Fazio, F – School of Integrative Science and Technology (IST)
Fall 2025

Water Simulation Tutorial

I. Introduction

This project begins with a physically-based fluid simulation developed in **Blender**, leveraging its Mantaflow engine for realistic water behavior and graphics rendering. Blender's integrated solver provides accurate particle-based liquid dynamics, ideal for visualizing fluid-object interactions in a confined domain. The primary geometry consists of a **cubic container** representing a fluid domain, within which are several vertical structures resembling buildings, serving as **static collision bodies**. Water is introduced through an inflow mechanism, and as the simulation progresses, it moves toward one side of the cube, **impacting the obstacles and generating a wave-like splash**. The end goal is to **transfer this simulation to Unity**, allowing real-time interaction and visualization in a game and interactive application environment.

For this project, we followed the YouTube tutorial **"Beginner's Guide to Liquid Simulation in Blender" by BlenderMadeEasy** step by step to create the water scene. The video provided a clear walkthrough of setting up a fluid domain, adding a flow object, placing collision obstacles, and configuring the liquid simulation using Blender's Mantaflow system. By closely following the tutorial, we were able to simulate realistic water movement and splashing effects within a container, forming a wave as the fluid interacts with building-like pillars.

II. Blender: Scene Setup

1. Create the Fluid Domain

- Use the default cube as the **domain**
- Scale it:
  - X = 9m, Y = 5m, Z = 5m
  - Move it up so it sits on the ground
- This cube defines the bounds of the simulation

2. Add the Flow Object (Water Source)

- Duplicate the domain cube (Shift + D)
- Scale it down:
  - Make it roughly 2m wide
  - Move it to the **left side** of the domain
- This will be the **initial block of water**

3. Add Collision Objects

- Add **3 cylinders** (Shift + A > Mesh > Cylinder)
- Set vertices to 128 for smooth edges
- Scale them tall and thin
- Position them in front of the flow cube (like pillars)
- Make sure they **sit slightly below the domain floor**

4. Apply Scale

Why: Unapplied scales can cause fluid glitches, such as water leaking through obstacles or behaving unrealistically.

How:

1. Select all relevant objects (domain, flow, obstacles)
   → Press A to select everything
2. Press Ctrl + A → choose Scale
3. In the Item panel (N), confirm all scale values are 1.0

5. Remove the Default Lamp

Why: The default light can create unwanted shadows or interfere with your custom lighting.

How:

1. In the Outliner, find the object named Light
2. Select it → Press X or Delete

III. Blender: Fluid Simulation (Mantaflow)

6. Domain Fluid Settings

- Select domain → Physics tab → Enable **Fluid**
  - Type: **Domain**
  - Domain Type: **Liquid**
  - Resolution: **196** (or lower if needed)
  - Time Scale: **0.85**
- **Mesh Settings**:
  - Enable Mesh

    - Upres Factor: 2
    - Particle Radius: **1.0**
- **Cache Settings**:
    - Type: **Modular**
    - Enable **is Resumable**
    - End Frame: **180**
    - Bake folder: change from "temp" to a custom directory if needed

7. Flow Object Settings

- Select the small cube → Fluid → Type: **Flow**
    - Flow Type: **Liquid**
    - Flow Behavior: **Geometry**

8. Collision Objects Settings

- Select each cylinder → Fluid → Type: **Effector**

9. Bake Fluid & Mesh

- Select domain → Bake Data
- Then → Bake Mesh

10. Hide Flow Cube

- Hide from viewport and render (eye & camera icons)

11. Add a Ground Plane

- Add a plane → Scale it
- Extrude the back edge upward
- Bevel the corner for smoother backdrop

12. Setup the Camera

- Align view → Ctrl + Alt + NumPad 0
- Adjust camera position to focus on the simulation
- Set resolution to **1000 × 1000 px**

13. Fake a Glass Box

- Duplicate the domain cube → Remove fluid physics

- Slightly scale it to wrap around the fluid mesh
- Add a **Wireframe Modifier**
- Create a **black glossy material**

14. Apply Materials

- **Water (domain):**
  - Use a Glass BSDF
  - Roughness: 0, IOR: 1.333
  - Color: Light blue
- **Wireframe container:** dark glossy material
- **Cylinders:** link them to same material as container
- **Ground:** Glossy material (Roughness = 0.1)

15. Lighting & EVEE Settings

- Add a **Sun Lamp**, rotate for shadow
- Enable in Render tab:
  - **Ambient Occlusion**
  - **Screen Space Reflections**
  - **Refraction**
- In Material → Settings:
  - Enable Screen Space Refraction
- In Color Management:
  - Set Look: Very High Contrast

16. Render Output

- Set output folder and format: MPEG / MP4
- Set end frame to **180**
- Render > Render Animation

IV. Exporting to Unity

A. Exporting the Fluid as Alembic (.abc)

Why Use Alembic?: Alembic is ideal for **fluid simulations** because it preserves:

V. Mesh animation frame by frame

VI. Vertex deformations (perfect for Mantaflow fluids)
VII. Realistic motion with high accuracy
VIII. Works well with **Unity's Alembic Stream Player**

How to Export the Fluid (Step-by-Step):

