# Pyro smoke simulation

The smoke simulation for the big shards begins the same way as the particle simulation for the small shards. Under the File node that loads in the geometry are the same two nodes and for the same purposes: Collision Source for collisions and Blast to isolate parts of the geometry used as sources. A pyro simulation requires more nodes to get started, so the initial setup was done by selecting Billowy Smoke from the Pyro FX shelf. The first node created is a Pyro Source which scatters points onto the incoming surfaces and creates attributes relevant to a smoke simulation on the points, namely density and temperature. The second node is Volume Rasterize Attributes and it turns the points from the previous node into volumes with the same name as the attributes on those points. The shelf tool also creates a DOP Network with many of the required nodes already created. A pyro simulation has a similar node structure to other simulations. The Smoke Object defines the voxel size and the initial size of the simulation container. The Gas Resize Fluid Dynamic node dynamically changes the size of the container and it's set to track the source geometry. The maximum bounds are also turned off so the container can grow as large as it needs to. A Volume Source node brings in the volumes created earlier, it is automatically set up by the shelf tool so nothing needs to be changed on it. The collisions however need to be added manually, so a second Volume Source is wired in with a Merge node. The Initialization is changed to Collision and pointed at the volume created by the Collision Source node. This is a decidedly different way to do collisions than the usual static object. The behavior of the Pyro simulation is handled on the Pyro Solver. First, the temperature diffusion, how fast the temperature differences in the gas even out, and cooling rate, how fast the gas cools down, were lowered. Buoyancy lift was also lowered, this controls how fast the gas rises. Only smoke is needed in this simulation, so the setting for enabling combustion could be turned off. The look of the smoke is controlled on the Shape tab. When working with the different controls, it helped to turn off every other setting entirely, going through them one at a time. Once the simulation had been set up, it was imported back to the geometry level with a DOP I/O node. The same node can also write the selected fields to disk once the correct DOP network and object are selected. 24 5.9 Building the final scene Once everything had been simulated and written to disk, a new scene file is created. Technically everything could have been done in the same file, but it was much cleaner to move to a new one and load just the files from disk. The original intent was to have all the sequences play one after another, time shifted into correct positions. This idea was discarded as it was unnecessarily complicated. Instead, a camera was created for each part and rendered separately. For the first shot, the camera tracks the tendril of liquid. To achieve this, the lines used for guiding the liquid in the simulation were imported but without the noise applied to them. A sphere is then created and made to follow one of the lines. The camera was then made to look at the sphere. Both of these, path follow and look at, use CHOPs or Channel Operators. CHOPs are a part of Houdini that was not studied much for this project. For the features mentioned earlier, the shelf tools were used. The position along the curve the sphere is at is a value from 0 to 1 on a Follow Path node and this value was keyframed to roughly match the speed at which the liquid advances. The Lookat node determines the object the camera is looking at and required no additional setup after it was created by the shelf tool. For subsequent shots the camera positions were either static or had their movements manually animated. During the formation, the crystal is only visible when it's covered by the liquid. To achieve this, volumes from the fluid simulation were used to find the parts of the crystal that were inside the liquid. This was done with an Attribute Wrangle, the function volumesample(), and a simple ifstatement. The value of the fluid volume at the sampled position was stored into a temporary variable. If the value was a negative number, that part of the crystal was inside the liquid. Two attributes are then written to the point: opac and emitint. Both are attributes that are recognized by the renderer, opacity with opac and emission intensity with emitint. Outside the liquid those values are 0, making that part of the crystal invisible. The shattering crystal switching between normal speed and slow motion is done with Switch and Retime nodes. As its name implies, the Switch node switches between the nodes wired into it. The switching can be done manually, by keyframing, or with expressions. In this case it was easiest with expressions. The Retime node is used to shift the cached animations to correct positions in time. The same combination of Switch and Retime nodes is used in three networks to sync three parts of the animation together. 25 Out of the simulation, particles simply vanish at the end of their life. They have the age and life attributes on them and those can be used to make their disappearance smoother. With an Attribute VOP a particle's age is divided by its life creating a value going from 0 to 1. The result of the division is wired into two Ramp Parameters, one for color and one for a float value. The output of the color ramp is connected to the Cd attribute, which is the default attribute name for color in Houdini. An extra parameter is created and multiplied with the output of the float ramp and exported to the attribute pscale. Pscale is another default Houdini attribute, this one for controlling point scale. With both ramps shaped, the particles now change color and size as they age, eventually shrinking to 0 size at the end of their life.

