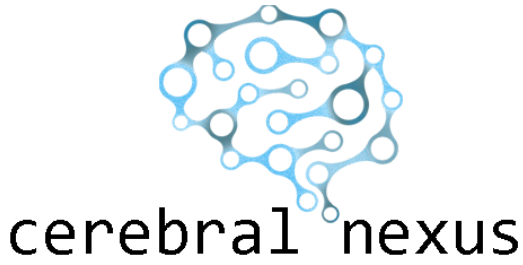


Closed Flow System 1.0

Table of Contents

Closed Flow System Overview

The Closed Flow System provides physics based 2D fluid flow simulation through a static set of enclosed pipes. For more information, visit www.cerebralnexus.com, or contact support@cerebralnexus.com with any questions or comments.

What DOES it do?

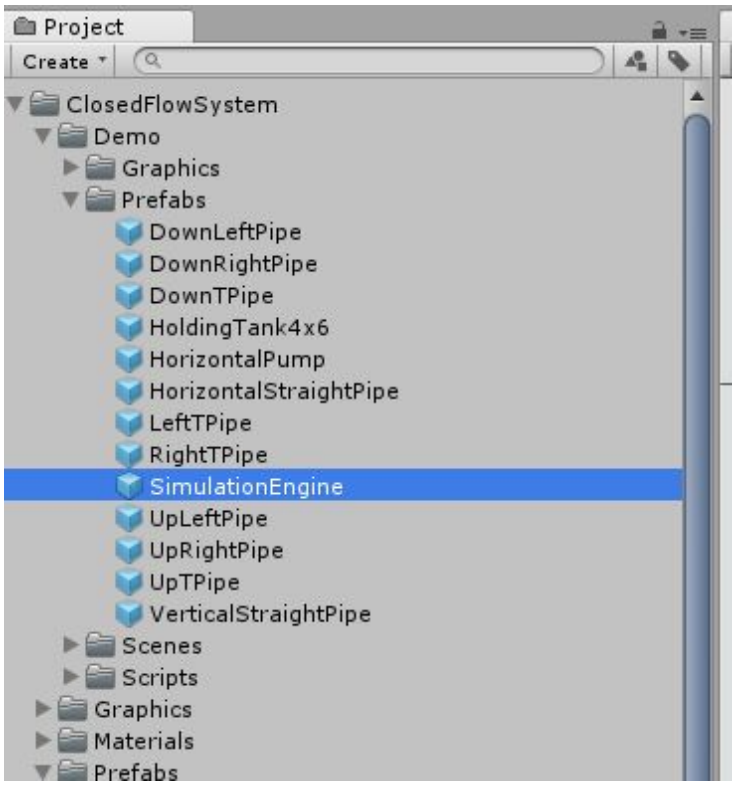
- Simulates how water would flow through a given pipe system.Supports pumps that can add acceleration to the system.
- Has a full set of straight, right angled, and T pipes, along with larger holding tanks, collectively referred to as flow nodes.
- Flow nodes must be aligned on a grid
- Has a custom editor script that will automatically snap all flow nodes to the grid after they are approximately hand placed.
- Runs fluidly on iPhone 6s.

What DOESN'T it do?

- Detailed realistic water surface simulation.Buoyancy calculations.
- Any flow outside of the static, predefined pipe system.
- Work with WebGL

