Advancing Geotechnical Visualization: Integrating Houdini and Material Point Method for Enhanced Water Flow Dynamics in Dam Structures

CIVENG 299 PROJECT

University of California, Berkeley

Geosystem Engineering

By

Yulia Nugroho

May 2024

TABLE OF CONTENT

I. INTRODUCTION 1
II. STUDY CASE AND SCOPE OF WORK 1
III. OPERATIONAL WORKFLOW I: PARAVIEW 2
IV. OPERATIONAL WORKFLOW II: HOUDINI 3
4.1 Inputting MPM Data into Houdini 5
4.2 Creating a Barrier Block Composed of Crushed Rocks 8
4.2.1 Creating a Basic Box 8
4.2.2 Randomizing Particles in the Barrier Block 9
4.2.3 Creating the Shape of Crushed Rocks 11
4.3 Creating a Water Flow Simulation 13
4.3.1 Creating Point Attributes 14
4.3.2 Water Simulation Nodes 14
4.3.3 Preliminary Result 17
4.4 Application of Materials to Add Color and Texture to the Model 18
4.5 Lighting and Camera Configuration 19
4.6 Rendering Process 20
V. OPERATIONAL WORKFLOW III: HOUDINI – ADDITIONAL 23
5.1 Creating a Backdrop 23
5.2 Creating a Transparent ‘Aquarium’ Like a Box 24
5.3 Creating a Drawer for Placing the Box 25
5.4 Rendering Result After all Materials are Put Together 26
VI. LIMITATIONS OF WORK 27
VII. CONCLUSION 28

LIST OF FIGURES

Figure 2.1 Dam design and dimension 1
Figure 2.2 Raw MPM VTP File 2
Figure 3.1 Exporting VTP file to ParaView 3
Figure 3.2 Houdini writer on ParaView 3
Figure 4.1 Geometry node in Houdini with blue and green flags on 5
Figure 4.1.1 Placement of nodes in Houdini 5
Figure 4.1.2 File node for opening MPM Data 6
Figure 4.1.3 MPM data's Geometry Spreadsheet 6
Figure 4.1.4 Nodes on water_MPM's SOP level 7
Figure 4.1.5 Raw MPM data simulation 8
Figure 4.2.1.1 Basic box with 'transform' node 9
Figure 4.2.2.1 A set of nodes to randomize particles 10
Figure 4.2.2.2 Attribute Randomize (left) and Attribute Wrangle (right) 11
Figure 4.2.2.3 Barrier block after randomized nodes applied 11
Figure 4.2.3.1 Sphere before and after fracture node is being applied 12
Figure 4.2.3.2 Blast node settings 12
Figure 4.3.3.3 Sphere after blast node is implemented 12
Figure 4.2.3.4 Whole nodes chain in 'box_crushed_rocks' geometry node 13
Figure 4.2.3.5 Barrier block with gravel of rocks 13
Figure 4.3.1 Point attributes for water simulation 14
Figure 4.3.2.1 'Particle Fluid Surface' parameter setting 15
Figure 4.3.2.2 Whole nodes chain in 'visualize_water' geometry node 17
Figure 4.3.3.1 Three basis nodes for simulation 17
Figure 4.3.3.2 Several steps of simulation before applying material for texture and color 18
Figure 4.4.1 Principal shader material for crushed rocks 19
Figure 4.4.2 Uniform volume for varying water depth color 19
Figure 4.4.3 Basic liquid material for water reflection and clarity 19
Figure 4.5.1 Lighting and Camera Nodes 20
Figure 4.6.1 Mantra rendering settings 21
Figure 4.6.2 Group of Raw Process Nodes and Render Nodes 21
Figure 4.6.3 SOP level within Render node: 'object merge' (left) and 'material' nodes (right) 22
Figure 4.6.4 Rendered preliminary results step 1 – step 209 23
Figure 5.1.1 Backdrop as background 24
Figure 5.2.1 Making of the transparent box 24
Figure 5.2.2 Principal shader settings for the transparent box 25

Figure 5.3.1 Drawer for placing the box 26
Figure 5.4.1 Final rendered results from step 1 – step 209 27
Figure 6.1 Recommended computer specs to run Houdini appropriately 28
Figure 6.2 The current laptop specs used to run Houdini 28

I. INTRODUCTION

In the field of geotechnical engineering, the visualization and modeling of engineering data play a pivotal role in effectively communicating research outcomes and facilitating informed decision-making. As the complexity of engineering challenges continues to increase, especially in areas such as dam safety and water flow management, the need for advanced simulation techniques becomes more critical. The Material Point Method (MPM) has emerged as a robust tool in this context, offering detailed insights into complex physical phenomena that are often difficult to capture with traditional methods.

The goal of this project is to demonstrate how combining MPM with advanced visualization tools like Houdini can produce accurate and visually appealing simulations that make it easier for engineers and the general public to understand critical safety issues. While MPM has traditionally been used in the simulation of deformable solids, its application in hydrodynamic studies represents a novel and significant expansion of its utility. By integrating MPM with cutting-edge computer graphics, we aim to produce animations that not only adhere to scientific rigor but also enhance the comprehension of these complex phenomena for both technical and lay audiences.

This project explores the innovative application of Houdini to translate MPM data into dynamic 3D visualizations. The subsequent sections will detail the methodologies employed, from the initial data processing in ParaView to sophisticated simulation workflows in Houdini, leading to detailed visual outputs that apply these complex methods to practical engineering problems.

II. STUDY CASE AND SCOPE OF WORK

The data used for this research comprises MPM particle points depicting water flow. On one side, a block composed of rocks impedes the water flow, while on the other side, the water flows freely. The goal of this work is to visually represent the simulation, which can be termed as an animation.

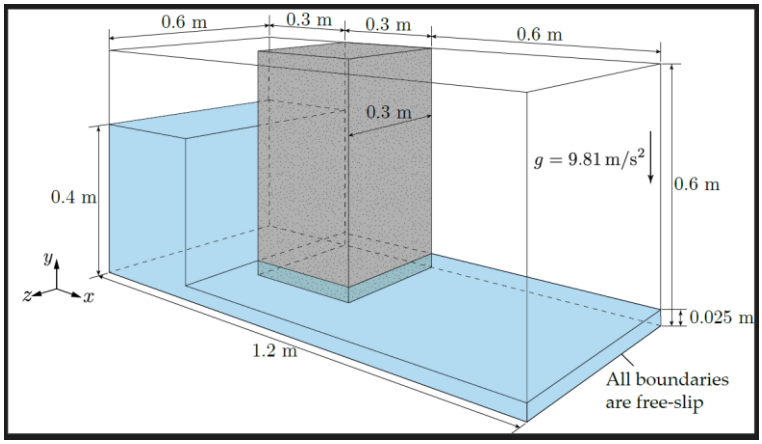


Figure 2.1 Dam design and dimension.

To prepare the MPM data for use in 3D modeling software, it must first be input into a software called ParaView. Once the MPM data is loaded into ParaView, it can then be saved in various formats compatible with 3D modeling software. For this simulation, Houdini is the preferable software. Although any 3D modeling software like Blender or Maya could be used, previous studies have shown that Houdini is particularly well-suited for creating simulations from MPM data. This study primarily focuses on creating realistic visualizations of the water movement in a dam, aiming to simulate the water's motion as accurately as possible based on available water pressure and velocity data. There are **240 steps** of simulation that will be done in this project.

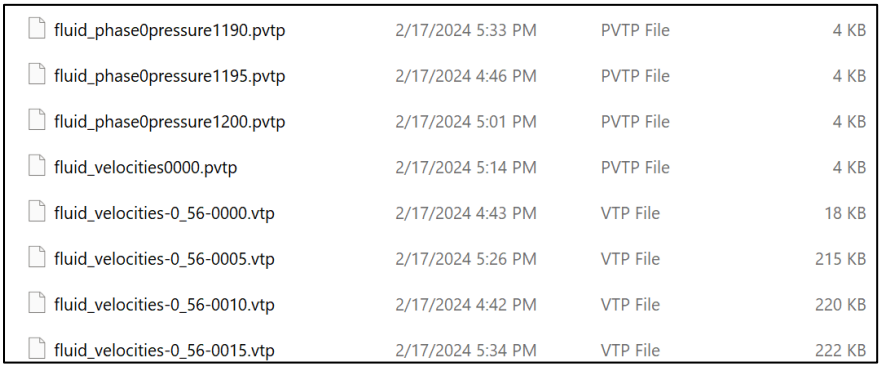


Figure 2.2 Raw MPM VTP file.

In the next section, operational work will be divided into two parts: importing MPM data into ParaView, and exporting it to Houdini. The discussion about ParaView will be very brief since it is only used to convert MPM data into a file format that can be read by Houdini. Subsequently, the majority of the discussion will focus on the methods of simulation and animation to be carried out in Houdini.

III. OPERATIONAL WORKFLOW I: PARAVIEW

ParaView is software capable of opening MPM data. Therefore, the first step is to open the MPM file from ParaView. The initial step is straightforward, similar to opening a regular file: start with 'file,' then 'open file,' and then select all the MPM files that will later be exported to Houdini. Once all the files appear in the pipeline browser, make sure to 'save as' all these files in the Houdini file format (*.geo).

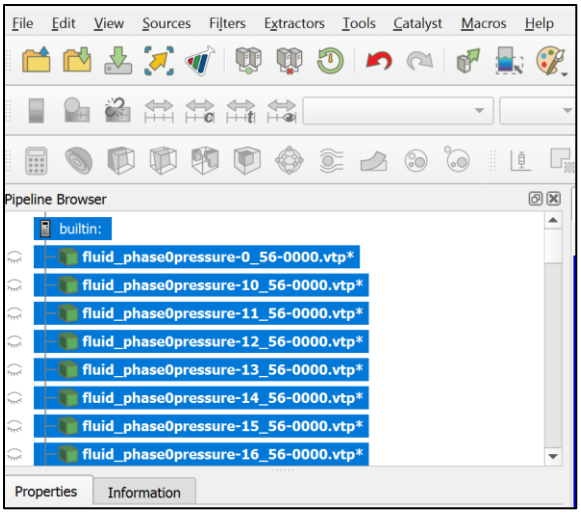


Figure 3.1 Exporting .vtp file to ParaView.

After selecting the .geo format and choosing 'save as', a message like the one shown in Figure 3.2 will appear. Make sure to check 'Choose Arrays To Write' and 'Write Time Steps', and then the steps to export to Houdini format are complete.

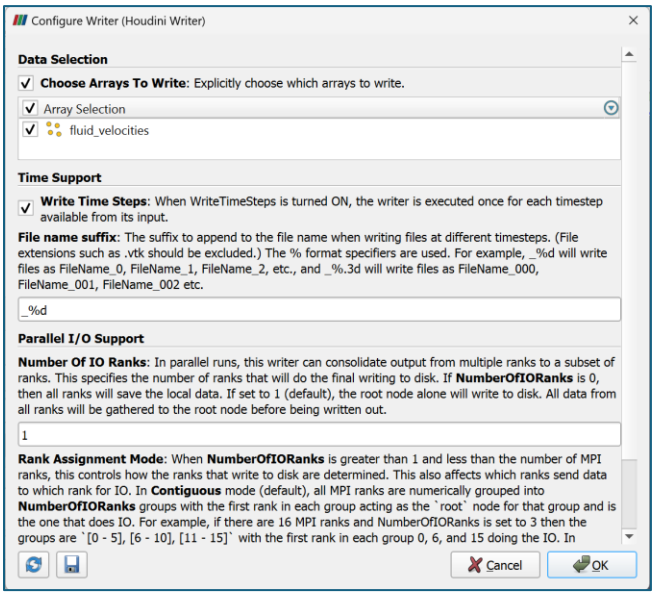


Figure 3.2 Houdini writer on Paraview.

IV. OPERATIONAL WORKFLOW II: HOUDINI

A brief overview of the use of nodes in Houdini reveals that nodes are the fundamental building blocks of projects. They are utilized to create, manipulate, and define the behavior of various elements within a simulation. Each node performs a specific function, and they can be combined in a network to create complex effects and simulations.