Rendering and lighting

The entire animation takes place inside a dome. The dome itself is lit by a single point light set to only illuminate the dome. Lighting for the rest of the animation is done with an environment light and an environment map from the website HDRIHaven [6]. The dome was excluded from casting shadows from the environment light. The renderer used by Houdini is called Mantra. For rendering the animation not many settings were changed. To keep render times low, render quality could also not be increased much. The main driver of quality in the render are the pixel samples. Pixel sampling of 3x3 means a minimum of 9 rays are sent into the scene per pixel. Higher values were tested, but they resulted in much higher render times, so the setting was kept at 3x3. Motion blur had to be turned off for the segments of the crystal shattering. Geometry motion blur relies on point velocities to calculate the blurring and since the shard topology, and subsequently the point numbers, changes each frame no accurate velocities can be calculated. The animation was rendered out as numbered sequences of PNG image files. No additional image planes were used as no post-processing was planned. After all the sequences were rendered, they were compiled into a video file using Blender. Blender is a completely free 3D modeling software which also includes a video editor. 26 6 Closing thoughts The original intent of this thesis was to learn the basics of Houdini. This goal was reached, but in doing so it showed how only the surface was scratched and just how much more there is to learn. The project did instill a desire to learn more and new ideas for future projects came up while working on this. But what of the project done for this thesis? The final product is quite simple, and different from the initially imagined version. This is, of course, to be expected. It was a learning process after all. Even the simplest things seemed insurmountable in the beginning, but towards the end there was a conscious effort to stop adding new things and concentrate on finishing. In hindsight it might have been better to create multiple self-contained effects rather than one where each part led to the next. This might have resulted in more varied effects and given more freedom in them. Whether Visual Effects will be the primary focus going forward remains to be seen, but it will be studied further. 3D character design is another field that will be studied alongside VFX. And while character design might be more weighted towards game development, VFX has a home in film and TV as well. This might hopefully widen the scope of potential work opportunities in the future.

**Creating a flipbook animation**

*Renders the contents of the viewer as images, allowing you to play them back at full speed.*

**On this page**

Overview

When you are working with complex scenes or simulations, the viewport can have trouble playing back at 24 FPS (or whatever your frame rate is). However, displaying the scene in the viewer is still faster than rendering the animation.

You can use a *flipbook* as a “middle ground” between playing back in the viewer and actually rendering the animation. It lets you view the OpenGL representation of the animation at full frame speed.

When you start a flipbook, Houdini plays back the animation in the viewer, but at each frame it captures the viewer content as an image. After Houdini captures each frame, you can view the images as an animation at full speed in mplay. Alternatively, you can save the images to disk to allow storing and sharing the flipbook.

How to

| **To...** | **Do this** |
|---|---|
| Create a new flipbook | 1. In the tools (on the left side of the viewer), right-click the Flipbook tool. 2. Choose “Flipbook with new settings”. 3. Use the controls in the dialog box (If you haven’t created a flipbook before in the current session, you |

| | |
|---|---|
| | can just click the icon to open the dialog box.) |
| Recreate the previous flipbook | In the tools (on the left side of the viewer), LMB click the Flipbook tool. |

## Creating Flipbook ( Cache Creating)And its  controls

**Output tab**

**Frame range/inc**
The start frame, end frame, and number of incremental frames to render. The default start and end frame are variable references($RFSTART and $RFEND), which grab the current values from the Houdini timeline. The menu at the right has a list of common frame ranges.

**Leave Playbar at Last Frame**
Instead of setting the playbar frame back to the frame it was at before the flipbook was started, leave it at the last frame the flipbook wrote.

**Flipbook to MPlay**
When enabled, all frames will be written to MPlay for interactive viewing.

**Output Files**
Allows saving of the frames to image files. Normally this is blank as frames are only written to MPlay ( **Flipbook to MPlay** defaults to on). If a filename is placed here, it will additionally write the frames to files as well. If __Flipbook to MPlay is off, frames will only be written to disk.

**Flipbook Session Label**
When **Flipbook to MPlay** is enabled, you can render to separate flipbooks at the same time by giving them different labels. If you use a label corresponding to an open mplay window, Houdini will send the new frames to that window.

**Visible objects**
Only objects whose names match this pattern will be visible in the flipbook. Use this to focus the flipbook on one or a few characters/objects.

**Render**

What part of the Scene View to render to the flipbook.

### Current Viewport

Render the currently selected viewport.

### Current Beauty Pass

Render the beauty pass of the currently selected viewport, which excludes the background, grid, handles, and decorations.

### All Viewports

Render all viewports into a single image, including any viewer annotations such as messages, **Draw Time**, and **Geometry Information**.

## Object Types

Which objects to include in the flipbook, by type.

### Currently Visible

All object types that are currently visible will be rendered.

### Geometry Objects

Only geometry objects will be rendered, which excludes bone, muscle, null, camera, light, and blend objects.

### Currently Visible except Geometry

All currently visible object types except for geometry will be rendered.

### All

All object types will be rendered.