Basic Scene Setup

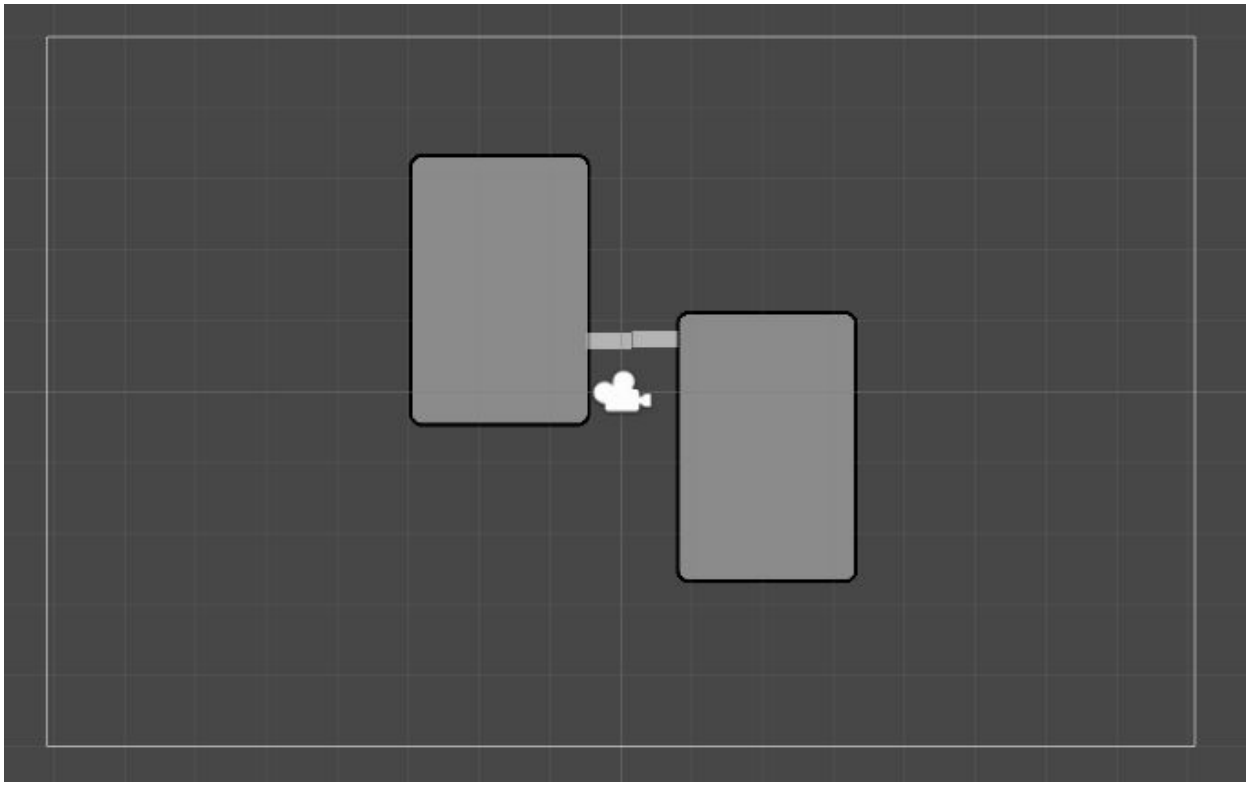
This section will go through setting up a flow system from a new, blank scene. The first step will be to add the demo **SimulationEngine** prefab to the scene. This is located under **ClosedFlowSystem\Demo\Prefabs\SimulationEngine**.



