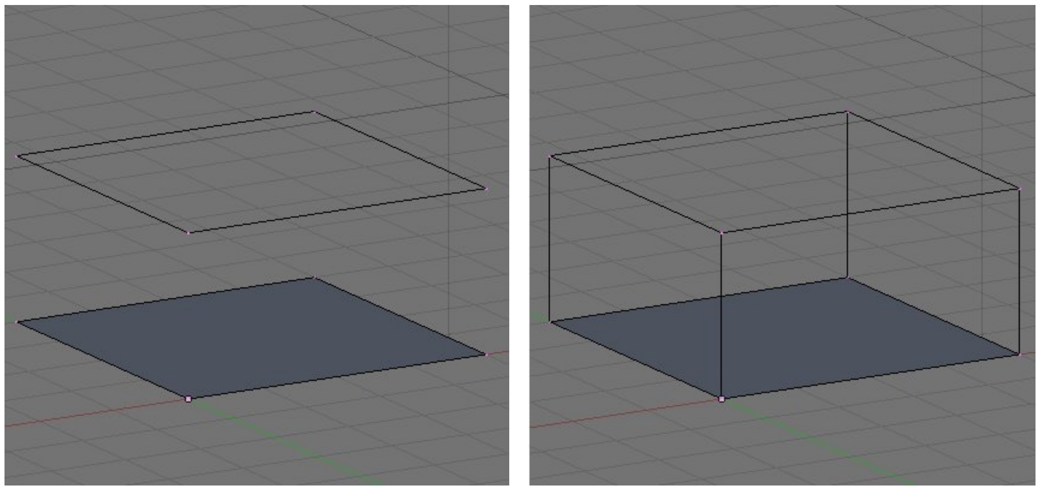

(a) Adding the top edges (b) Connecting to the bottom

Figure 5: Creating the edges at the corners

2.1.3 Creating Side Walls and the Gate

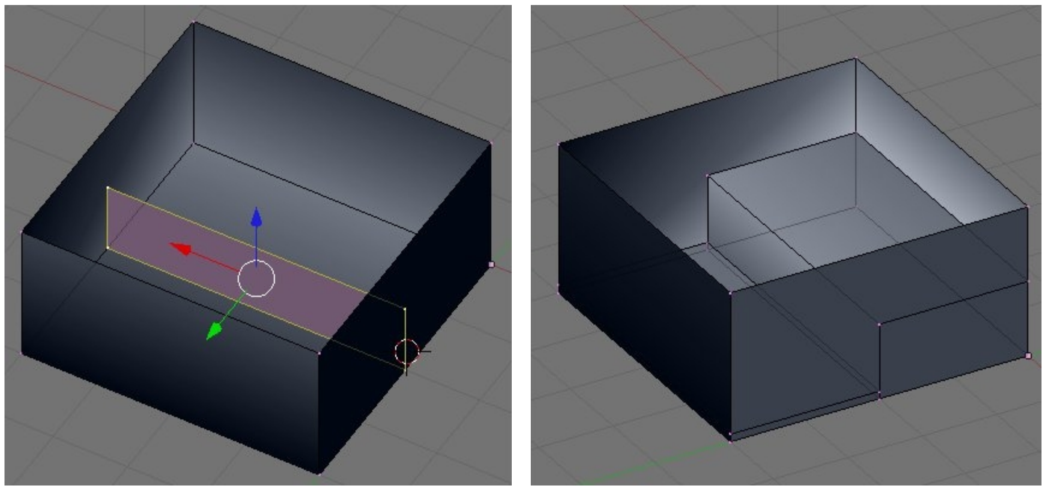

(a) Creating the gate (b) Adding water surfaces

Figure 6: Creating the gate and the water surface

Now select four vertices that belong to one side wall and create a face. Continue until all four walls are created. To create the gate, deselect all and create four vertices at (0,2,0),(4,2,0),(4,2,1) and (0,2,1), connect them via edges and create a face (see Figure 6(a)). Now all that is left to do is to specify the water *surfaces*. We aim to fill on the lefthand side of the gate and add a wet bed on the right. So create two

Table 1: Color Codes for Blender

	R(ed)	G(reen)	B(lue)	A(lpha)
Fixed Boundary	1	0	0	1
Fluid Surface	0	0	1	1
Moving Boundaries	[0,1[	0	0	1
Floating Objects	0	]0,1[	1	1
Periodic Boundaries	0	]0,1]	0	0.5

vertices as (0,0,1) and (4,0,1) and connect them to (0,2,1) and (4,2,1) via two edges and a face. To create the wet bed four vertices at (0,2,0.2),(4,2,0.2),(4,4,0.2) and (0,4,0.2) are needed once again connected by edges and a face. The final result can be seen in Figure 6(b).