1. **Select the Fluid Domain Object**
   - This is the cube that holds the baked simulation mesh.
2. **Make Sure It's Baked**
   - Under Physics > Fluid > Cache, verify that both Data and Mesh are baked.
3. **Go to File > Export > Alembic (.abc)**
   - From the top menu:
     File > Export > Alembic (.abc)

Table 1: Configure the Export Settings (bottom-left of the window)

Setting	Value	Notes
✅ Apply Transform	Checked	Ensures correct world-space position in Unity
✅ Flatten Hierarchy	Checked	Combines into a single mesh hierarchy (Unity-friendly)
✅ Export as Mesh	Checked	Ensures you're exporting visible mesh, not particles
✅ Include Animation	Checked	Includes time-based mesh deformation
Sampling Rate	1.0	Exports 1 frame per Blender frame (real-time fluid motion)
Start / End Frame	Match simulation range	e.g., 1–180

4. Name the File

- Example: fluid_simulation.abc
- Save It to a Location you'll import from Unity

B. Exporting the Container and Pillars as FBX

Why Use FBX?: FBX is ideal for static mesh objects like:

- The glass container cube
- The pillar-like obstacles
  It preserves object hierarchy, transforms, and smooth shading — and Unity supports FBX natively.

How to Export the Static Geometry (Step-by-Step):

1. **Select All Static Meshes**
   - Select the outer container cube (the wireframe or glass cube)
   - Select the cylinders (pillars)
   - Hold Shift to multi-select
2. **Go to File > Export > FBX (.fbx)**
   - From the top menu:
     File > Export > FBX (.fbx)

Table 2: Configure Export Settings (bottom-left panel)

Setting	Value
Limit to:	✅ Selected Objects only
Apply Transform	✅ Checked
Apply Unit	✅ Checked *(optional)*
Mesh: Apply Modifiers	✅ Checked
Smoothing:	**Face** or **Normals Only**
Forward:	-Z Forward
Up:	Y Up

3. Name the File

- Example: fluid_environment.fbx

V. Scene Comparison: Blender vs. Unity

This section illustrates the visual transition of the fluid simulation from Blender to Unity. The first figure shows the high-quality water simulation as created and rendered in **Blender** using the **Eevee render engine**. The second figure demonstrates how the simulation appears after being **exported and imported into Unity**, where it runs in real-time using the **Universal Render Pipeline (URP)**.

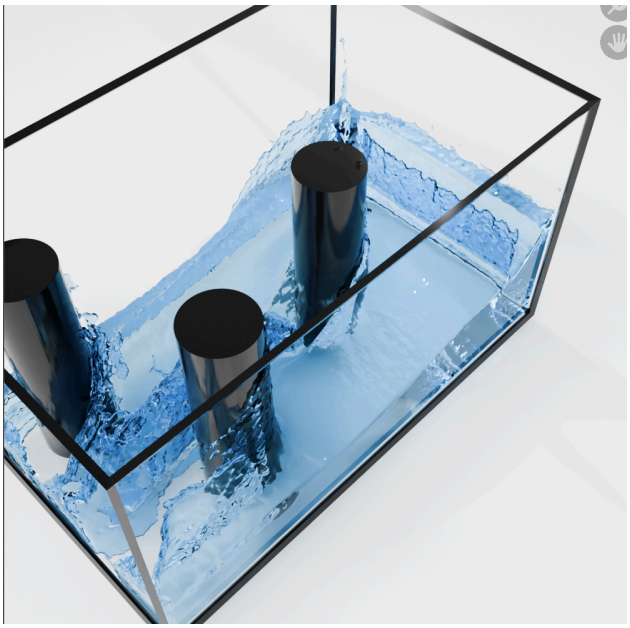

Figure 1: Fluid simulation in Blender with glass shader, reflections, and screen-space refraction enabled.

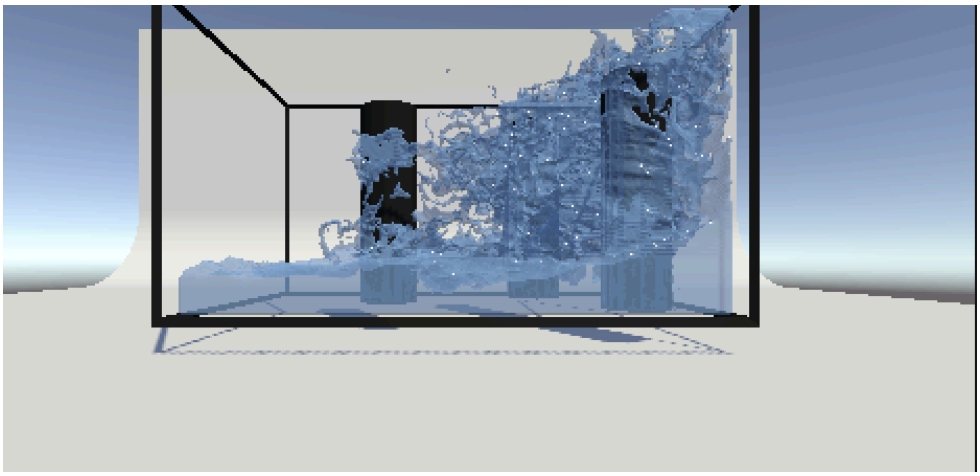

Figure 2: Fluid simulation playing in Unity using Alembic Stream Player and URP materials.