SimulationEngine prefab

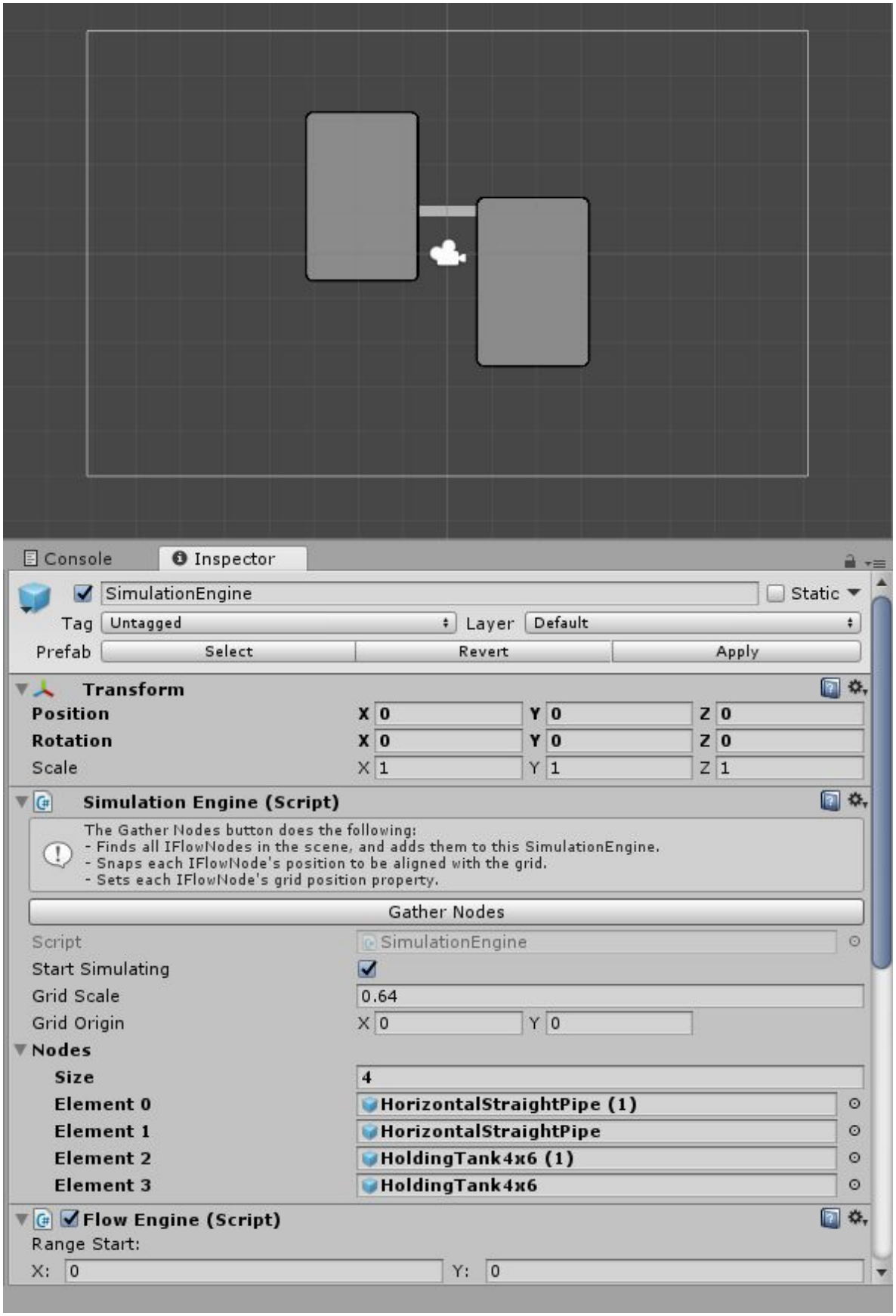
The **SimulationEngine** prefab has the **FlowEngine** and **SimulationEngine** scripts. The **FlowEngine** script is what does the flow simulation. The **SimulationEngine** script starts and stops the flow simulation, and keeps track of all of the flow nodes.

Next we'll add the flow nodes to the scene. In this example we'll just have one holding tank above another, with two pipes connecting them. Start by adding two instances of the **HoldingTank4x6** prefab and two instances of the **HorizontalStraightPipe** prefab to the scene. Then arrange them to look roughly like the below image. Don't worry about getting very exact with your placement.



Approximately placed nodes

Once you have placed the flow nodes, go back and select the **SimulationEngine** prefab that was added earlier. In the inspector, press the **Gather Nodes** button at the top of the **SimulationEngine** script area. This should snap the roughly added nodes to the grid, and add them to the **Nodes** list on the **SimulationEngine**. Your scene/inspector should look something like below.