As a last note, it is also possible to switch to different editing modes by pressing **Ctrl Tab** and then clicking on the desired mode (see also Figure 7).
This concludes the main geometry generation part but the function of each face (e.g. solid wall or water surface) is not yet defined.

2.2 Colouring the Faces

To define the function of each face (fixed or moving boundary, fluid, etc.) it is necessary to colour it. The colour codes are defined in Table 1. The type of movement for the moving boundary is later specified in SPHysicsGen.
To create the three required materials press **F5** to switch to the Shading tab (see also Figure 7) in the third window (note: the first window is the menu bar on top, the second is the 3D view and the third the property window at the bottom). Click **Add New** in the property window under "Links and Pipeline" and a screen similar to the one below will appear (Figure 7).
Change the color to red (R: 1, G: 0, B: 0) and rename it to 'MA:Red' as seen in Figure 7. Observe that

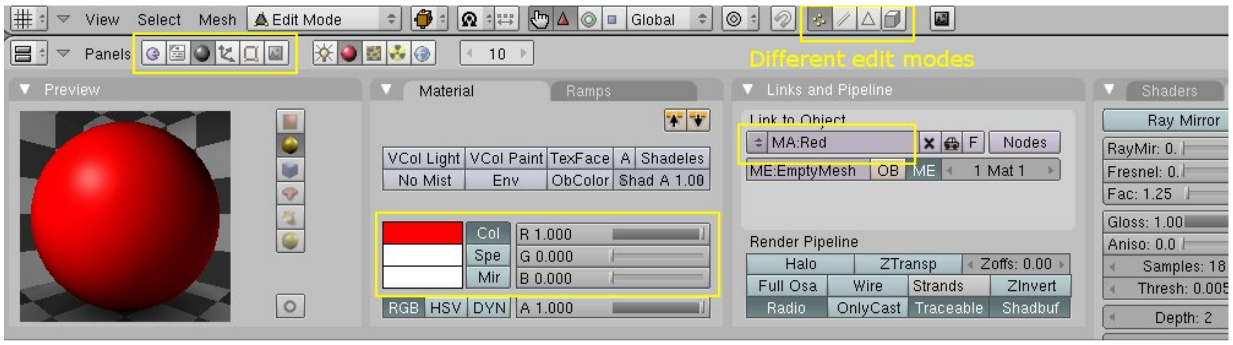
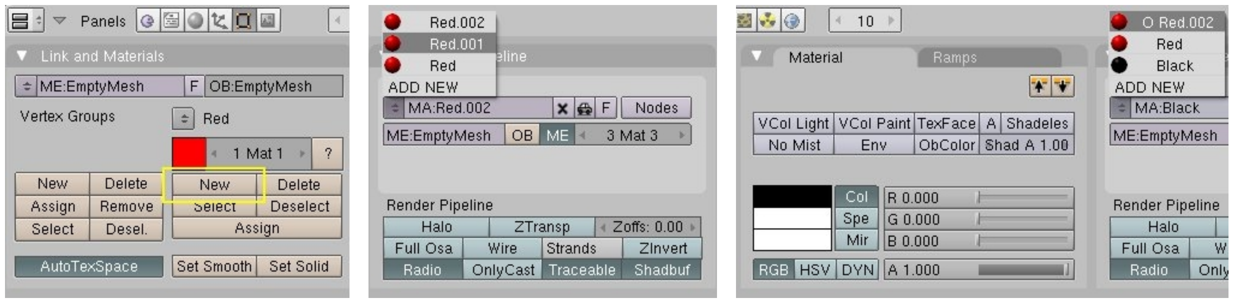


Figure 7: Creating the material for fixed boundaries

the object in the 3D view is now red. Switch back to the Editing tab (**F9**) and notice the change as seen in Figure 8(a).

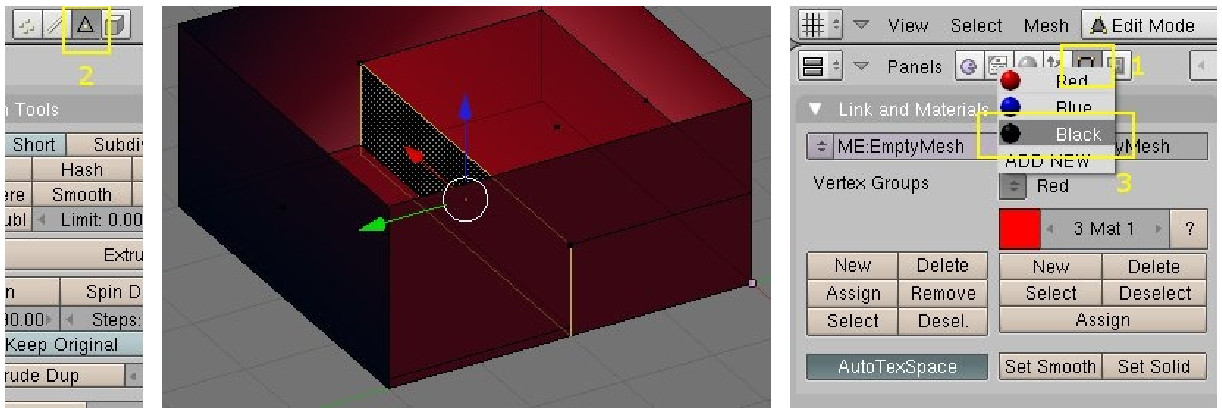
Create two more new materials for this 'Empty mesh' object by clicking twice on the highlighted **New** button. Switch back to the 'Shading tab' (**F5**) and select MA:Red.001 (Figure 8(b)) which then should be renamed to Black and reduce its red value to 0 to get (R: 0, G: 0, B:0) (see Figure 8(c)). Then select