Houdini has several types of nodes, each belonging to different categories based on their function:

a. Geometry Nodes (SOPs - Surface Operators): These nodes operate on geometry, allowing for the modeling of shapes, definition of surfaces, and manipulation of geometrical data.
b. Dynamics Nodes (DOPs - Dynamics Operators): These nodes are used for simulations involving physical behaviors like fluids, smoke, fire, and more complex dynamics like rigid bodies or grains.
c. Shader Nodes (SHOPs - Shader Operators): Shader nodes are used to define the appearance of surfaces, volumes, or lights through materials and shaders.
d. Composite Nodes (COPs - Composite Operators): These nodes are used for compositing and image processing tasks within Houdini.
e. Channel Nodes (CHOPs - Channel Operators): CHOPs are used for motion editing and audio processing, allowing you to refine keyframe data and manipulate audio tracks.
f. Render Nodes (ROPs - Render Operators): These nodes control the rendering process, including managing render settings, defining render passes, and writing output files.

After all files from ParaView have been converted to the .geo format, which can be opened in Houdini, it's time to create a new project in Houdini. To start the simulation, the first step generally involves creating a geometry node at the obj level. Regarding levels in Houdini, the concept of "levels" refers to the different contexts or environments where nodes operate. These levels organize nodes for working with data and operations, facilitating the management and manipulation of complex elements within a project. In this simulation setup, the two levels that will be frequently used are the OBJ Level (Object Level) and SOP Level (Surface Operator).

- **Object Level (OBJ)**: This is the topmost level in Houdini, serving as the container for scene objects. This level involves the management of various object nodes that represent entities in a 3D scene, including cameras, light sources, geometry containers, and more. At this level, the primary tasks involve the positioning, scaling, and parenting of different objects. At the OBJ level, different elements, such as the domain of the simulation, the emitters, and the colliders, are managed. Each of these elements is contained within their respective geometry nodes. For this simulation, the emitters and colliders **have already been programmed** into the MPM data itself, so there is no need for manual settings by adding nodes for emitters and colliders.
- **Surface Operator (SOP) Level**: This level is situated **within** geometry object nodes at the OBJ level and is where geometry and attributes are worked on. SOPs are employed to create, edit, and manipulate geometry data. This stage hosts most of the modeling and procedural generation activities. In this simulation, within a geometry node, SOPs are utilized to define the initial state of the simulation, including the shape and distribution of fluids, and to convert point cloud data (MPM data) into a format suitable for simulation.

To open MPM data files, start by creating a geometry node at the OBJ level. For convenience, name the node for transferring MPM data as ‘water_MPM’. Ensure that both the blue and green flags on the node are enabled so that the MPM data can be viewed in the viewport. A brief note on the flag color system in Houdini nodes:

- **Green flag:** This flag indicates that the node is being displayed in the viewport. When the green flag is enabled (sometimes called the display flag), Houdini renders the output of that node in the viewport. This helps you to visualize the current state of your geometry or effects at that particular stage in the node chain.
- **Blue flag**: This is the render flag, which designates which node’s output is used for rendering operations outside of the viewport. When the blue flag is active, it tells Houdini to use the data from this node as the final output for renders. This can be different from what is currently displayed in the viewport (green flag).

Figure 4.1 Geometry node in Houdini with blue and green flags on.

4.1 Inputting MPM Data into Houdini

The first step is to create a node at the OBJ level. For this process, the node at the OBJ level is named ‘water_MPM’. In the context of the node, if both the green and blue flags are enabled, it indicates that this node’s output is being visualized in the viewport and is also used as the output for rendering.

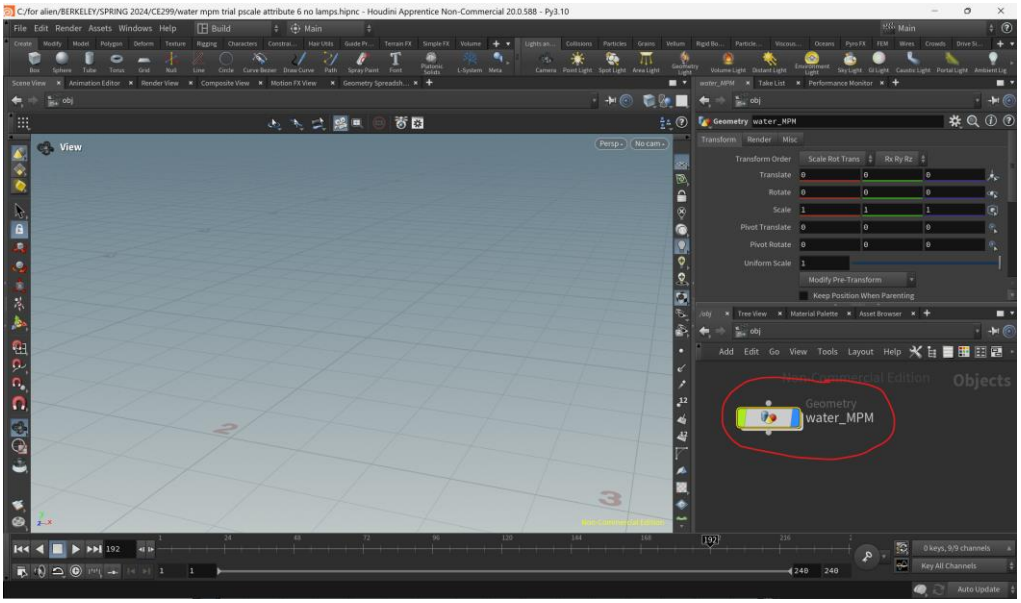


Figure 4.1.1 Placement of nodes in Houdini.

At the SOP level (within the geometry node 'water_MPM'), create a new node named **'File'** (named **'data_mpm'** in this simulation) to open the MPM data. Ensure that the blue flag is active on this node.

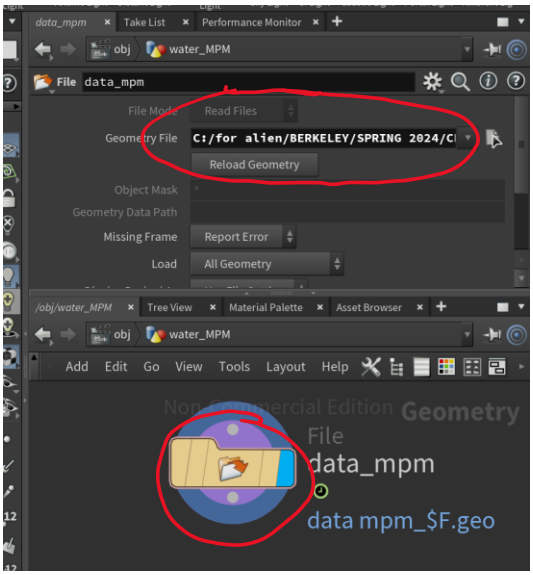
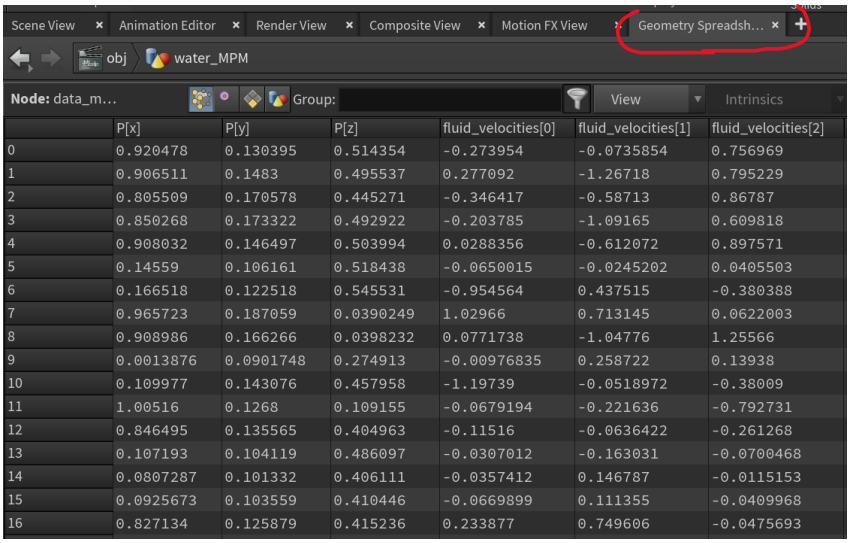


Figure 4.1.2 File node for opening MPM Data.

While activating this node, select the 'Geometry Spreadsheet' tab located above the viewport to view the parameters available in this MPM data.



	P[x]	P[y]	P[z]	fluid_velocities[0]	fluid_velocities[1]	fluid_velocities[2]
0	0.920478	0.130395	0.514354	-0.273954	-0.0735854	0.756969
1	0.906511	0.1483	0.495537	0.277092	-1.26718	0.795229
2	0.805509	0.170578	0.445271	-0.346417	-0.58713	0.86787
3	0.850268	0.173322	0.492922	-0.203785	-1.09165	0.609818
4	0.908032	0.146497	0.503994	0.0288356	-0.612072	0.897571
5	0.14559	0.106161	0.518438	-0.0650015	-0.0245202	0.0405503
6	0.166518	0.122518	0.545531	-0.954564	0.437515	-0.380388
7	0.965723	0.187059	0.0390249	1.02966	0.713145	0.0622003
8	0.908986	0.166266	0.0398232	0.0771738	-1.04776	1.25566
9	0.0013876	0.0901748	0.274913	-0.00976835	0.258722	0.13938
10	0.109977	0.143076	0.457958	-1.19739	-0.0518972	-0.38009
11	1.00516	0.1268	0.109155	-0.0679194	-0.221636	-0.792731
12	0.846495	0.135565	0.404963	-0.11516	-0.0636422	-0.261268
13	0.107193	0.104119	0.486097	-0.0307012	-0.163031	-0.0700468
14	0.0807287	0.101332	0.406111	-0.0357412	0.146787	-0.0115153
15	0.0925673	0.103559	0.410446	-0.0669899	0.111355	-0.0409968
16	0.827134	0.125879	0.415236	0.233877	0.749606	-0.0475693

Figure 4.1.3 MPM data's Geometry Spreadsheet.

After that, add a **'Time Blend'** node to interpolate MPM data results from a fluid simulation. This allows for a smoother and more realistic visual display of fluid animations, even if the stored data is sparse or infrequent. This node is crucial for optimal results in the visualization and rendering

of complex simulations, providing greater control over the final appearance of animations in Houdini.

Adding a '**Cache File'** node after a Time Blend node in Houdini is a strategic approach for managing and optimizing workflow efficiency, especially when dealing with heavy simulations like those using the Material Point Method (MPM). The node takes the output of upstream nodes (like simulations or heavy geometry calculations) and writes it to a specified file on the disk. This file is usually in a .bgeo.sc format for geometry or could be in other formats depending on the data type. Once the data is cached, the File Cache node can also read from the disk cache. This avoids the need to recompute the data and speeds up scene loading and interaction, which is especially useful in rendering or further refining stages.

A **'Null'** node is added after the **'file cache'** node to be used as an output node, and it doesn't perform any computational node. For instance, after setting up a simulation and caching it with a 'File Cache' node, placing a 'Null' node afterward can signal that this is the final output of a particular process or data stream. It is also used as a hook for rendering and visibility flags. This makes it easier to manage what gets rendered or displayed without altering the main operational nodes, keeping those processes clean and focused on their primary tasks.

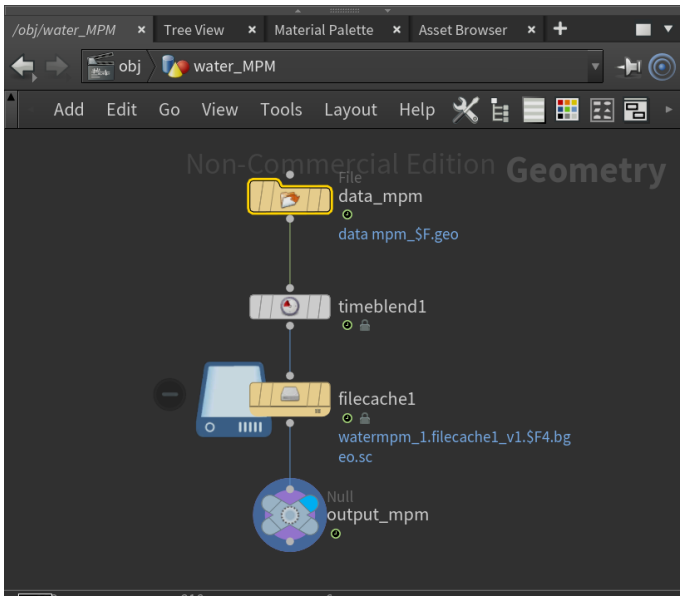


Figure 4.1.4 Nodes on water_MPM's SOP level.