## Append frames to current flipbook

Adds the frames on to the end of the last flipbook you created, instead of starting a new flipbook. This lets you put together a flipbook of different parts of a larger animation.

## Scoped channel key frames only

Only render at frames where there exists currently scoped keys.

## Enable block editing

When **Flipbook to MPlay** is enabled`, generate a flipbook with keyframe information to enable blocking in mplay.

Initialize Simulation OPs

Reset all simulation nodes in the scene before creating the flipbook.

**Flipbook tab**

**Use Audio Panel Settings**

Use the sound file and audio offset from the Audio Panel.

**Audio filename**

A sound file to play during the flipbook.

**Audio offset**

Time (in seconds) of the audio that corresponds to the first animation frame.

**Background images**

When **Output** is ip (that is, you are rendering the flipbook into mplay), an image to show behind the flipbook animation.

**Gamma**

When **Output** is ip (that is, you are rendering the flipbook into mplay), the gamma correction value to use in mPlay.

**LUT**

When **Output** is ip (that is, you are rendering the flipbook into mplay), a color look up table (LUT) file to use in mPlay.

**Limit Frame Time**

Progressive renderers in LOPs can take much longer to completely render than is needed for a flipbook. This optional time limit moves onto the next frame once it is hit, even if the progressive renderer is still rendering. This only applies to flipbooking LOP nodes.

**Limit Frame Percent**

Similar to **Limit Frame Time**, except the renderer progress is used as the limit. The renderer must report progress for this method to work. This only applies to flipbooking LOP nodes.

**Effects**

**Antialias**

Whether to do anti-aliasing of the viewport graphics, and how much. The default uses the current setting in Houdini.

**Motion Blur**

Render motion blur when creating the flipbook.

**Frames**

When **Motion blur** is on, the number of subframes to use when computing the motion blur. More frames produce better results, but take longer.

The menu determines when the subframes are sampled. For any frame F…

**Forward**

Blurs F to F+1.

**Previous**

Blurs F-1 to F.

**Centered**

Blurs F-1/2 to F+1/2.

**Shutter**

When **Motion blur** is on, how long the camera shutter is open. Longer shutter time produces more blur. The default ("from camera") uses the current camera's setting in Houdini.

For example, if the value is 0 there is no blur. If the value is 1 there is full frame blur.

**Note**

Depth of field is now performed by the viewport when looking through a camera.
See hou.GeometryViewportSettings.

**Size**

**Zoom**

Reduce the size of the images as they are saved (to disk or in memory). Use this to save disk/memory space.

**Resolution**

Specify a custom size for the flipbook images. When the checkbox is on, the flipbook uses the current viewer size.

**Sheet Size**
Individual frames can be collected into composite sheets when flipbooking. This parameter controls the arrangement of frames in a single sheet. The first value specifies the number of rows, while the second one controls the columns.

**Crop Out View Mask Overlay**
When this option is on, areas of the flipbook image outside the camera's aspect ratio are cropped.

When this option is off, areas outside the camera's aspect ratio are grayed out

# Fluid simulation (Emmiting Fluid)

There are two fluid simulations in the project, the results of the first leading into the second. The construction of both, and everything related to them, is detailed below. 5.3.1 Sources The source where the fluid particles are emitted from is a sphere shaped to resemble a water droplet with a Soft Transform node. Locationsforthe sources are brought in with an Object Merge from the Lines network, and the sphere is duplicated to each point with a Copy to Points node. A null node added to the end once again helps with pointing to the correct position from other networks. 5.3.2 Target To make the target for the fluid simulation to flow into, a sphere with the correct dimensions is needed. Instead of roughly estimating the correct size and location of the target sphere, its dimensions can be fetched from the crystal object itself using the bbox() function in the x, y and z 10 sizes. For the location the centroid() function is used to find the crystal's center. The functions can be entered into the parameters of the sphere in place of numbers. The resulting sphere is turned into an SDF, or a Signed Distance Field volume using a VDB from Polygons node. In an SDF each voxel stores the distance to the closest surface and whether it's on the inside or the outside of the surface. Positive values are on the outside, negative values are on the inside, at the surface the voxels have a value of 0. To get vectors from the volume a VDB Analysis node isset to calculate the gradient, or the change in the volume's values. The vectors can be given a custom name and they're named 'force' as it is a standard attribute name recognized by the Dynamics networks. Next, a Volume Wrangle and some VEX code shape the vectors. The code is commented in the picture below (figure 4). The vectors are shaped to spiral inwards to keep the fluid particles in motion and also attempt to avoid a cavity inside the fluid by forcing the particles towards the center.