(a) Updated "Links and Materials" (b) Selecting the second material (c) Creating the black material and selecting the third material

Figure 8: Connecting two vertices via an edge and four vertices via a face

MA:Red.002, rename it to Blue and change to colour (R: 0, G: 0, B: 1). Note that at all times it is necessary to have at least one blue and one red face.



(a) Switching to face edit mode (b) Selecting the face (c) Changing the face material

Figure 9: Coloring the gate blacke

Currently all our faces are red and so would be treated as fixed boundary particles (see Table 1). So first we want to color our gate face black since it will move. To do so switch back to the 'Editing tab' (**F9**) and select the 'face editing mode' (**Ctrl Tab**) in the second window (Figure 9(a)). Select the face gate by clicking **right-mouse** (Figure 9(b)) and then in the third window choose the black material (Figure 9(c)).

Do the same with both fluid faces but this time choose the blue material. Going back to 'Object Mode' now displays the final result as seen in Figure 10. Since it is necessary to save the model before exporting, do so now.

2.2.1 Floating Objects

Floating Objects can be any Blender Mesh object coloured appropriately (see Table 1). It is however important that the mesh forms a closed volume (see Appendix). This will not be checked by the Export Script and it will provide wrong calculations for Mass, Centre of Gravity and Moments of Inertia. Some sophisticated floating objects can be seen in Figure 11.

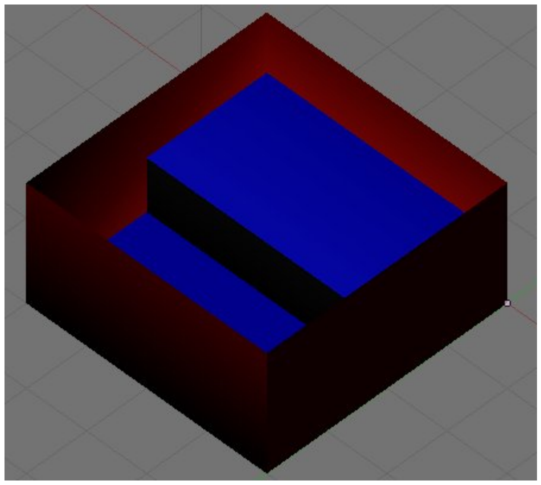


Figure 10: The model after coloring, ready to be exported

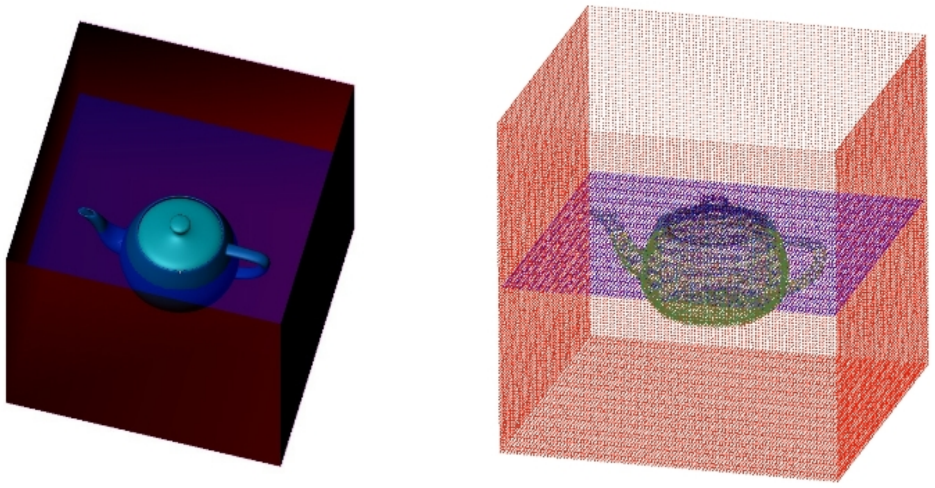


Figure 11: A Utah teapot as floating object

2.2.2 Periodic Boundaries

Periodic Open Boundaries are currently only implemented in y-direction. As shown in Table 1 they are usually coloured green with an alpha value of 0.5. Until SPHysics allows for more complicated Periodic Boundaries the following requirements have to be met:

- Normals must be identical to (0,1,0)
- Each face of the periodic boundary must have a twin on the other side (i.e. directly opposite and same size)
- Periodic faces must lie at the very outside with respect to y.