The MPM data opened in Houdini appears on the viewport, as shown in **Figure 4.1.5**. In the raw MPM data, water is represented by colorless particle grains.

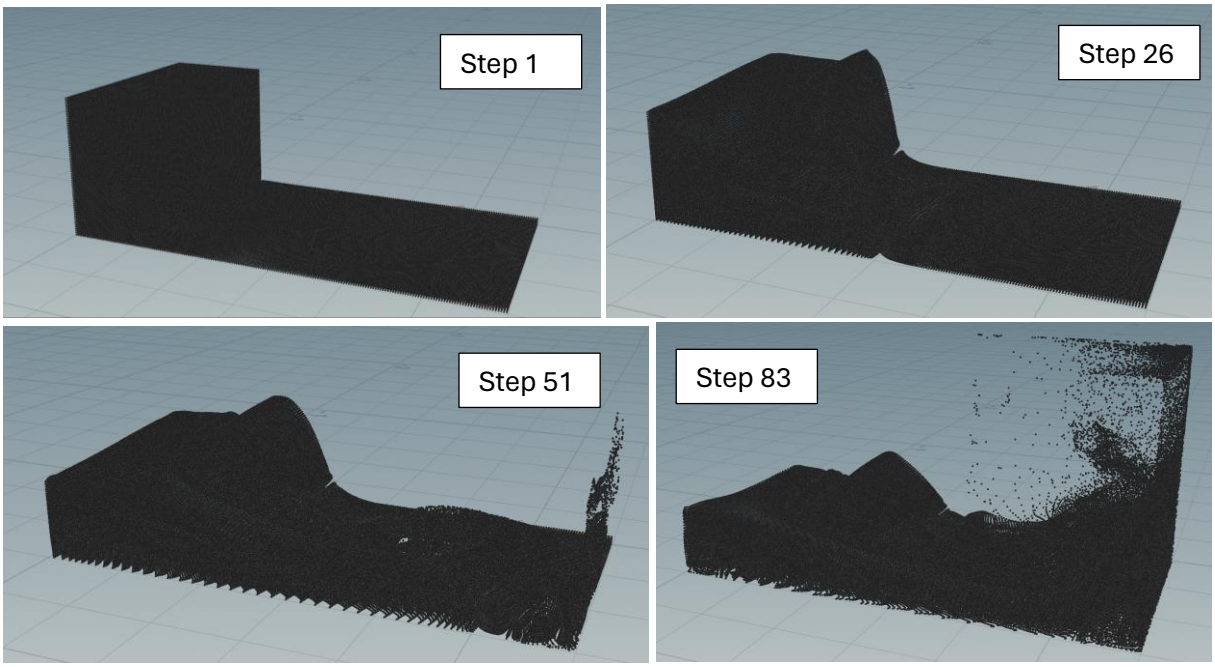


Figure 4.1.5 Raw MPM data simulation.

4.2 Creating a Barrier Block Composed of Crushed Rocks

The initial step always involves creating a node at the OBJ level. In this process, the node for creating this block of rocks is named **'box_crushed_rocks'**. To adhere to the design shown in **Figure 2.1** in section II, the crushed rocks must be 1.59 cm in size and densely packed to form a barrier block that obstructs the right side of the dam. The block to be created will follow the dimensions in the design, measuring (0.3 x 0.3 x 0.6) m.

4.2.1 Creating a Basic Box

At the SOP level, within the **'box_crushed_rocks'** node, create a **'box'** node and then a **'transform'** node. Adjust the size of the box by modifying the figures in the translate, rotate, and scale settings of the transform node. Always use a 'transform' node to change the scale and location of any object and avoid directly altering its size in the 'box' node.

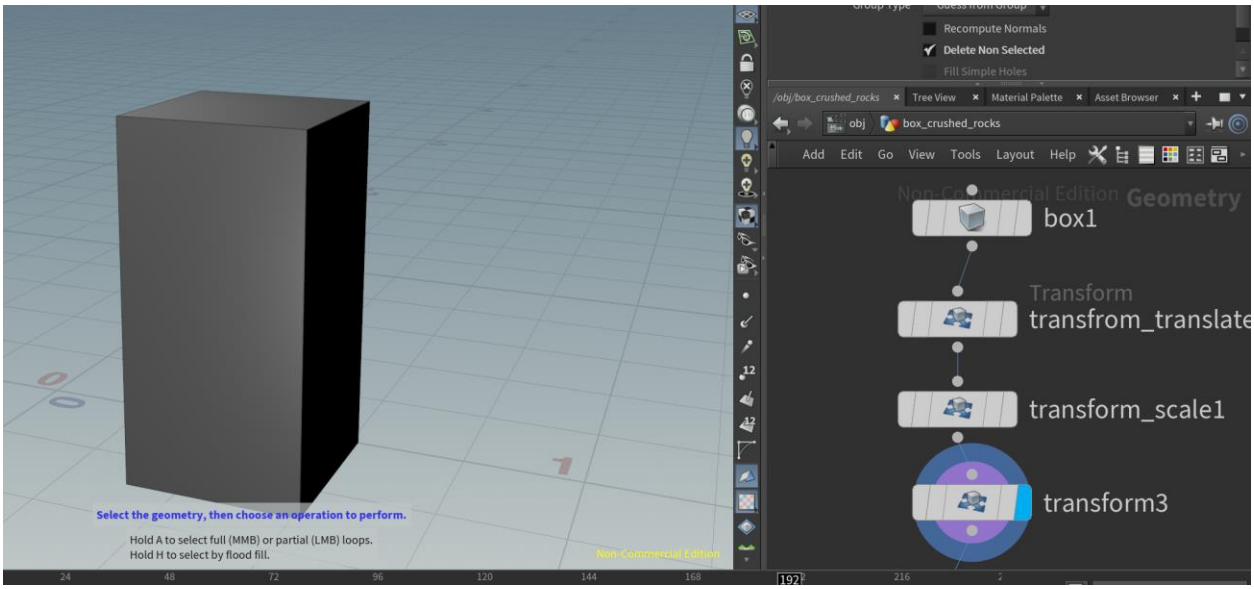


Figure 4.2.1.1 Basic box with 'transform' node.

4.2.2 Randomizing Particles in the Barrier Block

After creating the basic box, the barrier at the right side of the dam consists of densely packed crushed rocks with a diameter of 1.59 cm. To make it look realistic, the **'scatter align'** node distributes random points across the surface of the crushed rock block. This node spreads points that will serve as the base for placing smaller crushed rocks. Remember to adjust the scale radius setting in this node to half the diameter value (1.59 cm / 2). All measurements in Houdini are in meters, so remember to convert all units to meters.

Next, place a **'point jitter'** node after the 'scatter align'. This node adds variation to the positions of the points to enhance realism. The jitter adjusts the positions of random points to avoid an overly uniform or artificial appearance. After this, use the **'attribute randomize'** and **'attribute wrangle'** nodes. These nodes are used to add and modify necessary attributes on points, such as scale or orientation. **The 'Attribute randomize' node** might be used for randomizing attributes like **'pscale'** for the size variation of the rocks, while **'attribute wrangle'** could be used for further adjustments on the generated or new attributes.

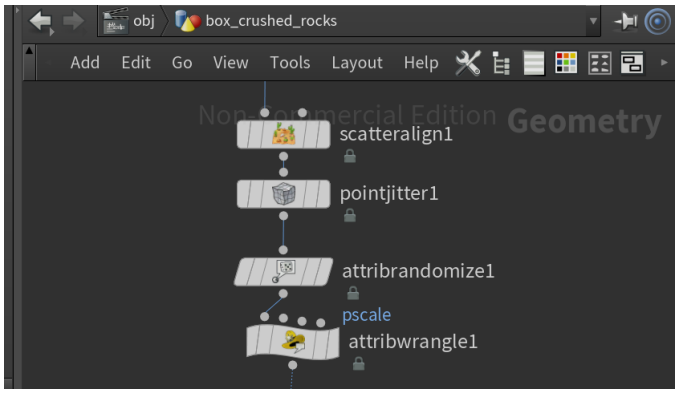


Figure 4.2.2.1 A set of nodes to randomize particles.

'Pscale' is an attribute commonly used in particle simulations and procedural modeling workflows, particularly when dealing with instances of geometry or other particle-based effects. Point attributes are data values stored at each point in a geometry. These attributes can store a wide variety of data types, including floats, integers, vectors, and strings. They are a fundamental aspect of how Houdini handles geometry and are crucial for driving simulations, procedural modeling, rendering, and more. The term **'pscale'** stands for "particle scale" and is a critical attribute for controlling the size of particles or instances at each point in a geometry.

Thus, the **'Attribute Randomize'** node is specifically designed to quickly apply random variations to attributes across points, primitives, vertices, or other elements within a geometry. It is commonly used to randomize attributes such as **'pscale'** or any custom attributes that need to be randomized. This node is ideal when a quick and easy method is required to apply randomness to attributes without the need for coding. However, in this simulation, an **'Attribute Wrangle'** is added after the 'Attribute Randomize' to enhance realism.

The **'Attribute wrangle'** node is ideal for more complex attribute manipulations where specific conditions or more elaborate computations are required. If it is needed to achieve more randomness, 'attribute wrangle' with VEX scripting is a way to go. 'Attribute wrangle' uses VEX (vector expression), a scripting language in Houdini designed specifically for performance and efficiency in handling large amounts of data. VEX has a syntax reminiscent of C, making it somewhat familiar to those with experience with C-like languages.

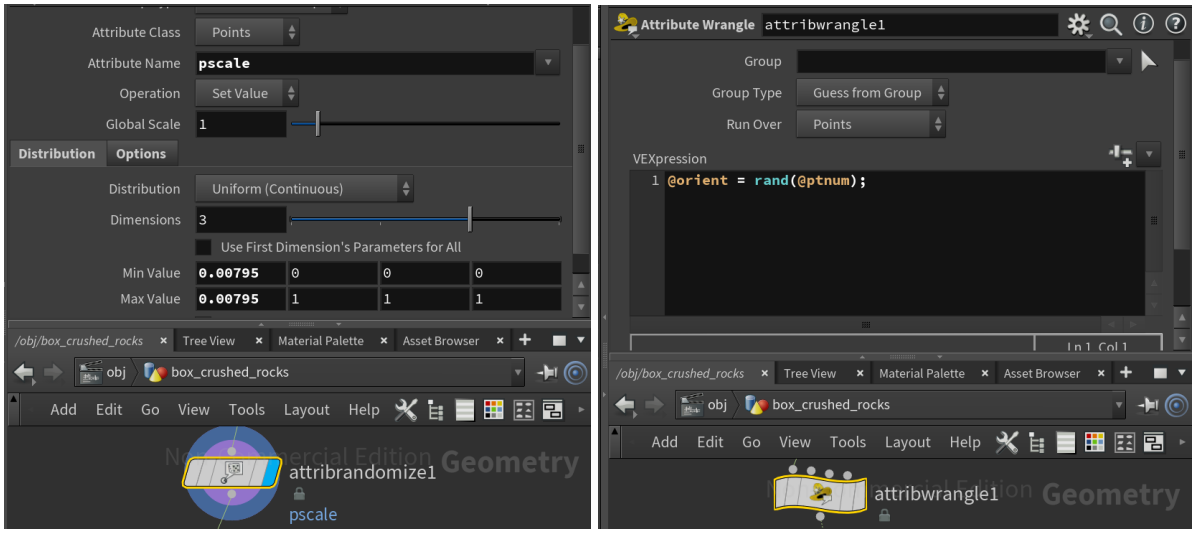


Figure 4.2.2.2 Attribute Randomize (left) and Attribute Wrangle (right).

After applying the 'Attribute Randomize' and 'Attribute Wrangle', the block will take the form seen in the image below. It is evident that the block, which was originally solid, now comprises a densely packed collection of particles.