To get started with simulations in Houdini it is often helpful to use the shelf tools to set up the basic nodes and networks. With the source geometry selected, it was turned into a particle fluid emitter. This creates a Dynamics network with the necessary nodes and adds a Fluid Source node to the source geometry. These nodes could be created manually but using the shelf tool creates the correct connections between them and links required parameters. 11 To keep everything clean, the created nodes are moved into their own sub-network. Even if nodes are moved to new networks all linked parameters are automatically updated. The lines that were created at the very beginning of the project are needed now, so they are imported with an Object Merge and connected to one of the available inputs on the Dynamics network. Simulations in Houdini follow a similar structure: they need a source, a container for the simulated objects and a solver. The shelf tool had already created the basic nodes for a fluid simulation, so only their settings needed to be tweaked. The first settings that need to be changed are on the FLIP Object and the solver itself. Particle Separation on the FLIP Object determines the "resolution" of the simulation by controlling how close particles can get to each other. The lower the value, the more particles and a more detailed simulation. The default base unit in Houdini is 1 meter, so a particle separation of 0.1 would mean 10 centimeters between each particle. This of course doesn't catch much of the finer details in a smaller scale simulation but might result in far too many particles in larger simulation. It's a trade-off between detail and simulation time, and storage space as simulations with tens or hundreds of millions of particles can require hundreds of gigabytes of hard disk space. For this project the particle separation was set to 0.035 for the final simulation. This gave a maximum particle count of around one million. But for initial look development the particle separation was increased to 0.2 so the simulation could run almost in real time and there were no lengthy wait times between changing other settings. Another very important setting is on the FLIP solver node under Volume Motion and Volume Limits. This determines the size of the simulated volume and is best minimized to cover only the space that is needed. To have the particles follow the lines that were created earlier and collect at the target location, several other nodes were needed. Because FLIP simulations deal with particles, nodes intended for regular particle simulations can be used with fluid simulations. With the POP Curve Force node, the lines are brought into the simulation. The node works best with single lines, so as there are four lines four nodes are needed. Later another four were added for additional control. All the nodes need to do the same thing, so all their important parameters 12 were linked so only one had to be edited. To achieve the desired look for the movement of the particles along the curves, values on the first two curve force node were animated with keyframes, the values would be copied to the rest of the nodes. These nodes are all merged together and wired into the Particle Velocity input of the solver. A picture of the simulation at this stage can be seen in figure 5. Figure 5 Fluid simulation with fluid sources in dark blue, curve forces in red, and the bounds of the simulation in light blue Two nodes are needed to get the target for the fluid working: A Field Force and a

SOP Vector Field. The vectors made earlier were named ‘force’, so they are automatically recognized by the node. The dimensions and divisions size of the Vector Field are linked to the dimensions and particle separation of the FLIP simulation. Apart from scaling the force up slightly, the Field Force node does not require any more set up. Gravity is not needed in this simulation so the automatically created Gravity Force is deleted. Once a desired look was achieved the simulation can be cached, but first the source node was configured to stop emitting new particles at frame 300 with ‘$F < 300’ in its activation text box. $F is a global variable in Houdini that refers to the current frame number

After the first simulation has been cached, its last frame can be used as the starting point for the explosion simulation. Because there are almost a million points all the unnecessary attributes are stripped from them leaving only velocity. For look development a temporary Attribute Wrangle is added that removes a certain percentage of the points to make the simulation faster. For the final simulation the node was disabled. Houdini 17 introduced a new fluid solver called POP Fluid that wires into a normal Particle Solver. Both the POP Fluid and the FLIP fluid were tried for this part, but a satisfactory look could not be achieved with the POP Fluid. Due to time constraints the more familiar FLIP fluid was used instead. This DOP Network was built from scratch. The geometry file from the previous simulation already had all the required fields, so they only needed to be piped into the solver. Also, because only the existing points will be used, no source is needed for the simulation. This simulation will also collide with objects, so they need to be imported. The crystal is imported with an Object Merge and wired into the DOP Network. Because the collisions are against static, or non-dynamic objects, a Static Solver is created. The crystal is brought in as a Static Object. A mistake was made when setting up collisions which was only discovered after the simulation had 14 been run and the final renders done. FLIP, like other particles, uses Rigid Body Dynamics for collisions, not Bullet. The default RBD collision was just good enough that the error wasn’t noticed. It was only discovered after the sequence had already been simulated and rendered. The fluid also has to collide with the ground, so a Ground Plane is created. The Ground Plane does not require any additional settings, it is just an infinitely large plane for objects to collide against. Both the Static Object and the Ground Plane are merged together and wired into the Static Solver. The Static Solver also has no settings to change. The FLIP Solver and Static Solver are also merged together and wired into a Drag Force which adds air resistance to the simulation. The final node is a Gravity Force. Most of the look development for this simulation happens on the FLIP Solver. On the FLIP Object only the particle separation is set to that of the previous simulation and a very small amount of viscosity added. To get the viscosity working it also has to be enabled on the solver. On the same page of settings, Slip on Collision is enabled with a value of 0.9. This means upon collision the fluid particle is allowed to slip along the colliding surface with 90% of its initial velocity instead of sticking to the surface or bouncing off. A Life attribute was added to the particles outside the simulation and to take advantage of it the Age Particles and Reap Particles settings are enabled so the particles will be removed after reaching the end of their lifespan. Once the desired look was achieved the simulation was once again cached to disk and surfaced just as it was with the previous fluid simulation.