An example of a periodic open boundary can be seen in Fig. 12. Note that it is not required that the seabed agrees on either side of the boundary, a special algorithm in SPHysicsGen should take care of any possible leaks.

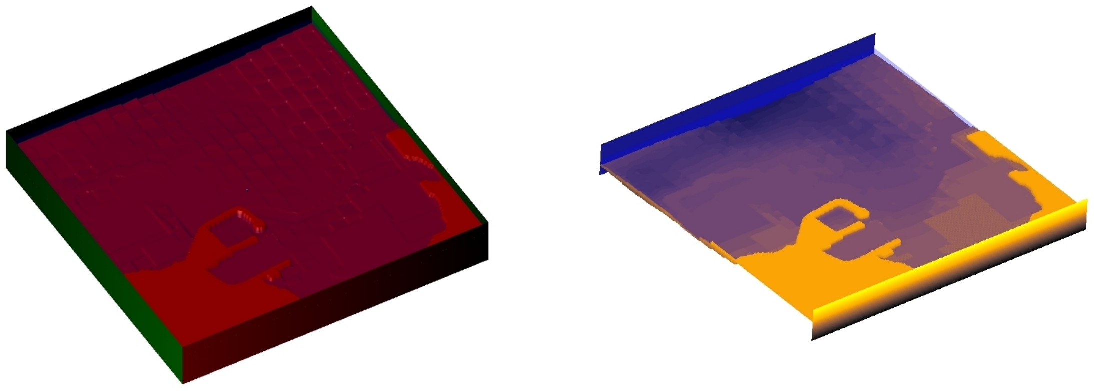


Figure 12: Real bathymetry with periodic open boundaries

2.2.3 An important operation

This section might answer your question why the output does not look as it was in Blender.
Lets start with explaining a mesh in Blender. First of all, everything you can create in Blender is an Object which has properties such as position of the centre, rotation and scale. Obviously a Mesh is also an Object and when you are editing one vertex you'll see that it has too different coordinates. The first one is the global coordinate, whereas the second is the local coordinate. The relation between the local and the global coordinate is as follows. The local coordinate is the position measured with respect to the Object axis where the origin is the centre, the scaling and rotation of the axis is according to those of the Object. The global coordinate however is with respect to the World axis. So the local coordinates are different from Object to Object and since the export script has only access to the local coordinates this can cause troubles.
To avoid this we would need all Objects with centre at the World origin, no rotation and scaling equal to one. Luckily this can be achieved with only three commands executed in 'Object Mode'

1. **Object→Clear/Apply→Apply Scale/Rotation to ObData** (**Ctrl+A,1**): Resets Rotation and Scaling of the selected Object.

2. Press **Shift+C** to centre the cursor at the origin

3. **Object→Transform→Center Cursor**: Resets the centre of the selected Object to the Cursor which is at the World origin.

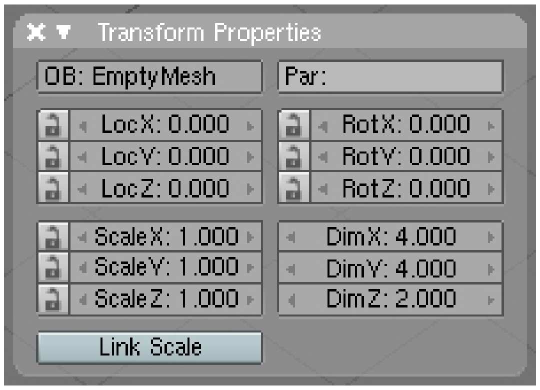
To check whether this has worked you can select any vertex in 'Edit Mode' and the local and global coordinates should be identical. Alternatively in object mode the object information in the Transform Properties window should look as in Figure 13.