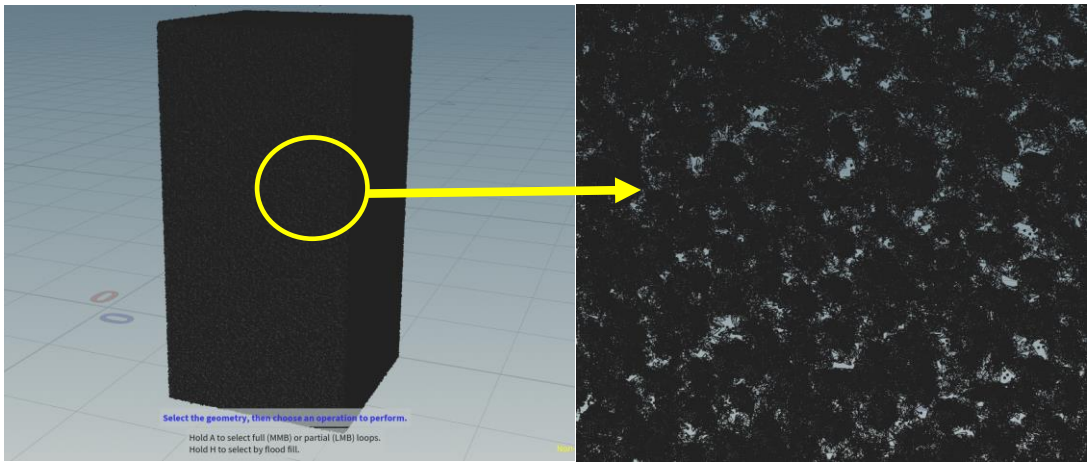


Figure 4.2.2.3 Barrier block after randomized nodes applied.

4.2.3 Creating the Shape of Crushed Rocks

A simple form is actually sufficient to create the shape of the rocks without using excessive memory. Start by creating a **'sphere'** node as the basic shape of the rock fragments. Adjust the scale and radius settings to achieve a form similar to that shown in Figure xx. After that, add a **'rbd material fracture'** node, which is useful for creating natural-looking cracks on the modified sphere. The settings on the **'rbd material fracture'** node do not need to be altered, as the default settings already provide cracks that look natural.

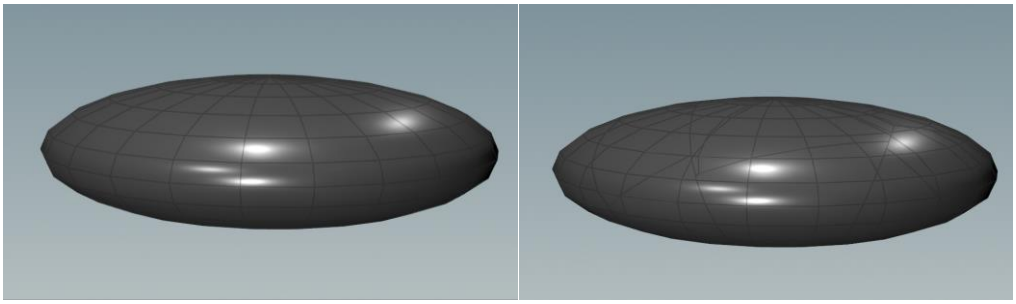


Figure 4.2.3.1 Sphere before and after fracture node is being applied.

After that, the next step is to add a 'blast' node to obtain the shape for one piece of crushed rock. Typically, this node automatically appears if a part of the object is deleted. For this process, one part of the already fractured sphere will be deleted, and the 'blast' node will automatically appear. After the red part in **Figure 4.2.3.3** is removed (by pressing the 'delete' key on the keypad), the blast node will appear. In the 'blast' node, it is important to check the 'Delete Non Selected' option to retain the part that was originally deleted but is intended to be used as the shape of the rock fragment.

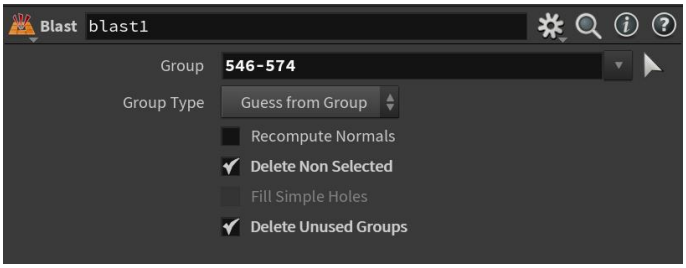


Figure 4.2.3.2 Blast node settings.

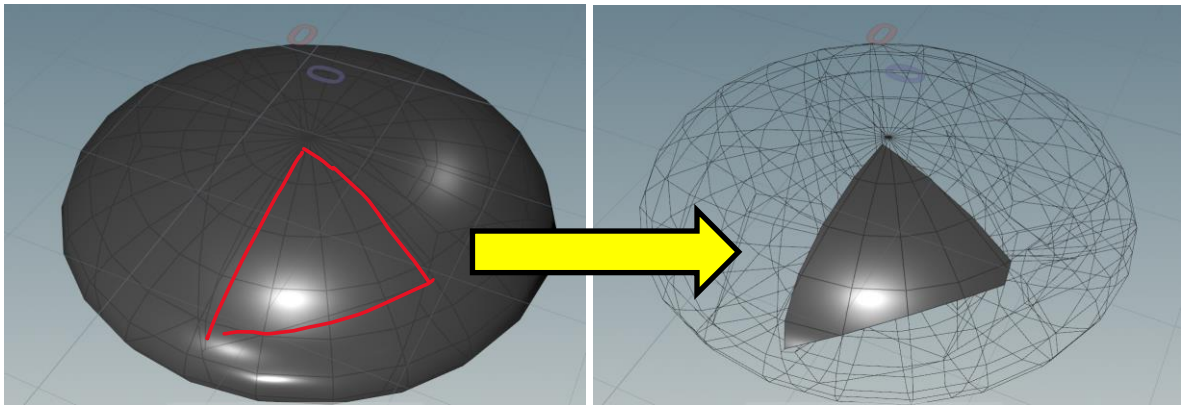


Figure 4.2.3.3 Sphere after blast node is implemented.

After completing the crushed rock shape, it's time to combine everything into one. Use the **'copy to points'** nodes to merge the side node for creating the crushed rocks shape (left side) with the node for creating a random placement (left side). Add a **'null'** node at the end of the node chain to display all nodes in the viewport.

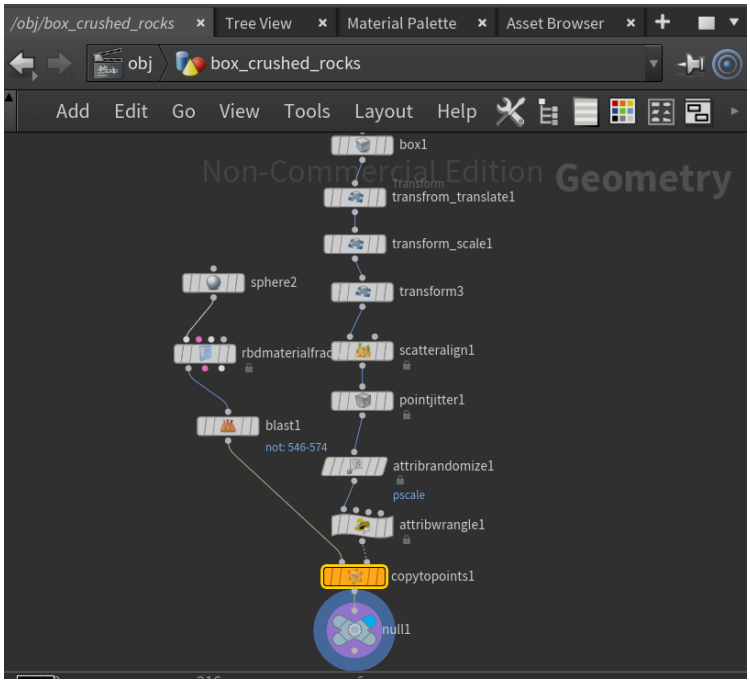


Figure 4.2.3.4 Whole nodes chain in 'box_crushed_rocks' geometry node.

The barrier block of rocks will appear in the scene view as shown in the image below. The particles in the block have transformed into more realistic gravel-like shapes.

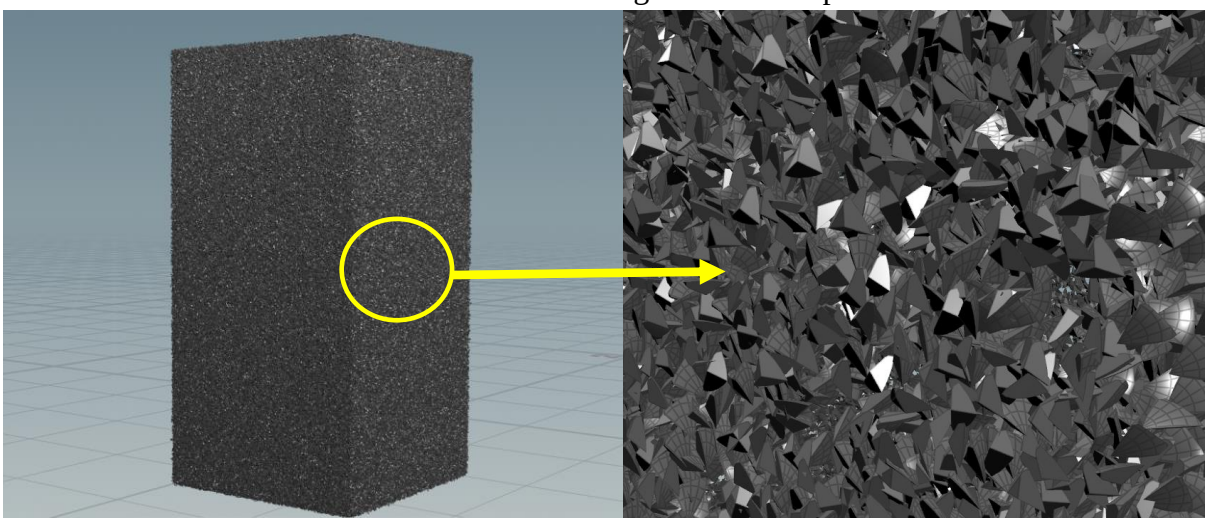


Figure 4.2.3.5 Barrier block with gravel of rocks.

4.3 Creating a Water Flow Simulation

The next step is to create a new geo node at the OBJ level (in this process, this node is named **'visualize_water'**). This step is essential for creating water animations that replace the uncolored water particle grains, which appear less appealing and aesthetically pleasing. Then, at the SOP level, create the first node, which is an **'object merge'**. The function of the **'object merge'** node

is to retrieve MPM data from the first created geo node, namely from the **'water_MPM'** node (refer to Section 4.1). Whenever data is needed from a process, take it from the **'Null'** node at the SOP level within the 'water_MPM' node. It is important to remember that since the raw MPM data already simulates water movement, there is **no need to use FLIP sim** from the beginning again, as the data for the water and how it moves already exists.

4.3.1 Creating Point Attributes

Next is the use of an **'attribute wrangle'** node. This node has been previously used to create random placements for crushed rocks in the previous subsection. The functionality of the 'Attribute Wrangle' node depends heavily on the VEX script written within it. It can be used for various purposes in different contexts, such as editing, creating, or manipulating attributes in simulations or modeling. As previously used to set random positions for crushed rocks, this node can adjust the position, orientation, or scale of objects based on certain logic or random data sources.

For these water simulations, **'attribute wrangle'** is utilized to adjust the position, orientation, or scale of objects based on certain logic or random data sources by implementing the VEX script. This VEX script processes fluid simulation data by extracting the velocity vector from the **fluid_velocities** (from MPM raw data) attribute and converting it into a scalar speed value. It calculates the magnitude of the velocity vector to determine the speed of each fluid particle, providing a simple measure of how fast each particle is moving. This scalar speed is then stored in a new attribute **'v'**, which can be utilized for visualizing, analyzing, or conditioning behaviors based on the speed of particles in the simulation. This water simulation now has seven attributes ready to be modified for the next nodes.

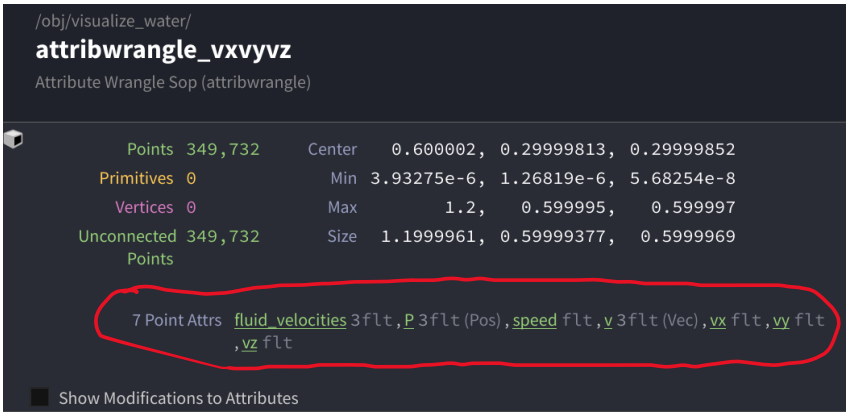


Figure 4.3.1 Point attributes for water simulation.