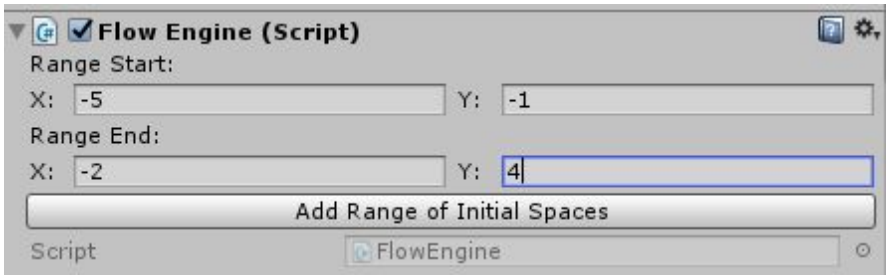
After nodes have been gathered

Now you've got the nodes all set up, but no water. In this example, we're going to fill the upper tank with water. To do this, we first need to know where in the grid the upper tank is. To find

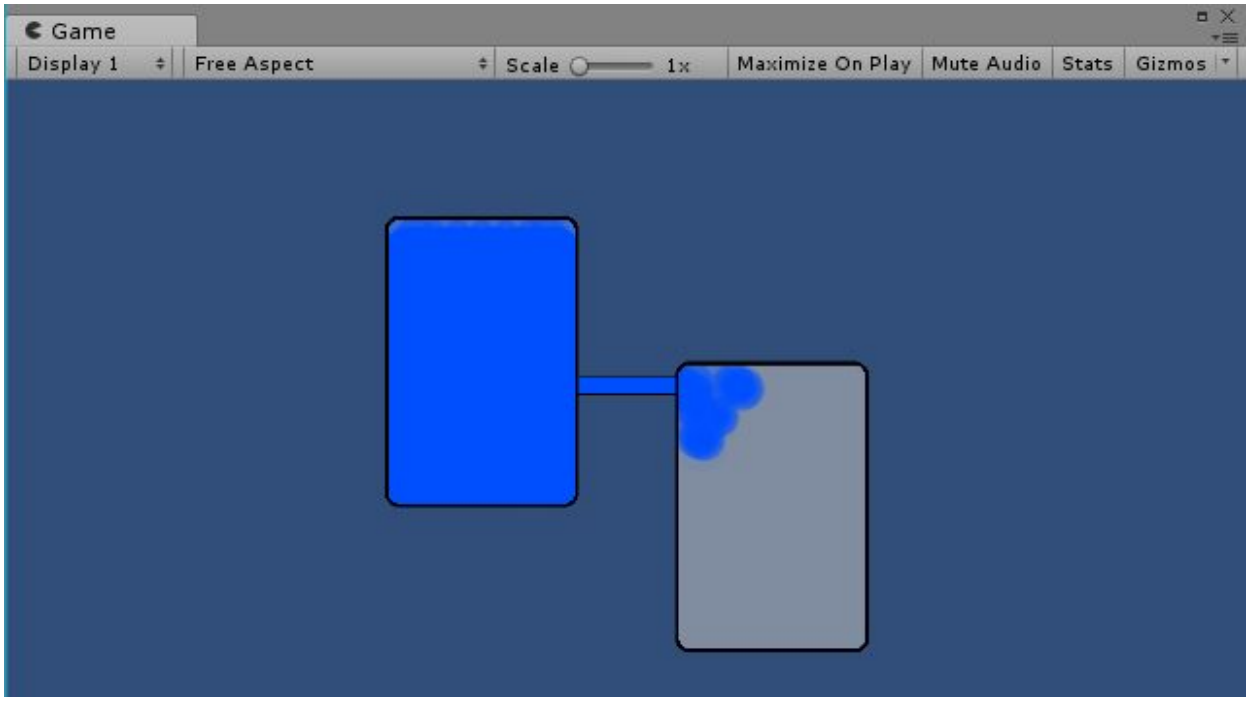
out, select the upper holding tank. As you can see, the holding tank has a **Lower Left**, and a **Width** and **Height**. In the example to the left, the lower left is **(-5,-1)**, and it is **4** spaces wide and **6** spaces tall (hence the name **HoldingTank4x6** ;)).