2.3 Exporting the Mesh

2.3.1 Script Installation

Before the mesh can be exported for the first time, a python script has to be installed manually. The script can be found in the *source/blender* folder and is called *sphysics_export_interface.py*. It has to be copied (or symbolically linked) to one of the following directories[3] depending on the operating system

[3]In case none of the described directories work, open a Script Window in Blender as described avoe, go to **Script→System→Interactive Python Console** and execute the command *Blender.Get('scriptsdir')*. This should provide the location of the scripts. Then copy *sphysics_export_interface.py* to 'scriptsdir'



Transform Properties

OB: EmptyMesh Par:

LocX: 0.000 RotX: 0.000
LocY: 0.000 RotY: 0.000
LocZ: 0.000 RotZ: 0.000

ScaleX: 1.000 DimX: 4.000
ScaleY: 1.000 DimY: 4.000
ScaleZ: 1.000 DimZ: 2.000

Link Scale

Figure 13: Transform Properties window after resetting axis

and installation type. If Blender is installed then the following directories should be correct:

- Linux: ~/.blender/scripts/blender or ~/.blender/scripts/blender
- Windows XP: C:\Program Files\Blender Foundation\Blender\.blender\scripts
- Windows XP (old): C:\Documents and Settings\USERNAME\Application Data\Blender Foundation\Blender\.blender\scripts
- Windows Vista: C:\Users\USERNAME\AppData\Roaming\Blender Foundation\Blender\.blender\scripts

If Blender is only unzipped then the scripts directory can be found in

- All OS: PathToUnzipFolder/.blender/scripts

Then switch the second window (the 3-D View) to the 'Scripts Window' (click on the grid next to the 'View' menu in the lower left part and choose **Scripts Window**) and after that go to **Scripts→Update Menu** to load the script. This has to be done only once and normally only the following section has to be followed.

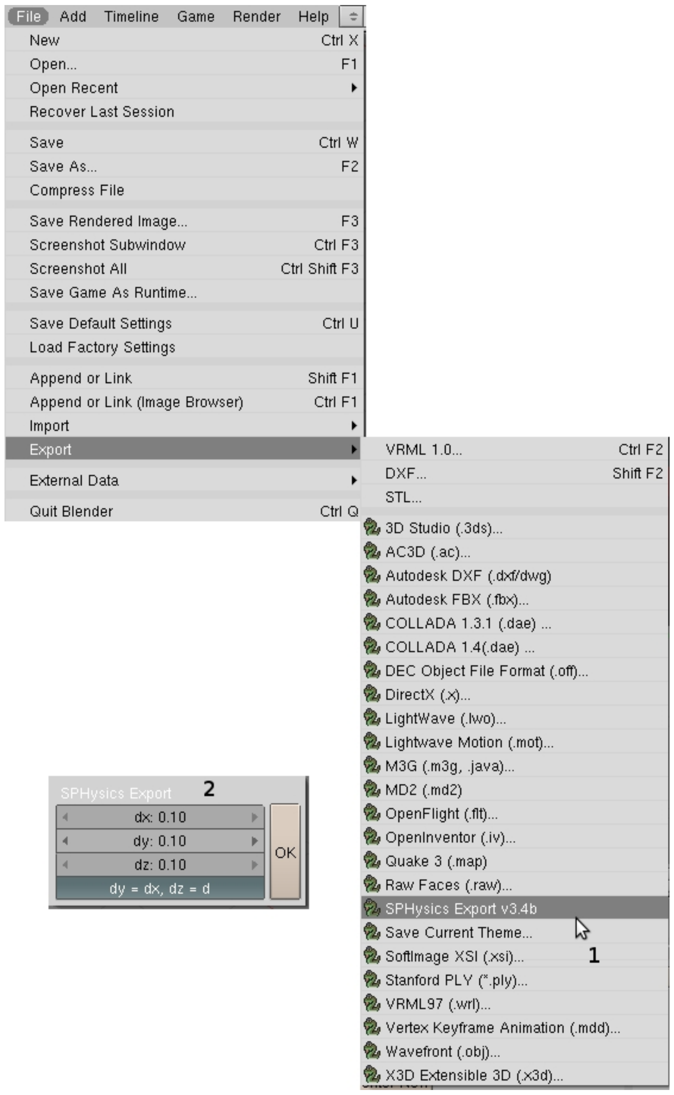
2.3.2 Exporting via Script

Before exporting there are two things that need to ensured:

1. Be sure to save the blender file.
2. The blender file must be in the correct directory. This means that it has to be in a folder inside *run_directory*[4] (here *run_directory/Case9*).

Once this is done export the blender file by executing the following steps as shown in Figure 14.

- Choose **File→Export→SPHysics Export**,
- enter the values for dx,dy,dz (for now the default values are just fine),
- click 'Ok',
- done.



dx: 0.10
dy: 0.10
dz: 0.10

Figure 14: Exporting the Blender File

If any errors occur they should show up in the Blender console.
In the input pop-up there is a button with caption **dy = dx, dz = dx** at the very bottom that is enabled by default. If it is not disabled the values of dy and dz will be identical to dx.
It is also possible to visualise the exported geometry. For this purpose please see Appendix B.

3 SPHysicsGen

The general guide to SPHysics [1] can be found on the official homepage (`http://www.sphysics.org`) and this section will only cover the differences to the normal approach. It is assumed that you have copied the executable to the directory with the Blender files. At this stage it would be possible to execute the corresponding **.bat** file to simulate the geometry. To explain the novelties in SPHysicsGen we will however show a detailed instruction on how to run it assuming the input in **Case9.txt**.