4.3.2 Water Simulation Nodes

The next step is to add a **'Fluid Compress'** node. The 'Fluid Compress' node is primarily utilized to optimize fluid simulations by reducing the memory footprint and potentially increasing the calculation speed. This node focuses on memory optimization and is designed to manage the

memory usage of fluid simulations by selectively compressing particle data. This is particularly useful in large-scale fluid simulations, where managing many particles can be memory intensive. By compressing the fluid data, the node can help speed up the computation times for subsequent operations, as there is less data to process.

After ensuring the fluid data is compressed, the next node is the **'Particle Fluid Surface'** node. This node is one of the most crucial for creating water simulations, and many parameters need to be carefully considered as they are instrumental in converting a particle-based fluid simulation into a mesh that can be rendered. This node takes the raw particle data and uses it to create a surface by interpolating the particles into a smooth, continuous mesh. Figure xx shows the parameter values selected for creating the water simulation in this process.

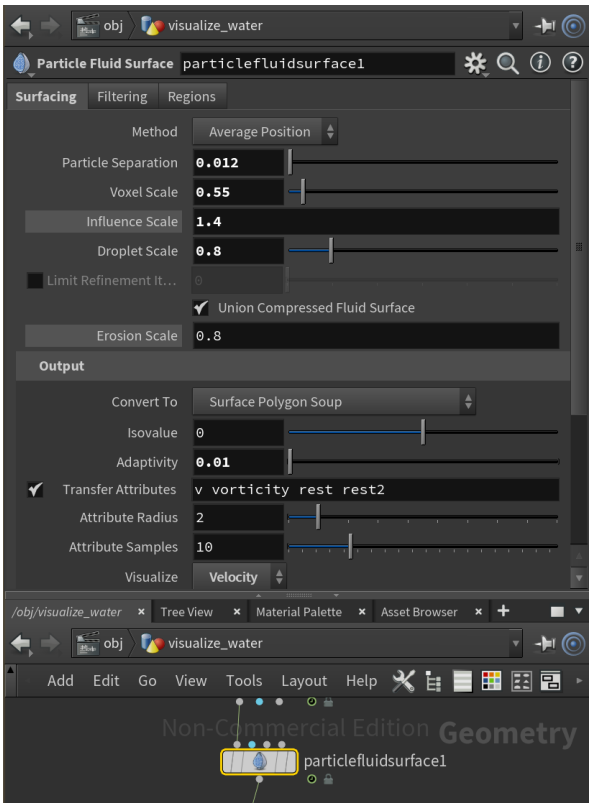


Figure 4.3.2.1 'Particle Fluid Surface' parameter setting.

A brief explanation of the parameters modified that significantly affect the shape of the water simulation:

- **Particle separation**: This sets the target world space distance between particles when generating the surface. Smaller values result in a finer, more detailed mesh but increase computation time and memory usage. A value of 0.012 was chosen as it indicates a high-resolution mesh suitable for detailed splashes and intricate water surface details.
- **Voxel scale**: The concept is similar to a pixel, but a voxel denotes a point in three dimensions, whereas a pixel has two dimensions. A lower value increases the resolution,

capturing finer details of the fluid's surface. A setting of 0.55 provides a balance between detail and performance.

- **Influence scale**: Adjusts the radius of influence for each particle, affecting how much each particle impacts the surface calculation. An influence scale larger than 1.0 can make the surface appear smoother and thicker, which can be useful for creating more cohesive liquid surfaces.
- **Droplet scale**: This parameter adjusts the size of detected droplets relative to the particle separation. Increasing it helps retain smaller splashes and droplets in the final mesh. A setting of 0.8 helps maintain finer splash details without overly fragmenting the water surface.
- **Erosion scale**: Used to erode the fluid surface, simulating the natural thinning of fluids due to spreading and splashing. An erosion scale of 0.8 moderately reduces the thickness of fluid layers, adding realism to the simulation by preventing an overly bulky fluid appearance.
- **Adaptivity**: This parameter reduces the number of polygons in areas where less detail is needed, optimizing the mesh without significant loss of detail. An adaptivity setting of 0.01 helps in reducing mesh complexity while preserving essential details.

It is highly recommended to always check the visual results of the water flow in the viewport to see if it meets expectations, as incorrect settings can lead to very unrealistic and coarse water appearances.

The next step is optional, which involves adding a **'subdivide'** and a **'smooth'** node. These are used to refine and enhance the geometry of a surface, typically in the context of fluid simulations where a higher level of surface detail is desired. The **'Subdivide'** node increases the geometry's resolution by subdividing the faces of the mesh. It operates by splitting each polygon into smaller polygons, increasing the mesh density, which adds more detail to the geometry. The **'Smooth'** node adjusts the positions of vertices in the geometry to make the surface appear smoother. It averages the positions of vertices based on their neighbors, which reduces sharp angles and makes the surface look more refined. If the settings in the 'particle fluid surface' node are deemed sufficient, these two nodes are not necessary.

The next step is to add a **'UV texture'** node, which assigns UV coordinates to the geometry, which is essential for texture mapping in 3D models. UV mapping is the process of projecting a 2D image onto the surface of a 3D model. Once UV is mapped, materials can be applied to this water. The following step is to add caching to speed up the workflow in Houdini. The final step, as usual, is to add a **'Null'** node.

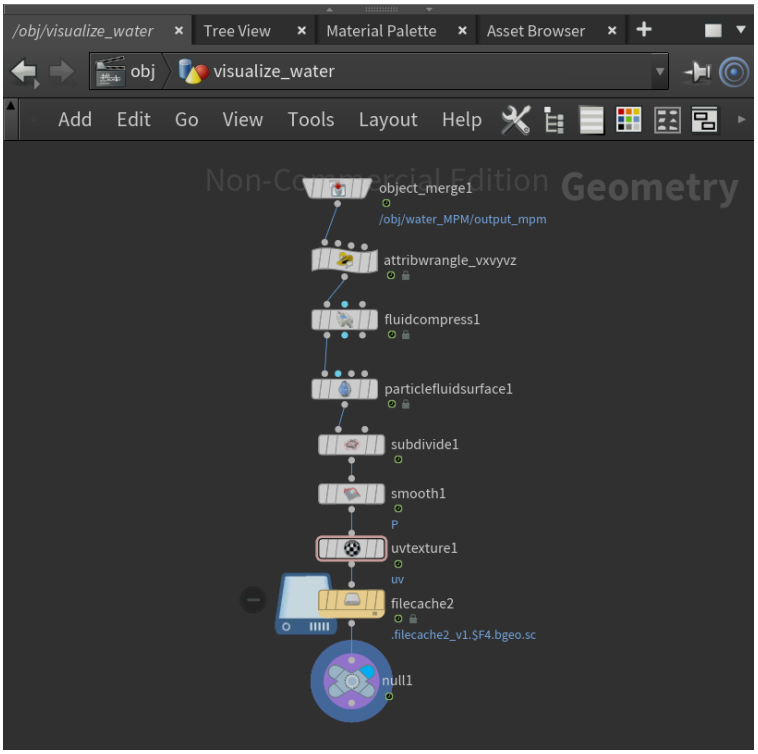


Figure 4.3.2.2 Whole nodes chain in 'visualize_water' geometry node.

4.3.3 Preliminary Result

So far, three main elements have been created at the OBJ level: **water_MPM, box_crushed_rocks**, and **visualize_water**. With these three nodes at the OBJ level, the form of a real dam consisting of crushed rock blocks and water is already beginning to be visible in the viewport. However, for rendering purposes, materials must be used to add color and texture to the model.

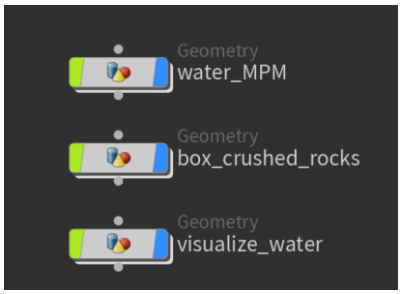


Figure 4.3.3.1 Three basis nodes for simulation.

Below is an example of several steps from the results so far, as seen in **Figure 4.3.3.2** in the viewport.

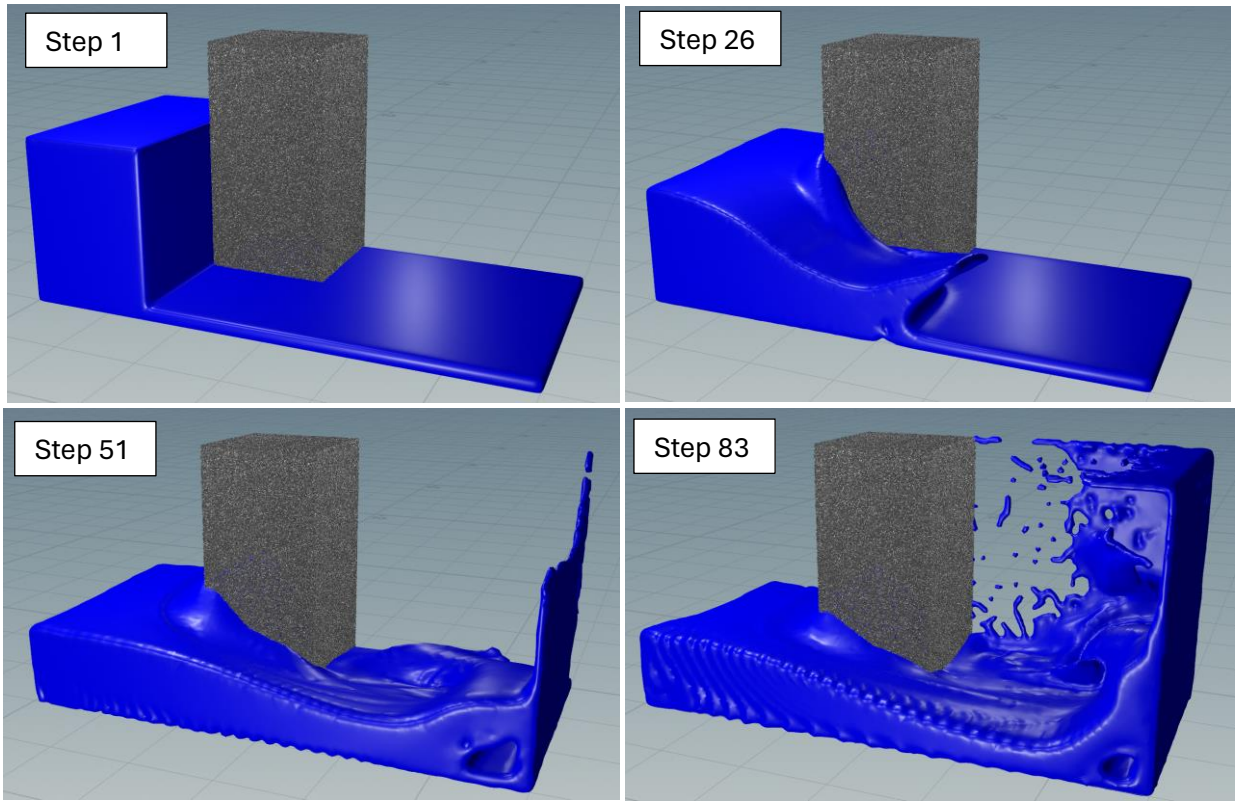


Figure 4.3.3.2 Several steps of simulation before applying material for texture and color.

4.4 Application of Materials to Add Color and Texture to the Model

The process of adding color and texture is no longer carried out at the OBJ level but at the **MAT level** (material). The material nodes for crushed rocks will use a **'principal shader'**, but the water simulation will employ two types of materials, namely **'uniform volume'** and **'basic liquid'**. In practical use, a **'Uniform Volume'** is employed when simulating a body of water with varying depths and aiming to capture the light interaction within the water (such as light fading out in deeper water or cloudiness). For a scene focusing on the surface of the water, where clarity, reflections, and texture (like waves) are important, **'Basic Liquid'** is the way to go. Since this simulation will display variations in water depth as well as clear water reflections, both material nodes will be applied.

The settings for color and several parameters in the materials are found in figure xx. However, color settings are subjective and can be adjusted according to individual preferences..

