Fluid Flow

You can always find the most up to date documentation here:
https://drive.google.com/drive/folders/1YzdllLIZMiyoiQXxa5WZUSETRrSyIQAs

Setup (standard): https://www.youtube.com/watch?v=c8V8i6_dU00

Attach a 'FluidSimulator' component to your object. If there is a 'Renderer' component on the same game object it will get assigned automatically.

Open the 'Fluid Object Creator' (Window/Fluid Object Creator) and set your desired fluid texture resolution.

Then assign the mesh you want the fluid to be simulated on. This mesh must have a not overlapping set of UVs and UV islands should be separated by a few pixels. Choose this UV set under 'UV Chanel'.

If your mesh has multiple submeshes you can also decide which will be included in the simulation.

After pressing 'Generate FluidObject' your 'FluidObject' will be generated within a few seconds (depending on the texture size and triangle count).

When finished there should be a 'MeshName_FluidData' object in your assets folder.

Assign this to the 'Fluid Object' field of your 'Fluid Simulator' component.

Now change the shader of your material to 'Fluid Simulation/Fluid'. (or HDRP/ Universal depending on your rendering pipeline)

Create a new empty game object in your scene and add a 'Fluid Simulation Manager' component.

You always need one(!) 'Fluid Simulation Manager' component in your scene, otherwise one will be created at the start of the game.

The last step is to define how the fluid will be added to your simulation.

You can attach the included ‘Easy Fluid Interaction’ script to your object and assign your main camera and ‘FluidSimulator’. This will shoot a ray from the camera to where you clicked on the screen and draw a sphere of fluid where it hits. So, make sure your object has a collider attached.

Alternatively, you can write your own script, which calls the DrawTexture/Sphere/Disc methods on your simulator.

Setup (skinned):

Setting up the fluid simulation for an animated object is very similar to a normal mesh.

The first difference is that you have to check ‘fromPose’ in the ‘Fluid Object Creator’ when generating the ‘FluidObject’. Then you will have to assign a ‘SkinnedMeshRenderer’ from your scene instead of a mesh. The current pose of the ‘SkinnedMeshRenderer’ will be baked and the baked mesh will be used for fluid simulation. For example, you can make the arms of your character point downwards, so the fluid will flow down the arms at runtime.

The second difference is that when you want to paint fluid on the simulator you will have to use the ‘SkinnedDrawTexture/Sphere/Disc’ version of the methods. Because the mesh was baked at the ‘FluidObject’ generation, the bone positions of your character might not match the baked positions, so the fluid would be created at the wrong positions.

The skinned draw methods require a reference to the closest bone to the brush position. Using this bone, it will calculate the correct position on the baked mesh for you.

You can use the ‚TransformPointToDefaultPose()‘ method of the ‘FluidObject’ in order to do this calculation manually.

How it works:

The ‘Fluid Object Creator’ generates a texture for the mesh, where each pixel holds information to which triangle it belongs to and where on the object it is located in local 3d space. If the pixel is outside the uv islands, it searches the closest one. It also knows where on the uv map the connected uv island is located. This information is stored in 4 uints per pixel like this:

([X, Y], [Z, normal], [UV], [ID, inside)

- Position XYZ are values from 0x0-0xFFFF representing the position of the texel inside the bounding box of the mesh
- Normal of the triangle is stored in polar coordinates compressed into two bytes
- UV points to the neighboring UV island if the pixel is outside, or to the position of the pixel itself if it is inside a UV island. X and Y are compressed in one 4-byte uint
- ID is the ID of the triangle the pixel belongs to. Needed to calculate the gravity pull on this pixel
- inside is one bit signaling if the pixel is inside a UV island or outside.

Additionally, a 3d to UV conversion matrix is computed for each triangle, so that the gravity pull in UV space can be calculated in real-time.

The update cycle is made up of three steps:

- each pixel checks how much fluid is pulled away by gravity and saves this value to a ‘flow’ texture
- each pixel samples from this ‘flow’ texture where it expects new in-flowing fluid from
- the in-flowing amount gets added and the out-flowing amount subtracted from the total fluid amount of the pixel
- finally the fluid amount, the ‚age‘ of the fluid and the normal is written to the final fluid texture

Repeating this cycle multiple times per second creates the flowing effect.

The resulting fluid texture is used as a mask to tell the material where to draw fluid.

Custom Fluid Shaders

When your ‘FluidSimulator’ awakes, it will call ‘SetMaterialData()’ on your renderer and try to set the ‘_FluidTex’ property of the materials. And enable/ disable the ‘_Fluid_UV1’ or ‘_Fluid_UV2’ keywords, depending on the UV set used to generate the ‘FluidObject’.

You can also access the fluid texture directly by calling ‘GetFluidTexture()’ on your simulator.