[4]As a matter of fact any directory will do as long as ../../ contains the *source* folder.

3.1 The New Geometry Option

The first changes are visible at the choice of the geometry. Here a fourth option, called **EXTERNAL POINT FILE**, has been introduced and it must be chosen in order to read in the *IPART_B* and *FO* files. Note that these files must be in the same directory as *sphysicsgen3d.exe*.

```
 Geometry of the zone
  (1) BOX
  (2) BEACH
  (3) EXTERNALLY FIXED GEOMETRY
  (4) EXTERNAL POINT FILE
           4
```

After entering the fluid lattice structure the program reads in *IPART_B* and an output similar to the following should show

```
 --- Generating Geometry from IPART_B ---

  Spacing dx,dy,dz
  0.10000000000000001       0.10000000000000001       0.10000000000000001
  Initial particle count
       6753
  3D Array size
          41          41          21
```

At this stage the boundary particles have already been initialised and the next step is the filling of the scenery.

3.1.1 The Filling Algorithms

How the algorithm works in detail is described in [2]. So we will only describe how it is used.
To initialise the filling algorithm you will have to specify a point that lies within the fluid. For the current example we will use (0.1,0.1,0.1) and additionally (3.5, 3.5,0.1). As seen below.

```
  Please provide seed point for filling (x,y,z)
  0.10000000      0.10000000      0.10000000
  Another seed point? (1 = yes, 0 = no)
   1.0000000
  Please provide seed point for filling (x,y,z)
   3.5000000       3.5000000      0.10000000
  Another seed point? (1 = yes, 0 = no)
   0.0000000
 Propsed Still Water Level:   1.0000000208616258
 Current Still Water Level:   2.0000000
 Replace current with proposed SWL? (0 = No, 1 = Yes)
           1
 h_SWL =    1.0000000
```

You can also see that due to the fact that we have entered 2 as h_SWL at the beginning the algorithm complains and wants to correct it to the appropriate value 1. You can allow this by typing 1 after which you will receive a confirmation that h_SWL is now indeed 1.000000.

3.2 Moving Boundary Behaviour

After all particles have been created the type of movement for the moving boundaries has to be defined. There are four different types of movement currently implemented. The dialog below shows an example situation which could be used for the dam break modelled above.

```
Select movement pattern for moving boundaries:
   (1) linear (e.g. gate)
   (2) sinusoidal (e.g. piston)
   (3) rotation (e.g. flap)
   (4) position file
           1
  Specify variables for linear movement:
  Enter velocity for moving boundaries:
  (ux, uy, uz)
  0.0000000E+00  0.0000000E+00 0.3000000
  Start and stop time for movement?
  0.0000000E+00   1.000000
```

It is also possible to define multiple moving objects. To do so use a different colour for each moving object. The allowed RGB values are: $(r, 0, 0)$ where $r \in [0, 1[$. As can be seen in the output above the program tells the user the first and last particle of this moving object in order to identify it. A different type of movement can be assigned for each boundary. The specific values for the movements is then stored in a new output file *MULT_MV*.
In the following subsections the four different types shall be examined in detail.

3.2.1 Constant Velocity

The only input for this is the velocity and the standard start and stop time. As soon as the object leaves the computational domain during runtime it is destroyed and sent to the trash. Examples for this type of movement include for example a gate or a ship.
Input:

- u_x ... velocity in x direction [m/s]
- u_y ... velocity in y direction [m/s]
- u_z ... velocity in z direction [m/s]
- t_0 ... start time of movement [s]
- t_1 ... stop time of movement [s]

3.2.2 Sinusoidal Movement

Mostly used for piston wavemakers these type of movement takes the following input.
Input:

- a ... amplitude [m]
- b ... period [s]
- c ... phase [degree]
- x ... direction of movement, x coordinate

- y ... direction of movement, y coordinate
- z ... direction of movement, z coordinate
- t_0 ... start time of movement [s]
- t_1 ... stop time of movement [s]

For the following movement types, the computational domain is calculated entirely in *SPHysicsGen*. The program evaluates all possible positions of the boundary particles and saves these results in *INDAT*. Movement continues only while time t is larger than t_0 and smaller than t_1.