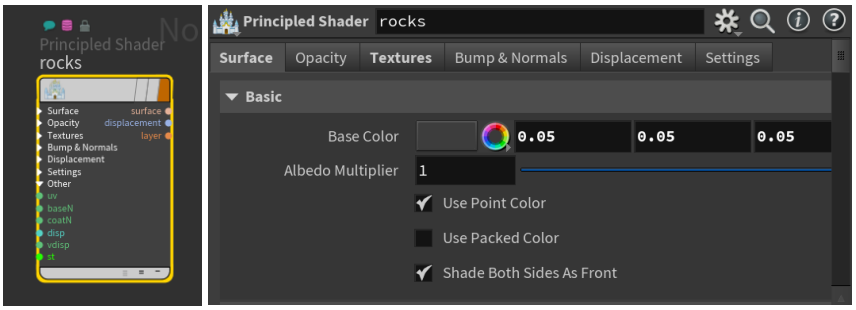


Figure 4.4.1 Principal shader material for crushed rocks.

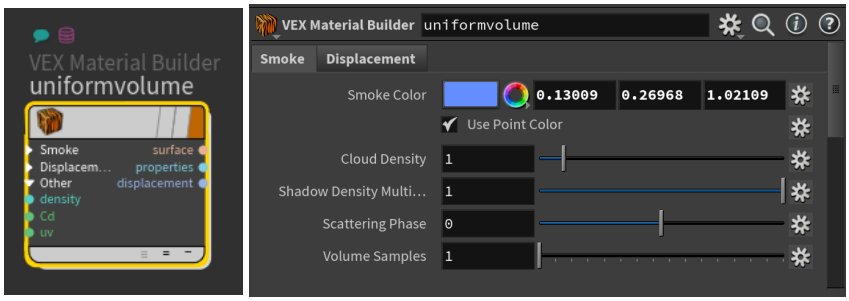


Figure 4.4.2 Uniform volume for varying water depth color.

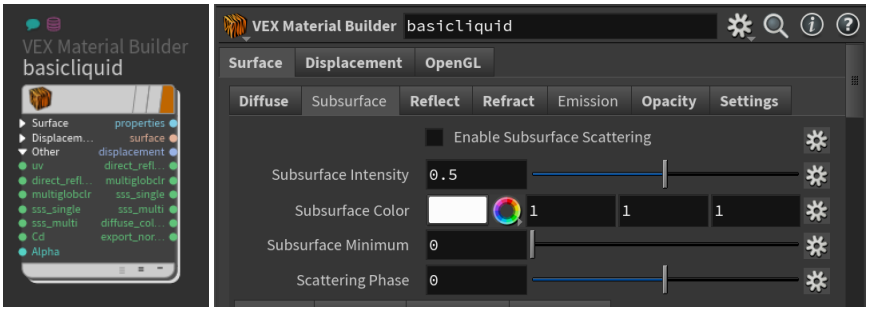


Figure 4.4.3 Basic liquid material for water reflection and clarity.

4.5 Lighting and Camera Configuration

Proper lighting settings are crucial because even a slight change in the direction of light can significantly alter the colors produced during rendering. The number of lights and the type of lighting are also critical, as Houdini offers various kinds of lighting options, such as Spot light, Point light, Ambient light, Environment light, and many more. Each type of light serves a specific purpose in terms of how it affects the scene and how it interacts with materials and surfaces. Here are some of the lighting types used for rendering:

A. **Point Light**: emits light in all directions from a single point in space. It is akin to light bulbs, radiating light equally in all directions.
B. **Spotlight**: emits a cone of light from a specific position and direction, allowing control over the angle of spread (cone angle) and the falloff of the light.

C. **Ambient Light**: provides non-directional, soft illumination that uniformly illuminates all scene parts without casting shadows. It simulates a combination of direct and reflected light scattered so much that it appears to have no discernible source.
D. **Environment Light**: simulates realistic environmental lighting by emitting light based on the texture from all directions.

For lighting direction, the best advice is to experiment to find the best angle while consistently rendering continuously. This process can be quite time-consuming to truly pinpoint the most suitable direction, especially to achieve the best coloration of the water.

The most crucial aspect for rendering is the camera settings. While not a light source, the camera's position and orientation are essential for determining how the scene is viewed and how the lights affect the visual outcome. It captures the interaction of all these lights with the scene from its perspective. The view displayed in the viewport from the camera is what will appear when the model is rendered. Changes in camera position can also affect the texture and color displayed in the final output.

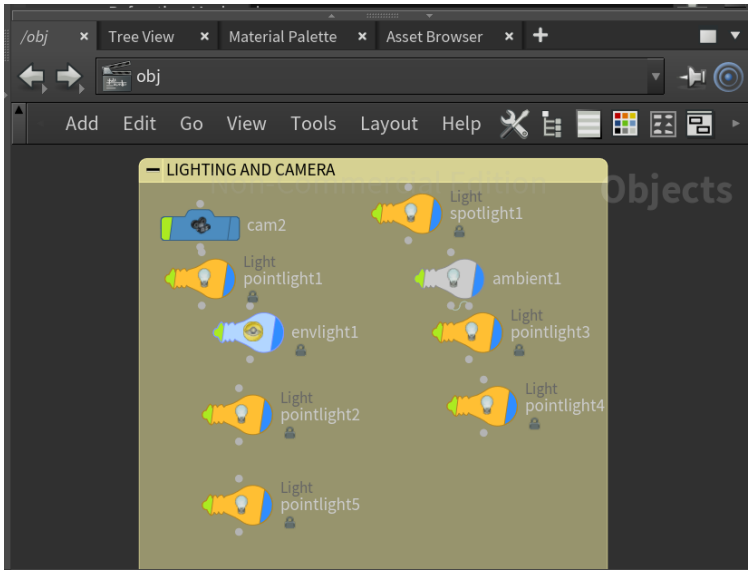


Figure 4.5.1 Lighting and Camera Nodes.

4.6 Rendering Process

The rendering process will be conducted using the **Mantra render engine**. This process is carried out at the **OUT level**, not at the OBJ level. At the OUT level, use the '**Mantra'** node and adjust any parameters that might need to be changed. The most important parameters to focus on are the start/end frame, camera selection, the location for saving the rendered output (as an image), and the rendering method. In this process, the method used is ray tracing. Other methods, such as path tracing, can produce more accurate lighting and shadows, but they require a longer rendering time.

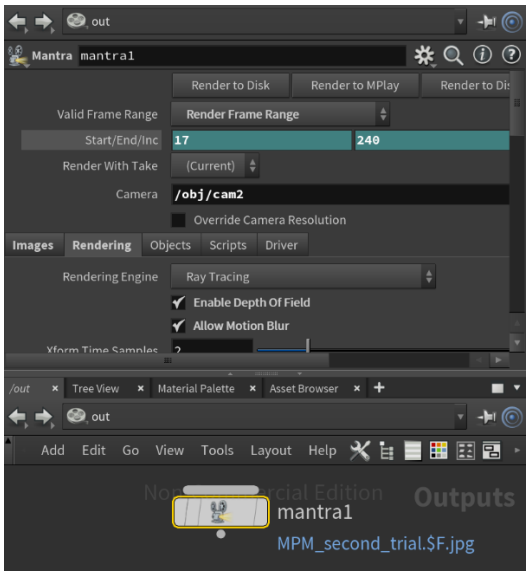


Figure 4.6.1 Mantra rendering settings.

For a better display during rendering, using the **'Render to Mplay'** option is recommended. However, for a quick check of the approximate results, visiting the **'Render View'** tab located above the viewport is advised. Activation of the blue flag on the node intended for rendering is essential.

A useful tip to accelerate the rendering process involves creating a new node separate from where the initial work was conducted. This simplifies the application of colors and textures from materials. At the SOP level, the node designated for rendering will comprise only the **'object merge'**, **'material'**, and **'null'** nodes. The method involves using the 'object merge' node to pull data from the raw process node. Typically, to retrieve data, 'object merge' extracts data from the last node of the raw process nodes, which is the 'null' node. Therefore, it is crucial to always have a 'null' node at the end of every process. The SOP level on the node to be rendered will consist only of these three nodes, as displayed in **Figure 4.6.3**. Remember to deactivate the blue flag at the OBJ level of the raw process node and activate the blue flag on the render node only.

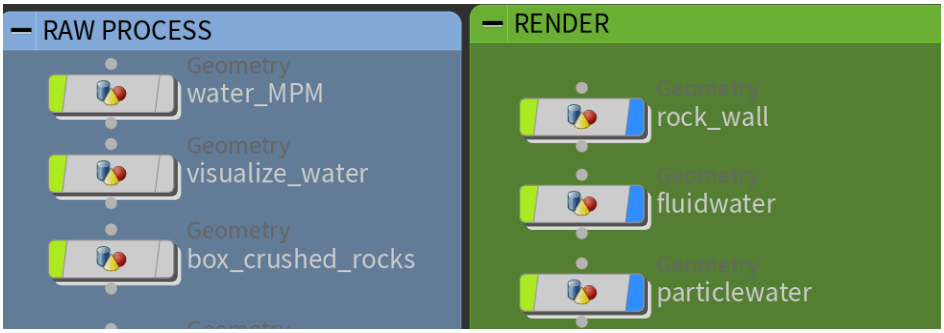


Figure 4.6.2 Group of Raw Process Nodes and Render Nodes.

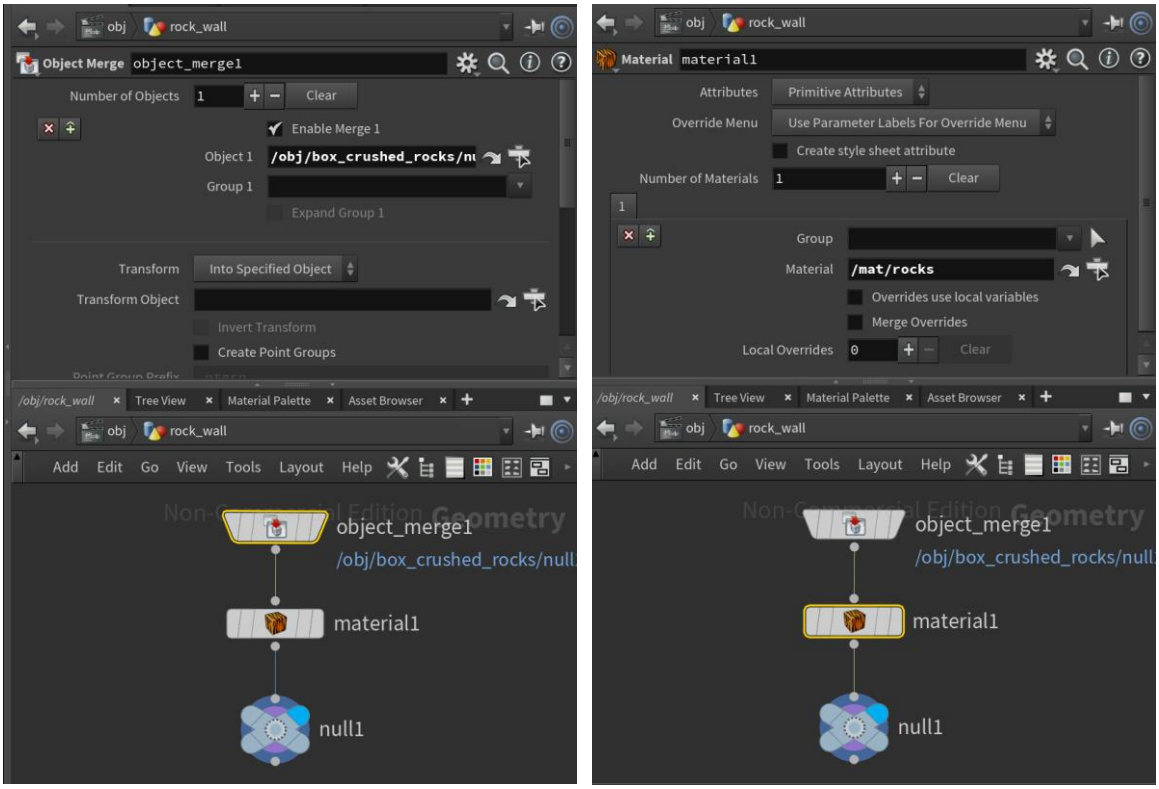
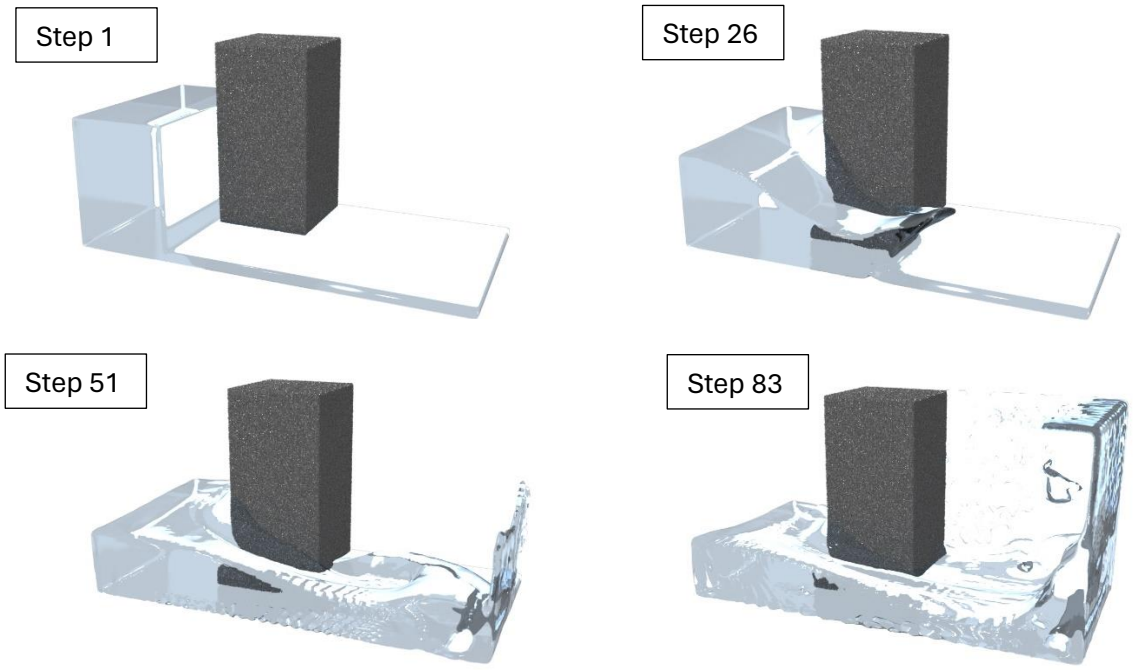