Holding Tank data

We're now going to use that holding tank position data as input to the **FlowEngine**. So first select the **SimulationEngine** GameObject. In the **FlowEngine** section of the inspector, put in the holding tank's **Lower Left** corner as the **Range Start**, and the holding tank's upper right corner as **Range End**, then click **Add Initial Spaces**. Note that the upper right coordinate is **(left + width - 1, bottom + height - 1).** Before you click, it should look something like below, given the holding tank example above.



After clicking, **Initial Water Spaces** should now have a size of 24. Now just click play, and watch water flow from one tank to the other!



Water flowing!

Congratulations, you just finished setting up your first flow simulation!

Script Options

FlowEngine

The **FlowEngine** is the main workhorse that runs the flow simulation. It is built to plug into a potentially more comprehensive simulation engine implementing the **ISimulationEngine** interface. The **SimulationEngine** class included in the demo provides an example implementation of **ISimulationEngine**. Other than **Initial Water Spaces**, the default values for the FlowEngine properties should be sufficient. They are described below for completeness.

Editor Properties

Add Range of Initial Spaces (Button)

Used for bulk adding a range of spaces to initially contain water. Adds each grid space in the rectangle whose opposing corners are defined by **Range Start** and **Range End**. Any spaces already in the list are ignored and not re-added.

Initial Water Spaces (List<GridSpace>)

A list of grid spaces that will have water in them when the simulation starts. Values can be added manually, or in ranges via the **Add Range of Initial Spaces** button.

Marker Sprite (Sprite Renderer)

The SpriteRenderer prefab that will be instantiated for each water particle that is flowing in the system.

Filled Sprite (Sprite Renderer)

The SpriteRenderer prefab that will be instantiated to fill in the gaps between water particles in areas that should be a continuous body of water.

Solver Max Iterations (int)

The fail safe maximum to ensure the iterative part of the algorithm will never loop infinitely.

Solver Tolerance (float)

The value deemed close enough to zero to be used as a finishing point for the iterative part of the algorithm.

Fill Increment (float[0-30])

The rate at which the fill sprites fade in to fill in gaps. If you are finding a lot of awkward looking gaps between particles, increasing the **Fill Increment** can help ensure they are filled in quicker. In general it should be greater than the **Drain Increment**.

Drain Increment (float[0-30])

The rate at which the fill sprites fade out when a space is no longer considered entirely filled. If too small, this can lead to water seeming to linger long after it actually left. If too large, or greater than the **Fill Increment**, it can lead to the fill sprites strobing in and out if the fill state is oscillating quickly.

Fill Divisions (int[1-4])

The amount each grid space is divided in each direction to place the fill sprites. There will be Fill Divisions squared fill sprites per grid space. Lower values can result in low resolution edges, while higher values can potentially cause awkward gaps and performance issues.

Division Multiplier (int[1-8])

The fill sprites are shown if they are surrounded by water. The **Division Multiplier** is the number of grid spaces it checks in each direction to determine whether there is water around the space. Increasing this tends to make the fill sprites more stable, but also adds significant processing time that can slow down performance.

Starting Particles Per Space (int[4-16])

The number of particles instantiated in each **Initial Water Space**. Their position in the grid space is randomly generated in alternating quadrants of the space. More particles enables more accurate tracking of where exactly the water is, at the cost of more work per frame.

Fill Update Frequency (float[0-1])

Allows for rate limiting how often the fill sprites are updated. The value is specified in seconds. The fill sprites will be updated after the longer of the **Fill Update Frequency** and the standard **Update** call.

Auto Fill Particle Count (int[1-20])

Short circuit mechanism for showing the fill sprites. If there are at least this number of particles in a fill space, it will automatically start showing the fill sprite, and entirely skip checking the surrounding spaces for water.