3.2.3 Rotation Movement

This type of movement allows for a wide variety of simulations. The most common example is a simple flap wavemaker but also simulations for elliptic tank sloshing or rotating propellers are possible. To describe a general rotation in 3-D the following input is necessary.
Input:

- per ... period [s]
- min ... minimum elongation angle [degrees]
- max ... maximum elongation angle [degrees]
- c_x ... center of rotation, x coordinate [m]
- c_y ... center of rotation, y coordinate [m]
- c_z ... center of rotation, z coordinate [m]
- a_x ... rotation axis, x coordinate
- a_y ... rotation axis, y coordinate
- a_z ... rotation axis, z coordinate
- t_0 ... start time of movement [s]
- t_1 ... stop time of movement [s]

The rotation starts at angle 0 at time t_0 in the direction that is determined by the right hand rule and the rotation axis. Then

- if $min > max$ the particles are rotated constantly around the axis a with centre c with period per
- if $min < max$ the particles are rotated by max degrees, then back to min (in this case $min \leq 0$) degrees and then to 0 degrees within one period.

Movement continues until time t is bigger than t_1 or the end of the simulation

3.2.4 Position File

The last option is movement from a position file. The file has to be in the **execs** folder and should be named **IMV_B.i** where i is any integer. It is a simple ascii file with 4 columns t, x, y, z where t is strictly monotonically increasing. The coordinates x, y, z specify the offset of a moving boundary particle from the initial position at time 0. The positions in-between are interpolated linearly.

4 Conclusion and Outlook

With the current version 2.2 of SPHysics the code has reached a stable status, allowing for various possibilities and algorithms. Furthermore with the development of parallel SPHysics and GPU-SPHysics the particle count can be increased to millions due to the speedup available, allowing the simulation of real world problems which will be a future focus of the SPHysics group.
The problem up to now was that it was not possible to implement geometries of great complexity in SPHysicsGen. After some consideration it showed that another program, dedicated for 3-D design, had to be included to get as much freedom as possible. Blender was the optimal choice for that since it not only is open source, but its powerful Python-API allows us to create an export script that can be executed by a simple mouse-click. Due to import scripts it is also possible to include a wide array of different file formats, including some CAD files.

This guide provides a tutorial on how to create a first simple object in Blender and then export it to SPHysics. Furthermore various options in the new SPHysicsGen have been showed and the algorithms behind have been explained briefly. In Appendix A a more advanced problem has been shown. Starting with points describing a real world bathymetry, a way to import them as a mesh to Blender was described. This last example shows the power of this new approach.
The biggest restriction now is that only Dalrymple boundary conditions can be used since it is very difficult to describe corners in an arbitrary setup. In a future version these boundary conditions should be available. In case there are any significant changes to SPHysics itself the add-on should extend its capabilities accordingly. Furthermore it will be necessary to adapt the Export Script to the up-to-date version of Blender.
With the combined capabilities of SPHysics and this advanced preprocessing tool the program now tries to reach new goals. Hopefully this new addition makes SPHysics attractive to engineers to simulate various real environments and provides them with new insights on their problem.

Appendix

A Importing a Real Bathymetry

To simulate more realistic scenarios it is important to use geographical data for real bathymetries for example. In fact every type of point set can be used and transformed for use in Blender. This section gives a brief overview how this is achieved.
Assumed for this matter is a file of type *.csv* (comma separated values). In fact any string separating the values would be possible but for the sake of simplicity the following structure shall be assumed.
File structure of **.csv*:

```
x_1, y_1, z_1
x_2, y_2, z_2
...
x_n, y_n, z_n
```

The first line can be an optional header naming the three different columns.

A.1 Reading a CSV File in ParaView and Applying Delaunay Triangulation

To use the point data in Blender it is first necessary to create a mesh. This problem is commonly known as surface reconstruction and is by far no trivial task. For this matter we will use the VTK Toolkit in form of ParaView, which is also used for SPHysics post-processing.
After starting up ParaView go to **File→Open** and open the **.csv* file. In the Object Inspector choose the appropriate string delimiter and if headers are not used untick the **Have Headers** box, then press **Apply**. As result the main window should now display a table with the read in values and a fourth column called *Row ID*. To create points from this table go to **Filters→Alphabetical→Table To Points**. Once again adjust the options in the Object Inspector, i.e. choose the appropriate columns to be used for x,y and z coordinates (in case of no headers this would be: x = Field 0, y = Field 1 and z = Field 2), press **Apply**. Now the window in the middle will be blank and can be closed by clicking on the black **X** in the upper right area (don't close the program) in the new window then click **3D View** and in the Pipeline Browser click on the grey eye which should now become black and the pointcloud should be displayed in the main window.
To create the mesh itself go to **Filters→Alphabetical→Delauney 2D**. In case the surface is closed (i.e. no boundaries) choose **Delauney 3D**. It should not be necessary to change any options so the filter can be applied by clicking **Apply**. After computation is completed the main window should show a grey structure displaying the mesh.
Optionally apply the transformation filter to scale, shift or rotate the object.
To read the mesh in Blender save it as **.ply* file by **File→Save Data** and choosing **Files of type:** as **PLY Polygonal File Format(*.ply)**. This concludes the ParaView part so the program can be closed and Blender started.