Figure 4.6.3 SOP level within Render node: 'object merge' (left) and 'material' nodes (right).

Below are some of the rendering results for the progress made so far, as shown in the images below.



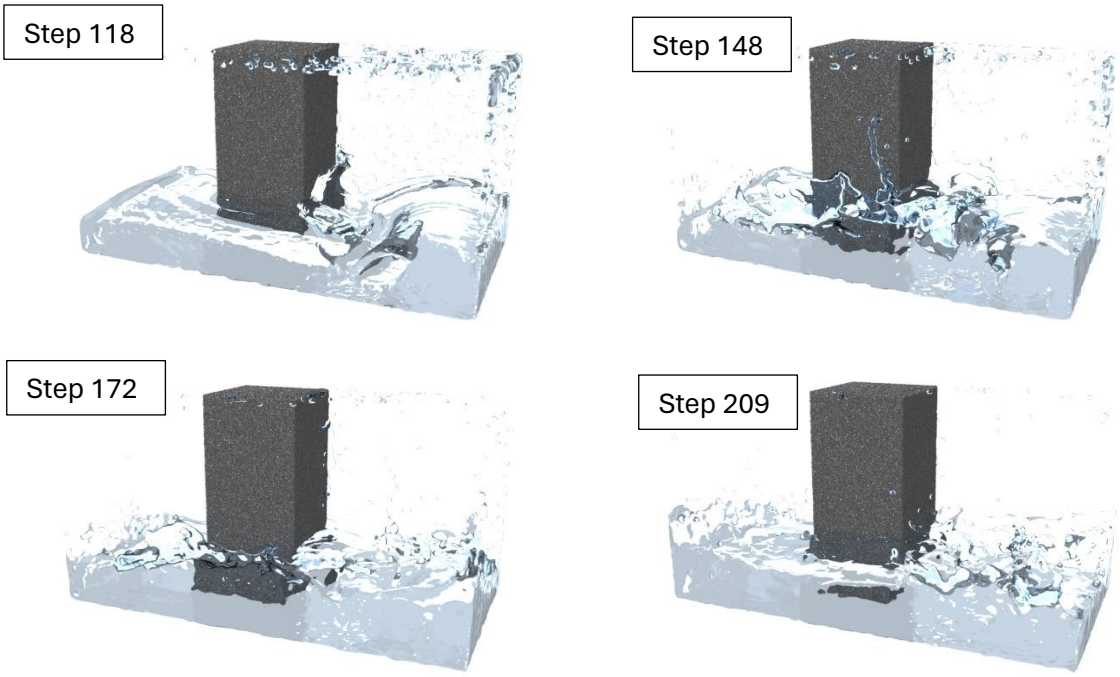


Figure 4.6.4 Rendered preliminary results step 1 – step 209.

V. OPERATIONAL WORKFLOW III: HOUDINI - ADDITIONAL

To enhance the aesthetics of the model, making it more beautiful and appealing, this simulation will include decorations such as installing a backdrop, creating glass akin to an aquarium, and adding a table, as if the dam is an aquarium placed on a table indoors.

5.1 Creating a Backdrop

Creating a backdrop is straightforward, requiring only the creation of a ‘grid’ node and adjusting its size and scale using a ‘transform’ node.

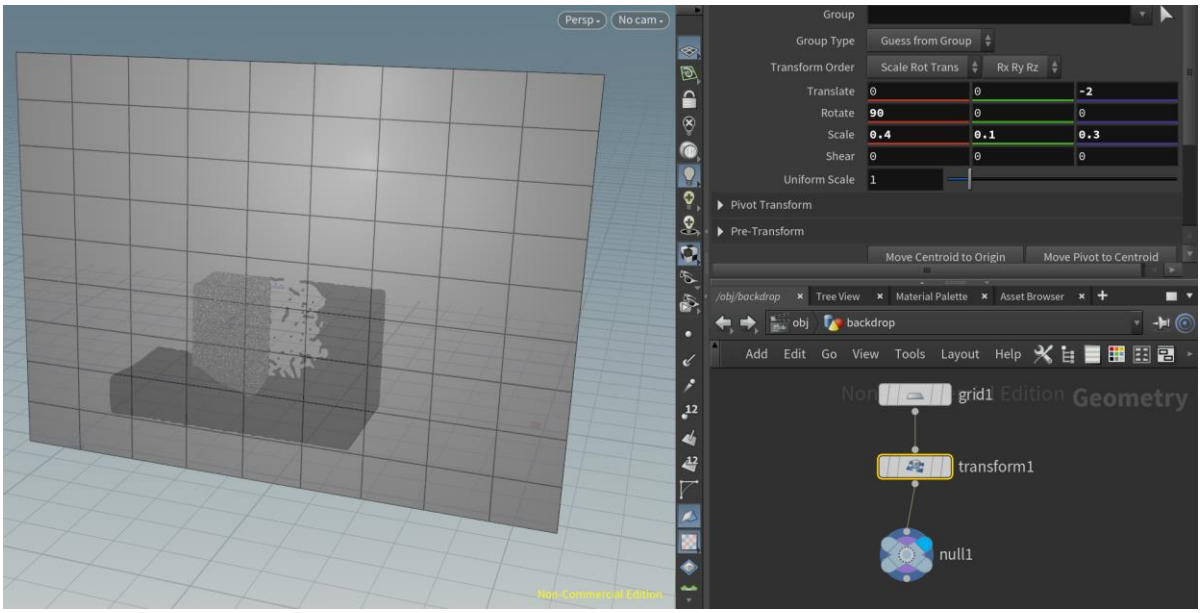


Figure 5.1.1 Backdrop as background.

5.2 Creating a Transparent 'Aquarium' Like a Box.

Creating glass similar to aquarium glass is not overly complex. The process begins with creating a 'box' node, which is then transformed in such a way that it forms a block encompassing the entire dam model. Then, the front part of the block is removed, and when deleting, a 'blast' node will automatically appear. To give thickness to the glass, use a **'polyextrude'** node. Adjust the thickness of the glass by changing the distance parameter.

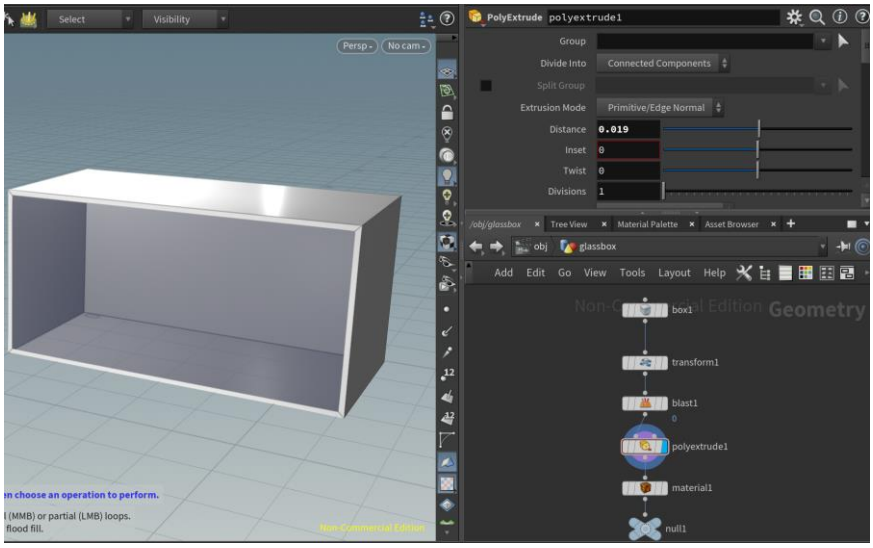


Figure 5.2.1 Making of the transparent box.

Setting up the material to create a texture and color is similar to aquarium glass can be quite challenging. The base material used is the 'principal shader'. However, several modifications need

to be made. Ensure to choose a **base color** that is somewhat dark, like dark gray. The **IOR** (Index of Refraction) value for the glass's shininess should be at least 1.5, but here it is set to 1.65 for a more reflective glass result. Since the glass surface is very smooth, set the **roughness** value as low as possible. Glass is a transparent object, so set the **transparency** value as high as possible. If you want the glass to be less see-through, adjust the transparency to a value around 0.9. To mimic the color of aquarium glass, which often has a slight green tint, set the **transmission color** to a very bright green, but choose a green that leans towards light turquoise. Another very important aspect is to set the **'at distance'** color, found under the 'transmission color' option, to a minimum value of 5. This setting ensures the green color in the aquarium does not appear too dense and looks natural. Other settings can be left at their default values.

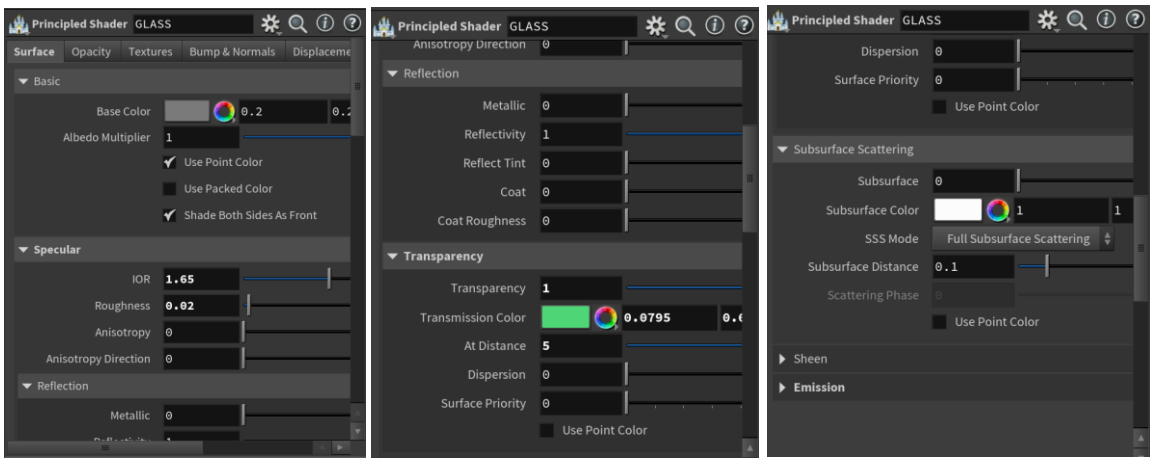


Figure 5.2.2 Principal shader settings for the transparent box.