Building an Ocean Simulation

In the current version of Houdini 17, there are an amazing amount of tools you can use to build an ocean body simulation. Currently as I speak, the Preview of Houdini 18 has just been released, and inside are some lovely updates and They've added a FLIP pressure projection tool. You can view the preview here: https://www.youtube.com/watch?v=hpDxWuBN3N4&t=42s

Now let's get into our current set of ocean/water simulation tools we can use.

Small Ocean: This tool is for creating an ocean surface, not an actual ocean simulation. It's good in the sense you can simulate an ocean surface relatively fast. But if less promising if you'd like to create something that dives in and out through an ocean body. This tool does not use DOPs.

Large Ocean: This tool is similar to the small ocean tool. It doesn't use DOPs and once again is great with creating an ocean surface. This tool is also great for creating large scale oceans that span long distances.

Guided Ocean Layer: Similar to the above tools, this tool creates an ocean surface. However, it also creates a layer of particles that are controlled by point velocities. This tool is primarily a FLIP simulation, and also creates an ocean volume to create realistic ocean water levels. Grabs velocity attributes from the ocean spectrum node. Great for shallow water simulations, or for adding boats to water.

Wave Tank: A FLIP particle fluid simulation that creates a tank to simulate waves. This entire tank will be filled with particles. Great for short timed simulations. However, it doesn't have great control of the waves it contains.

Beach Tank: Another FLIP particle tank. However, this tank slowly reduces the particles velocities over a period of time on the edges of the tank. This creates the appearance of the water washing up on the shore.

Flat Tank: Another FLIP tank. However, this tank can track objects inside of it and simulate FLIP particles around the object. Great for calm water. Or huge splashes from objects included in the simulation.

Tips for Ocean Simulations

- Turn of any whitewater nodes while you are working on your simulation. If they are turned on while you are working, your sim time will be incredibly slow, and your progress will be reduced.
- In the Beach Tank, the tool comes preloaded with default geometry for the beach. Feel free to replace this geometry if you need to. You can then tell the simulation to reference the new geometry.
- The higher you ramp up your velocity scale the more rough your waves will become, and vice versa.
- The lower you reduce your particle scale, the finer in detail the waves will become. Your render, cache, and sim times will also become longer.
- Don't be scared with long caching times. This is normal for water and tank simulations.

- If you are using a small or large ocean tool, try changing the shading view to Smooth Shading Mode. This will help you have a better view over your waves and their appearance.

- If you would like your waves to move twice as fast, try entering $T*2 into the Time parameter.

- If you are working with a Guided Ocean Layer. The fluid particles are controlled by the Layer size Parameter. You can find this in the Ocean Source Node.

- Decreasing the size of your particle separation will improve the appearance of your renders, and raise the amount of particles in your scene.

- To create more broken up foam on your ocean. Go to your Ocean Surface Shader and lower the frequency of the streaks.

Ocean Masks and Using Volumetrics

Let's start with volumes. Volumes are important to understand while working with Ocean simulations as they will help your simulation look it's best. Volumes can either be used to help create a mask, control the appearance of the depth of the water, the appearance of the waves, and the overall shading of your Ocean body.

Rendering your volumes in Mantra can be a bit more difficult as you have to tell mantra how you would like it to handle them. By adjusting or lowering the Volume Step Rate and Volume Shadow Step Rate, you can let mantra know that you don't need the volumes to be defined, but just to calculate them in the simulation.

Keep in mind you will have to bake out your ocean geometry to render your uniform volume shader correctly.

The Ocean spectrum Node also has an option to use custom volumes to create masks over your ocean layers. Houdini can use this mask to change the amplitude of your ocean simulation, and to create more detailed movement on the surface of your ocean. You can also you the volume mask to create very localized noise in your simulation.

Ocean Spectrum

Here we go!  The Houdini Spectrum Node is one of, if not the most important node you can use in your Ocean Simulations. This Node controls the volumes inside your ocean body, and the overall information for creating the waves. Let's dive inside, and try and figure it out.

To start, this node primarily relies on three attributes to operate. The Phase, Frequency, and The Amplitude. Any node placed after the ocean spectrum node will require these attributes to correctly evaluate the information for your ocean body. A node such as the Ocean Evaluate Node will refuse to operate at all without these values.