A.2 Loading a PLY File in Blender and Preparing for SPHysics Export

After starting Blender and deleting all objects go to **File→Import→Stanford PLY (*.ply)**. Choose the appropriate file and import it. The mesh should now be displayed in Blender and is ready for editing. Before exporting to SPHysics the mesh has to be coloured and a fluid surface has to be specified. Additionally more faces can be added for example to bound the region or to create new structures.
It is also advisable to have a look at the face count in the upper middle area. If this number is high, it might be advisable to run a poly reducer script (in Edit Mode choose **Mesh→Scripts→Poly Reducer**) and/or convert triangles to quadrangles by pressing **Alt+J**. Note that the poly reducer could smooth

some faces which might not be desirable.
Using the method described in this section the scenery displayed in Fig. 12 was created. A Blender render is shown as well as the object after the SPHysicsGen step.

B Checking IPART_B

To check how the geometry looks after exporting follow the steps below. (Read Appendix A first for using ParaView to import **.csv* data)

1. Copy the file **source/blender/convert2csv.py** to you working directory containing the **IPART_B** file.
2. In this directory run the command **python convert2csv.py** resulting in a file **IPART_B.csv**.
3. Open this new file in ParaView (cf. Appendix A) and apply the **Table To Points** filter.
4. In ParaView, close 'Spreadsheet' and select '3-D View'.

To check the different types of particles use **Field 3** as colour index.

C Useful Blender commands & tricks

In this section we will present a random compilation of different tricks we discovered during the creation of the models shown here and in [2].
This list is in no sense exhaustive and the authors welcome any recommendations for additions, please send them to `arnom@amconception.de`.

Switching Layers

If you press any number on your keyboard Blender will switch to a different layer (Observe the header in the second window when in Edit Mode). This will result in your current object being hidden. To get back to it press number **1**, since by default your objects are created in the first layer.

Advanced Selection

When wanting to select several faces at once that share some common feature you can try the **Select→Similar** command in Edit Mode. It will give you several possibilities from which you can choose from. Also if you have a connected sub-mesh you can use **Select→Linked Vertices** to select all connected components.

Creating Pipes

`http://wiki.blender.org/index.php/Doc:Tutorials/Modeling/Curves/Making_A_Tube`

Cutting holes in your mesh

`http://blendernewbies.blogspot.com/2006/09/video-cutting-hole-in-your-mesh.html`

Checking for non-manifold vertices

Go to edit mode and press **A** once or twice such that no vertex is selected. Then press **Ctrl+Shift+Alt+M** to select all non-manifold vertices. This is especially useful when designing meshes of floating objects since they need to be a manifold.

Working only on selected parts of a mesh
Assume you have selected a few vertices / edges / faces that you want to work on without being disturbed by the rest of the mesh. Press **Shift+H** to hide all deselected vertices. As soon as your editing is done press **Alt+H** to show everything. This trick is especially useful in combination with the previous one.
to zoom in on the selected vertices press **num del**.

C.1 Links to Tutorials

Blender is a very popular 3D suite and thus there is a large community surrounding it. There are many pages describing different aspects of Blender and there is a huge variety of different tutorials available. Most of the time a simple search with your preferred search engine will provide you with in-depth descriptions on how to achieve your task. In the following a short list of the most popular Blender websites will be given. Again any recommendations for additions are welcome.

- `http://www.blender.org/education-help/tutorials/`
- `http://www.blendernation.com/category/education/tutorials/`
- `http://www.blendernation.com/links/browse/tutorials-type/`
- `http://en.wikibooks.org/wiki/Blender_3D:_Noob_to_Pro` (A complete book)
- `http://www.cgtutorials.com/c3/Blender`
- `http://www.youtube.com/results?search_query=blender+tutorial`
- `http://vimeo.com/videos/search:blender\%20tutorial/`
- and many more...

There are also a few support forums available, however you should find most of your questions already answered by searching the internet.

D Citing the software

If you are using the software please do not cite this paper but rather [2]. The appropriate BibTeX code is

```
@INPROCEEDINGS{APS,
  author = {Mayrhofer, A. and G\'omez-Gesteira, M. and Crespo, A. J. C. and
            Rogers, B. D.},
  title = {Advanced Pre-Processing for SPHysics},
  booktitle = {Proceedings of the 5th International SPHERIC Workshop},
  year = {2010},
}
```

References

[1] M. Gómez-Gesteira, B. Rogers, R. Dalrymple, A. Crespo, and M. Narayanaswamy, *User Guide for the SPHysics Code v2.0*, 2010. [Online]. Available: http://wiki.manchester.ac.uk/sphysics

[2] A. Mayrhofer, M. Gómez-Gesteira, A. J. C. Crespo, and B. D. Rogers, "Advanced pre-processing for sphysics," in *Proceedings of the 5th International SPHERIC Workshop*, 2010.