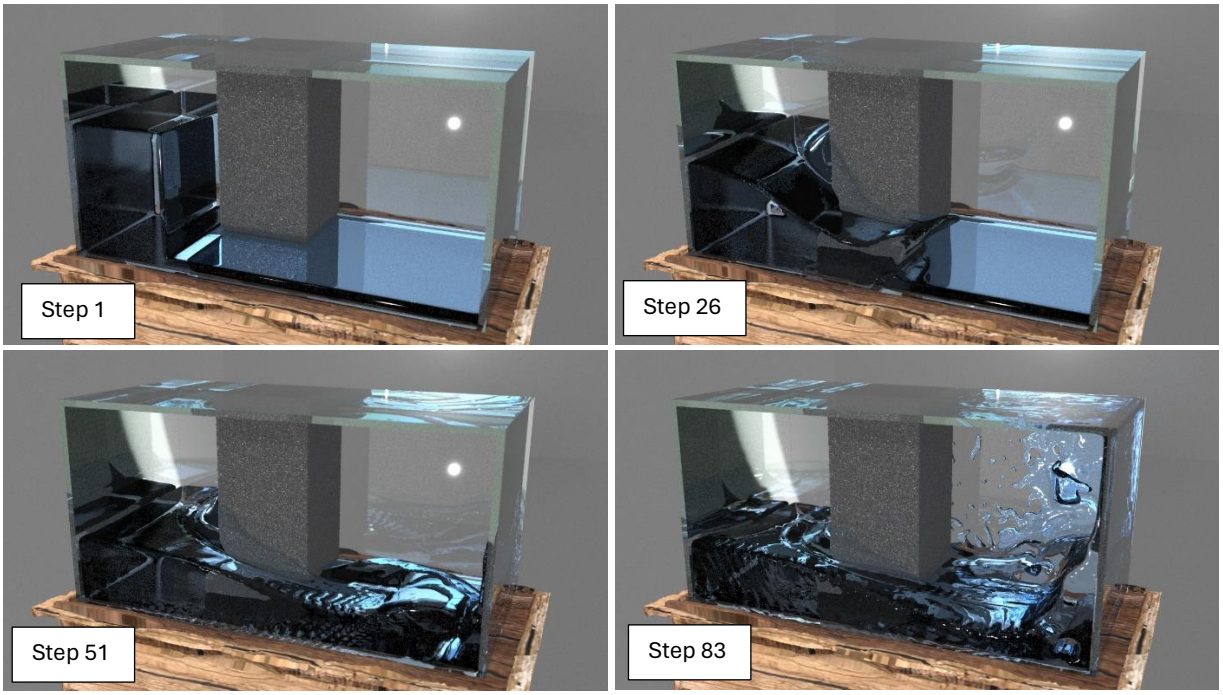
5.3 Creating a Drawer for Placing the Box

Lastly, to enhance the aesthetics, a drawer can be provided as a place to sit for the 'aquarium dam'. The type of table to be created is a drawer similar to those found in a bedroom. This can be constructed using several blocks stacked in a specific arrangement. Use the 'transform' node to adjust the scale and position until the desired shape is achieved. For parts that should not be too sharp, such as the surface of the drawer, the 'bevel' node can be used to make the edges of the drawer surfaces appear more natural. For easier texture applications, a pattern image can be searched on Google. The material can simply use a 'principal shader' and just insert a jpeg image from Google.

Figure 5.3.1 Drawer for placing the box.

5.4 Rendering Result After all Materials are Put Together

Here are 8 steps from the rendering results using the Mantra Rendering Engine after all aspects have been combined. The outcome appears more natural, resembling an aquarium box placed inside a room, providing a more calming impression.



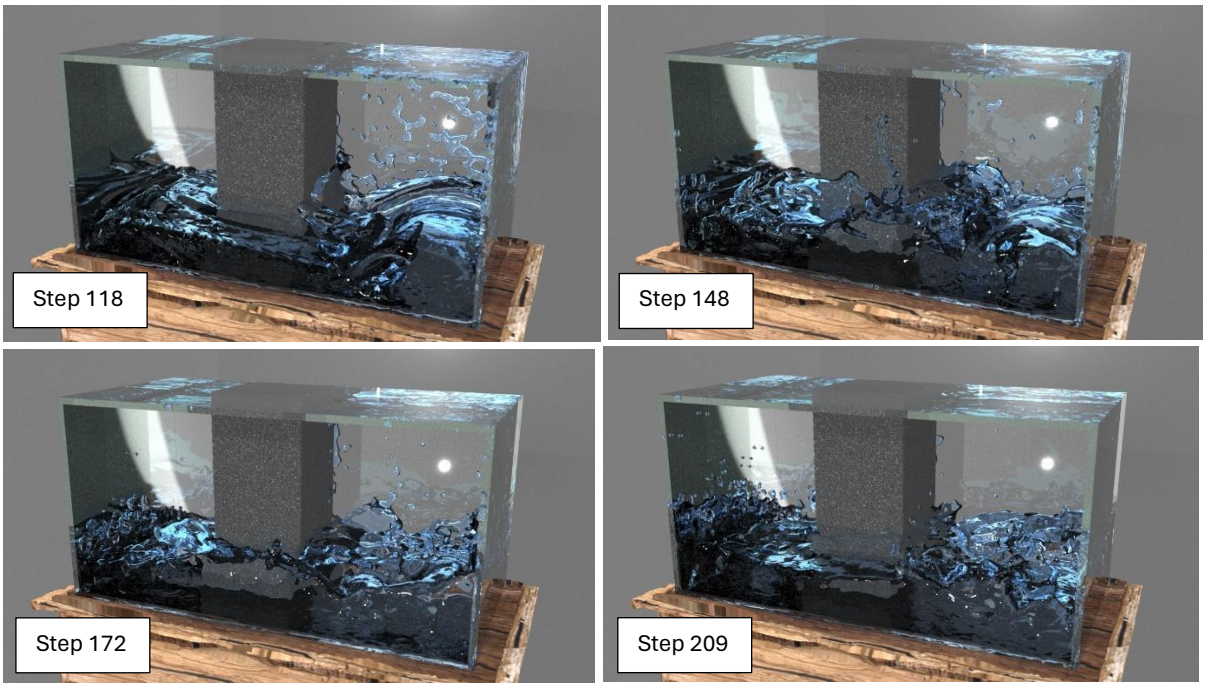


Figure 5.4.1 Final rendered results from step 1 – step 209.

VI. LIMITATIONS OF WORK

Creating MPM data simulations with Houdini can be done very realistically. All necessary features are available in Houdini, allowing for simulations that appear truly lifelike. However, in this first simulation project, the results may not yet appear entirely realistic or akin to real-world scenarios due to the need for further learning of more advanced methods to reach a higher level of proficiency. Mastering this software requires deeper learning and exploration.

A significant issue is also the use of inadequate computer specifications. Houdini is a 3D modeling software that demands a sophisticated CPU with multiple cores, substantial RAM, and a high-performance graphics card. The computer used in this project is an Alienware X2 laptop with an Intel i7 series 13620H CPU, 32GB RAM, and an Nvidia RTX 4060 DDR 5 8GB GPU. These specifications are still far below the standard needed to perform simulations in Houdini smoothly and quickly. Inappropriate computer use results in prolonged rendering processes, making it difficult for users to adjust details that enhance the realism of the simulation. For instance, checking changes in water flow or altering the color and shape of water splashes requires extensive wait times to see the results post-rendering.

The use of a desktop computer is highly recommended, as laptops typically do not yield optimal results. With a computer that meets Houdini's standards, the rendering process will be faster, and the overall software usage will be more efficient. Users will also not have to worry about adding features and components that can make the simulation more realistic because the computer can handle tasks quickly.

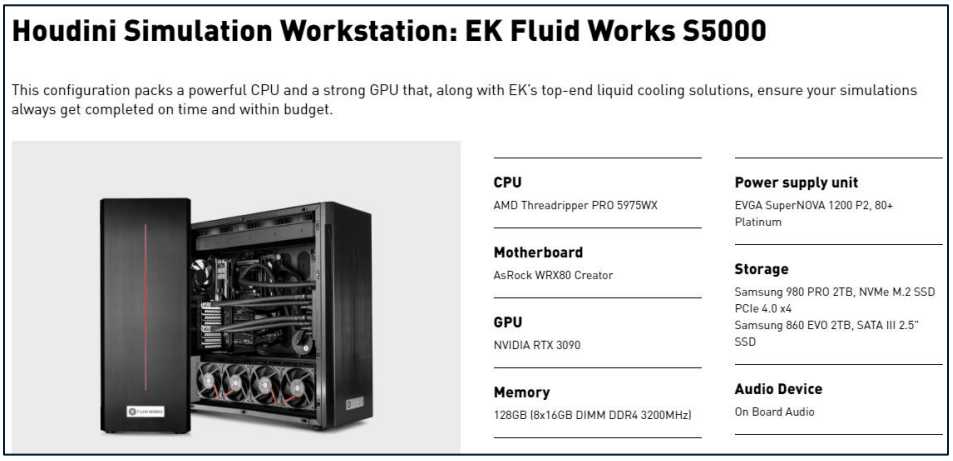


Figure 6.1 Recommended computer specs to run Houdini appropriately.

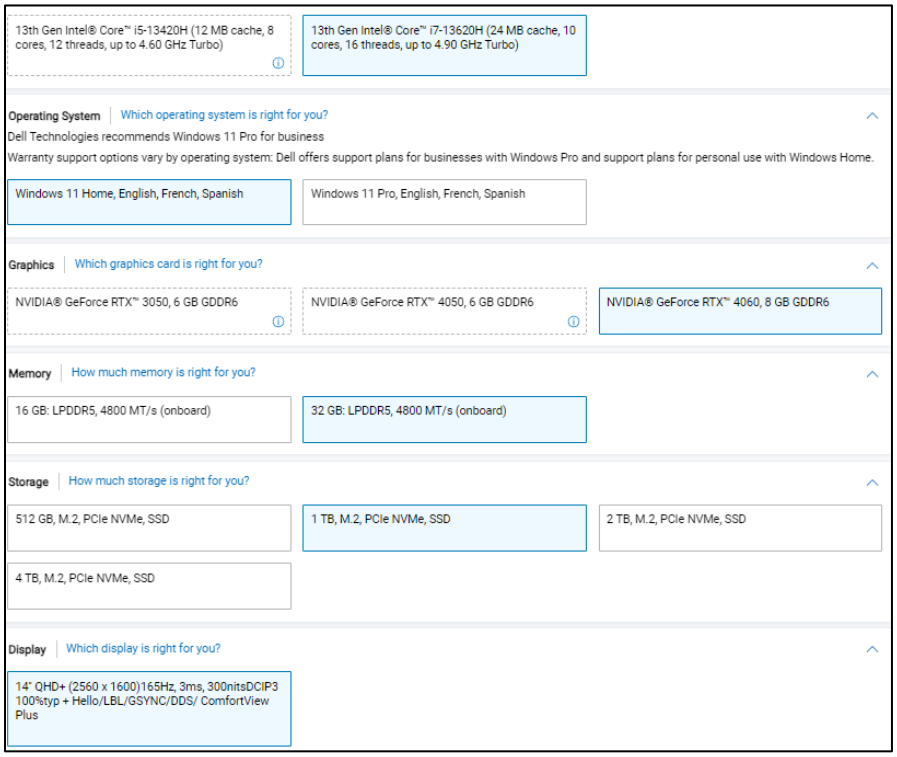


Figure 6.2 The current laptop specs used to run Houdini.

VII. CONCLUSION

This project has successfully demonstrated the potent combination of Houdini and the Material Point Method (MPM) for the visualization of complex geotechnical data, specifically in the context of water flow dynamics within dam structures. The 3D animations produced have not only clarified

the intricate behaviors of fluid-structure interactions but have also made these complex phenomena accessible to a broader audience, including those without a technical background.

Throughout this research, from initial data processing in ParaView to sophisticated rendering in Houdini, challenges related to data complexity and computational demands were addressed. Despite hardware limitations impacting the simulation's scope and detail, the results have underscored the critical role of advanced visualization tools in geotechnical engineering.

Moving forward, it is recommended that more powerful computing resources be utilized to handle the intensive demands of high-fidelity simulations and to explore Houdini's capabilities in other areas of geotechnical research. This approach will not only refine the visual outputs but also enhance the overall workflow efficiency. Future research should continue exploring Houdini's potential in various geotechnical applications, advancing our understanding of complex data through more intuitive and accessible visual representations.

This study affirms the importance of visualization in engineering and highlights the need for ongoing improvements in the tools and techniques used in the field. This will ensure that as our capabilities grow, so too does our ability to protect and optimize the infrastructure upon which we all rely.