*Phase:* Controls the Offset of the waves.
*Amplitude:* Controls the maximum displacement and distance of your waves.
*Frequency:* Controls the angle of your waves. As well as the time it takes for one wave to complete of cycle.

It is also recommended while working with the Ocean Spectrum Node to increase the Resolution Exponent. This is because in the default settings of this parameter, artifacts will be visible in your render. Increasing this parameter to something like 11 or 12 will remove these.

Other Tips About Ocean Spectrum

- If you are aiming for slow waves consider lowering your Timescale parameter to a lower value.

- You can use the Filter Above Resolution parameter to clear away any wave that have higher or lower frequencies than you would like.

- Play with the Grid Center if you'd like to adjust where you waves are coming from, or if you would like control over a particular wave in your simulation.

- Under the wind tab, you can use the Directional Bias to fix any waves that refuse to align with the direction you would like them to move in. They will be dampened out the more this parameter is increased.

- Beware of the Chop parameter. If this is set too high the waves will become inverted on themselves and start clipping.

- Using the Normalize Parameter under Wave Amplitude will allow you to change some more features of your waves. This function maintains the height of your waves. And at the same time makes sure only the speed and the scale of the Amplitude are affecting the waves. Turning this off will make the waves affected by other parameters as well.

- Always use your visualization tools! Visualizing the spectrum can help in so many ways regarding how you troubleshoot your simulation.

Lighting an Ocean Body

For lighting an ocean, always keep in mind the size of your ocean body, the material settings for the surface and volume of the ocean, lighting samples, the direction of your light sources, and the reflections of any other objects in the scene. The appearance of your sim will be affected accordingly.

Sometimes while lighting an ocean , you might have to change your lighting to something a bit different from your regular scenes. To start, always add an environment light. Your ocean needs reflections in it's environment to help create it's ocean colors. Try and match your HDRI as best as you can to your chosen scene. Both shadow, color, time of day, and environment wise. So far so good.

Here lighting can be a bit more fun. I have a friend who uses a secondary environment light in his scene so he can create a light striking the ocean volume, and the other for the ocean surface. He uses the same HDRI map for both. This seems to work for murkier water. Another way to approach the lighting is to create your default environment, directional, and ambient light and work from there. I'd recommend this route if you don't have any lighting reference. If you have a lighting reference, try and move your lights accordingly to match the shadows and direction of the light source.

If you need proxy geometry in your scene, go right ahead and add them. You can channel them out separately in your render passes later. But for example, let's say you have a beach scene. There are docks,boats, and buildings intersecting with the water. You'll still need geometry for those objects in your scene to create the reflections. This will further help your scene look cohesive.

# Fracturing(Breaking Geometry), and collision

## What is RBD, and How Do I Use It?

RBD, more commonly known as Rigid Body Dynamics, is a system inside Houdini.

This system allows for the motion, fracturing, and collision of objects to be handled within solvers. As well as a combination with SOP tools. It handles objects as if they were hard objects, such as buildings, bricks, or other structures.

RBD can handle objects in two states. Active or Passive. Active objects can interact with collisions and forces. While passive objects can collide with active objects, they don't move.

RBD also has several different material fracturing options. This allows you to generate accurate fractures for different types of objects. These fracturing methods can be found in the RBD Material Fracture SOP, and in the Voronoi Fracture SOP. The RBD Material Fracture SOP allows for you to create wood, concrete and glass fractures. Then it will pre fractures the geometry. It is an ideal node when working with multiple constraints, glues, and meshes.

The Voronoi Fracture SOP is great for small scale RBD sims. As well as generating fractures in concrete. The node is also on it's 2.0 version(As of 2021) so there have been several improvements to it.

There are also other tools that go along with the Voronoi fracture SOP to improve its use. As well as your RBD simulation as a whole. It has two inputs. The first is for the mesh you would like to fracture, and the second input is for the points it needs to fracture the mesh. You can create these points with scatter SOPS, or points from volume SOPS. Or even the Add SOP.

In combination with the node above, you can also use the Voronoi Fracture Points SOP to generate points for it to use. This node calculates data from the incoming mesh and a set of points to generate impact points for the fractures.

Internally, the node generates volumes from the chosen impact points from metaballs that are copied to each point. However, you can customize this metaball generation by adding metaballs into its second input. These metaballs will be merged into the metaballs the node generates, and then together they will form the impact volume. The volume is then used to divide the geometry into three regions.

These regions are: The Surface Region, The Interior Region, and the Exterior Region. The Surface Region is the area where the object intersects with the impact volume. The Interior Region is the intersection of the volume of the object, and the impact volume. Finally, The Exterior Region is the area of the volume that is outside the area of impact.