‚GetUVChanel()‘ gives you the UV channel the ‘FluidObject’ is using.

The color channels of the fluid texture are mapped like this:

R: amount of fluid stored in this texel (0-1)

G: dryness of the fluid (0 = fresh; 1 = dry)

B/A: X/Y of the normal direction (0-1; needs to be unpacked)

Look at the included example shaders (FluidFlow/Shaders/Fluid)

Fluid Object Creator (Window/Fluid Object Creator)

The Fluid Object Creator converts mesh data to a ‘FluidObject’ used for fluid simulation.

The mesh must have a set of not overlapping UVs and there should be a padding of a few pixels between UV islands.

- **Texture Size**: resolution of the fluid texture
- **From Pose**: if checked you can select a ‘SkinnedMeshRenderer’. When generating the ‘FluidObject’ the current pose will be baked. Otherwise you can select a mesh
- **Mesh**: geometry the ‘FluidObject’ will be generated for
- **Skinned Mesh**: deformed geometry the ‘FluidObject’ will be generated for. the current pose will be baked and used for the gravity maps
- **Submesh Mask**: if the mesh has multiple submeshes, you can select which will be used in the ‘FluidObject’ (i.e. if they have overlapping uv maps)
- **UV Channel**: UV channel used for the simulation. UVs must not overlap and there should be a gap of at least two pixels between UV islands
- **Generate FluidObject**: Start generation. Depending on vertex count, resolution and your system this might take a few seconds
- **Auto Compress**: automatically compress the ‘FluidObject’ after creation. You can also compress ‘FluidObject’s manually in the FluidObject-Inspector

Fluid Simulator

You need one ‘FluidSimulator’ component for each renderer you want to simulate fluid on

- **Fluid Object**: geometry data used for the simulation. Generated by the ‘Fluid Object Creator’ (Window/Fluid Object Creator)
- **Renderer**: renderer used to draw the model/fluid
- **Timeout**: after adding fluid X update cycles will be simulated
- **Flowspeed**: maximum distance (pixels) the fluid can move each cycle (high values may cause fluid ‘teleporting’)
- **Roughness**: amount of fluid each texel can hold
- **Normalmap**: depending on the surface structure and gravity more/less fluid can be stored in each texel (optional)
- **Normalmap Amount**: influence of the normal map
- **Slope Amount**: reduce the amount of fluid in areas with a high slope
- **Dry Time**: how fast does the fluid dry (seconds)
- **Fluid Evaporation**: enable fluid evaporation
- **Evaporation Amount**: the amount of fluid removed from each texel per second
- **Check for Drops**: this only works if your hardware supports AsyncGPUReadback. Otherwise CPU and GPU would have to sync which would cause very bad performance. Disable this option for better performance
- **Min Fluid Amount**: how much fluid is needed in a texel to form a drop
- **Updates Skipped**: increase update cycles between checking for drops, to maximize performance
- **OnDrop**: called when a drop is formed. First parameter is the world position and second the fluid amount

Call the ‘DrawTexture/Sphere/Disc’ methods to add fluid to the surface of your object. If your object is animated use the ‘Skinned’ version of the methodes. These also expect the transform of the closest bone, in order to transform the position from world space to the baked position on the ‘FluidObject’.

Fluid Simulation Manager

One(!) instance is necessary in each scene using ‘FluidSimulators’.

Created automatically if there is a ‘FluidSimulator’ and no ‘FluidSimulationManager’ present in the current scene.

It manages the update queue and shared simulation data (when there are multiple simulators using the same FluidObject some data can be shared between them)

- **Update Mode**:
  - **Time:** update X times per second
    - **Cycles Per Second**: amount of update cycles per second
  - **Frame:** update every X frame
    - **Skipped Frames**: frames skipped between update cycles
- **Maximum Updates**: each cycle a maximum of X simulators will be updated. The rest will be queued to be updated later
- **Visible**: only simulators visible to a camera will be updated
- **Minimum Evaporation**: after a simulator has timed out, fluid will keep on evaporating. In order to save draw calls, the texture will only update when the fluid amount has changed by this amount (higher value → less draw calls)
- **Use Interger RenderTexture**: switch between integer RenderTextures and integer ComputeBuffers as data storage for fluid simulation. Disable integer RenderTextures for mobile platforms

Gravity Updater

Attach this component to an object with a ‘FluidSimulator’ component in order to automatically update the gravity maps when the object is rotated

- **Deadzone Angle**: update when the gravity direction changes by this angle in degrees

Drop Handler

If you attach this component to an object with a ‘FluidSimulator’ component the ‘**Drop Prefab**’ will be instantiated each time the ‘OnDrop’ event is called on the ‘FluidSimulator’

If you have any further questions, feel free to contact me: mr3d.cs@gmail.com