Maximum Water Opacity (float[0-1])

The opacity value applied to marker particle sprites, and the maximum opacity the fill sprites will fade into. Note that it only gets set on newly instantiated marker particle sprites. As such, due to the use of a sprite pool, the game must be restarted for changes to this value to be reflected.

SimulationEngine

The **SimulationEngine** is the demo implementation of **ISimulationEngine**.

ISimulationEngine

ISimulationEngine is an interface that defines how the **FlowEngine** will talk to a more comprehensive simulation engine. Implementations are expected to all **FlowEngine.Initialize** as part of **Awake().**

StartedSimulation (event Action)

The **FlowEngine** keys off of this event to start the flow simulation. Should not be called if the simulation is already running.

StoppedSimulation (event Action)

The **FlowEngine** keys off this event to stop the flow simulation.

AllNodes (IEnumerable<IFlowNode>)

An enumeration of all the **IFlowNodes** from the scene that should be part of the flow simulation. The demo **SimulationEngine** has an editor script that automatically populates a list from what is in the scene. In a full fledged game, you may have other mechanisms for this list to change at runtime while the simulation is stopped. The **FlowEngine** will re-initialize itself from this enumeration each time the simulation is started.

TransformGridToWorld(Vector2 -> Vector2)

The **FlowEngine** works on a grid system that might not directly correspond to world coordinates. This function does whatever transformations necessary to take a **Vector2** position in the grid coordinate system, and returns where that position is in world coordinates.

Editor Properties

Gather Nodes (Button)

The Gather Nodes button does the following:

- Finds all IFlowNodes in the scene, and adds them to this SimulationEngine.
- Snaps each IFlowNode's position to be aligned with the grid.
- Sets each IFlowNode's grid position property.

Start Simulating (bool)

A bool value specifying whether the simulation should be started automatically when the scene starts.

Grid Scale (float)

The size in world units of a single grid space.

Grid Origin (Vector2)

The origin of the grid coordinate system in world space.

Nodes (List<GameObject>)

This is a list of GameObjects because the editor doesn't support interfaces by default. However it is expected that all of the GameObjects in this list will have a behaviour that implements **IFlowNode**. For this demo implementation, it is generally expected that this will only be modified by clicking the **Gather Nodes** button.

Pipe

Most of the **Pipe** properties are what define where water can enter/leave the pipe, and should not be modified.

Editor Properties

Pipe Space (GridSpace)

The location in the grid of the pipe. In the demo this is automatically set by the **Gather Nodes** button on the **SimulationEngine**. If you are placing pipes at runtime, this value must be updated before restarting the simulation.

Default Sprite (Sprite)

The sprite for the pipe.

HoldingTank

The **HoldingTank** represents a walled rectangular container for water. By default, all the edges are solid. Openings are automatically created on an edge if there is an adjacent pipe that has an opening on the same edge.

Editor Properties

Lower Left (GridSpace)

The location in the grid of the lower left corner of the holding tank. In the demo this is automatically set by the **Gather Nodes** button on the **SimulationEngine**. If you are placing holding tanks at runtime, this value must be updated before restarting the simulation.

Width (int)

The number of grid spaces wide the tank is.

Height (int)

The number of grid spaces tall the tank is.

HorizontalPump

The **HorizontalPump** is an extra script placed on a horizontal straight pipe. It imparts the ability to add extra acceleration to the system. See http://www.cerebralnexus.com/controlling-the-simulation-tutorial.html for a more detailed tutorial on controlling a pump.

Editor Properties

Maximum Acceleration (float)

This is the maximum acceleration value that can be added in either direction.

Functions

SetActuation (float)

SetActuation expects a float between -1 and 1. It will update the added acceleration to be the given float times that pump's **Maximum Acceleration** property. This could be called from code, or it can be hooked up to the On Value Changed event of a slider.