You can also use the Voronoi Spilt SOP to selectively cut up your geometry. It does this by separating and breaking the geometry by a set of polylines. Each line cuts along its midpoint. In order to create these lines, to get the node to work, you can use the Tetrahedralize SOP.

The RBD Interior Detail SOP, is also one of these nodes. It can be used to create high resolution pieces on your mesh on both their inside and outside.

The RBD Cluster SOP can be used to create large clusters of fractured pieces.

Probably the best preview tool in your RBD arsenal is the RBD Exploded View SOP. This tool will explode and separate your fractured pieces so you can see how they will fracture. It's also great for matching high res and low res geometry together to see how the different resolutions can have an impact on the simulation.

This node also has different visualization options. A few of them are:

Geometry Outside Proxy: Colors areas red to show where your geometry is poking out beyond your proxy geometry.

Volume: Shows the volume distribution amongst the pieces.

Pieces: Displays the fracture pieces in different colors.

Strength and Stiffness Constraints: This let's you visualize the strength values, glue constraints, and spring constraints if they are applied to your model.

## *How Do I Start To Build a Project Using RBD?*

There are several different ways to set up a RBD simulation. But let’s try and simplify it from start to finish.

#1 Prepare your objects. Fracture your models.Create any groups you might need to control the fractures and constraints.

#2 You can also do this before step one, depending on your model.You’ll need to clean your objects. So basically, make sure the geometry is closed, check for holes, wrong normals, and any unused points.

#3 Check your UVs. After simulating, your model’s UVs might be incorrect. This can be caused by incorrect UVs at the start of your simulation.

#4 Fracture your model.Choose your preferred way.

#5 Create the UVs for the inside of your now fractured model. If you don;t do this, and the inside pieces of your debris faces the camera, the materials will appear warped and inaccurate.

#6 Cache the object.

#7 Create your constraints if needed.

#8 Slap down a RBD configure node.

#9 RBD Solver. It’s time to simulate, baby!

#10 Drop down a RBD I/O. This is to cache the simulation.

#11 Do whatever you need to do next.

When setting up these scenes it’s important to consider these things.

Aesthetics: How realistic does your fracturing and overall simulation look?

Feedback: Can you change this setup without altering the end result too much if someone asks you to do so?

Further Flexibility: How does this setup work with certain geometry resolutions, or different geometry shapes? How functional is the setup? Can you easily alter the simulation without breaking it?

Performance: How long does the scene take to cache out? How long does it take to simulate? Can you improve cache, render, and simulation times?

Extensibility and Procce-durablity: Can you hand this build off to other users and have them easily understand it? Will someone have to break or add things to your build in order to use it?

Node by node this is pretty much what you should be doing:

#1 Get your Geometry
#2 Add a Convex Decomposition if you need it.
#3 RBD Material Fracture: Choose your material type.
#4 RBD Configure
#5 RBD Pack
#6 Add Velocity (If needed). This can be done with a wrangle.
#7 RBD Unpack (Turn on transfer attributes and add the attributes you want)
#8 And RBD or Bullet Solver
#9 You can now drop down an RBD Exploded view. You can also drop down this node to preview your simulation and geometry at any stage of building your simulation.

Also keep in mind that when trying to fracture different objects you will also need to consider this different ideas, as well as research them:

How can you fracture the object in the most efficient method?
Is a particle system enough to create the debris you need, or do you need to use a full RBD system? Sometimes you can fake things, but your supervisor might know better. ;)
Will you have to create any secondary simulations for this project to make it look realistic?

### *Workflow For Glue Constraints*

Constraints can be tricky. And even trickier to setup. So here is a breakdown on Glue Constraints and how you can approach them.

Long story short. Houdini allows you to glue RBD pieces together. These objects will stay together until they are told to break apart, or the strength of the “glue” is overcome.

There are two main workflows for using glue.

First one:

- Glue an object with no fracture groups or customized glue attributes together.

Second One:

- Get your object.

- Create fracture groups

- Import these chunks as pieces glued together.

Glue can be a great way to get your model to break apart in chunks and have a delay in its destruction. Or make the object fall piece by piece.

The way RBD is progressing in Houdini 19, to quote Mr CGwiki: RBD is copying vellum. It is becoming more SOP based, and very much so in the way it handles it’s constraint networks. Also, keep in mind that sometimes the SOP RBD tools work better without packing the geometry.

One easy way to set up your constraints before they enter your solver, is to use a RBD constraint property SOP.This node can help you set up your attribute constraints for bullet simulations.

### *Can I Use RBD To Influence Other Types of Simulations?*

Absolutely! In order to make your final simulation look realistic, you’ll need to combine it with particles, dust, more pyro, and other simulation types.

But you might want to consider making the moving pieces of your RBD simulation be the emitters for your particle and pyro simulations. This will allow the other simulations to inherit the velocity from the pieces and move with the debris accurately. It can also make the individual pieces look smokey or hot because they are emitting steam.

When using fully simulated RBD debris as an emitter , keep in mind you might not need all of it. Delete the pieces and chunks you don't need, and then work with the simulations from there.

You might also have to advect RBD pieces with other simulation types as well. Mainly with the velocity fields generated from them. Let's use this example. Let's say you want to simulate a tornado going through a building, and the pieces of the building being sucked up into the tornado. The tornado itself will probably be a pyro simulation. This means you will have a velocity attribute generated from this simulation. Blast away all the other attributes, and then plug that vel attribute into the solver of your scattered debris pieces. Then dive inside, and drop down a node that will allow you to advect the debris up into the air. This can be done with a POP advect by volumes.

# Explosions Or Fire Making

When building an explosion it's probably a good idea to understand what type explosion you are attempting to build, and how it is being created. Therefore, understanding the different types that you can build, and the creation process behind them is important. There are 6 main types of explosions.

Natural**:** Volcanoes, Bush Fires, Dirt impacts, or any explosion resulting in an Earthy phenomenon we can place here. These are effects caused by an interaction with the biosphere of the earth.

Astronomical**:** Anything that explodes that is not on this Earth. Supernovas, Planetary explosions, or really anything that is out of this world. These are usually caused by sci-fi event,chemical,and universal causes.

Chemical: This category covers most made-made or artificial explosions. Examples: rockets, guns, gunpowder, fuel tanks, cars, etc.

Electrical and Magnetic**:** Any explosion caused with magnets or electricity. For example if you had an explosion at a Hydro power Plant, you'd place it under this category.

Mechanical and Vaporous**:** An explosion that is rupturing out of a container will will under this category. Or any explosion that involves pressurization, liquids, or chemicals. Usually workplace environments feature these types of explosions.

Nuclear**:** Nuclear explosions, Nuclear missiles, Uranium, etc. These are the largest man made explosions possible.

There are also some main properties of explosions you should familiarize yourself with as well.

Force**:** The overall explosive force emitted from the source.

Velocity**:** The speed of the reaction. Usually the Velocity of an explosion is very high.

Heat Distribution and Characteristics: The dispersion of heat across the explosion. This can affect how the simulation dissipates, the color, and the expansion of the explosion itself.

Fragmentation: The pieces , particles, or other material an explosion gives off when it explodes.

Initialization: The process on how the explosion start. Whether that be by impact, shock, chemicals, or other environmental factors.

How to make a Basic Explosion

One way to always start with an explosion is using a shelf tool. There are two that seem to work pretty well. One being Fireball. and the other Explosion. Basically, start by selecting your chosen object to turn into an explosion, and hit that chosen shelf tool. Then your smoke import SOP and DOP should appear, and you should be good to go.

You can also build your DOP and Import inside your Object level SOP. The choice is up to you, but if you are new to the game, stick to the shelf tools. Then have fun either way with the force options on your solver, and source objects.

Rendering an Explosion

When rendering anything Pyro, you may run into some issues where the explosion may look poor in quality, the color may look wrong, render times are through the roof, etc. Here's some basic tips on how to avoid these problems. Some of these answers I found while lurking on the forums, so I would also recommend looking there.

Color and Lighting

In this section we are going to focus on materials and lights in your scene. Let's start with shaders. Based on the type of explosion or Pyro effect you've created, Houdini might have automatically added a shader on your sim for you. If not, you can select any of the materials in the Material Palette's Volume tab. If you are creating a shelf sim, then more than likely you'll be using a fireball or flames shader.

If you are using the fireball or flames material you'll notice that the fire has a color mode drop down menu. If I'm rendering realistic fire and smoke, I usually switch this over to Temperature to Physical Black Body Color. It gives you a lot more realistic ways to adjust your explosion's appearance. I also switch over to the Smoke Field and Fire Intensity Field tabs and turn on the use look up ramps. This also gives you more control over your explosion's fire and smoke output.

Lighting can be a bit more tricky. I always add an HDRI/Environment Light to all my scenes to create more realistic lighting, and match the lighting to my background plate more correctly. A Directional light is great for sunlight if your explosion is outside, or needs to be the center of illumination. If you need softened shadows in your creation add an ambient light. You can also change the tone of your shadows by changing the color of this light source. For sharper shadows, remove excess ambient light.

Rendering Engine and Object Selection

With rendering nowadays, I usually use PBR in Mantra for rendering everything. It works great, and your quality of work jumps up. Another thing that I'll mention in the Mantra node, is the Objects menu. Sometimes, while making VFX, you'll have to produce the matte for your fx. This is the menu to look at for excluding objects, forcing mattes, or creating solo lights. Take a look in your free time. :)