A great way to understand Houdini’s node-based workflow is to explore it in the context of a project. It is important to start learning how to think and work procedurally. In this lesson, you will learn how to create your own custom brickify tool using procedural nodes and networks to define its function and interface.

Along the way you will get to use different aspects of Houdini’s workspace. Be sure to refer to the overviews in the introduction to remind yourself of how these UI elements work together. The lessons will then give you a chance to put your ideas into practice, which is one of the best ways to learn.

LESSON GOAL
- To create a custom tool that turns any given 3D shape into toy bricks.

WHAT YOU WILL LEARN
- How to model a plastic interlocking brick
- How to break down a default rubber toy shape into a grid of points.
- How to use packed primitives and instancing to speed up interaction.
- How to use attributes to color the bricks using a texture map.
- How to work with nodes and networks to control the flow of data
- How to create a Digital Asset to package up and share your solution with others.
- How to animate the bricks appearing over time.

LESSON COMPATIBILITY
Written for the features in Houdini 17.5.293+

The steps in this lesson can be completed using the following Houdini Products:

- Houdini Core ✔
- Houdini FX ✔
- Houdini Indie ✔
- Houdini Apprentice ✔
- Houdini Education ✔
PART ONE:
Create a Single Brick

To get started, you will build a single brick model that you will later copy onto points to create the brickified shape. You will create this shape using a combination of polygon modelling tools. Along the way you will see how each action you take creates a node in Houdini that creates a recipe of the steps taken to create the geometry.

01 Select File > New Project. Change the Project Name to brickify_lesson and press Accept. This creates a project directory with subfolders for all the files associated with this shot. Select File > Save As... You should be looking into the new brickify_lesson directory. Set the file name to bricks_01.hip and click Accept to save.

02 In the viewport, press c to bring up a radial menu. From this menu, choose Create > Geometry > Box. Your cursor now shows the outline of a box waiting to be placed in the scene. Press Enter to place it at the origin. In the Operation Controls bar, set Size to 0.2, 0.2, 0.2 and Axis Divisions to 3, 2, 3. You can see that there is a box object in the Network view. This object level node contains the transformation information for this shape. The Operation Controls show you parameters from another box node one level down.

03 Click on the Select tool then Press 4 to go to primitive selection mode. Select the top four faces on the box. Press c to bring up a radial menu and choose Model > Polygons > PolyExtrude. In the Network view you can see the box node feeding into a polyextrude node. In the Parameter pane, use the slider to set Inset to 0.04 which creates new polygons on the top surface of the box. Each node contains parameters relevant to the purpose of that node. These are geometry nodes that are otherwise known as surface operators or SOPs.

04 Next, press the T key to call up the Move tool. This adds an edit SOP node into the network. In the viewport RMB-click in empty space to bring up a menu and select Make Circle to round out the selected polygons. This menu is associated with the edit node. Every node has its own interface that you can access as long as a relevant tool is active. In this case, the Move tool gives you access to the handle. In other cases the Handle tool would be used to access the interactive handles for a node.
05 Press c to bring up a radial menu and choose Model > Polygons > PolyExtrude. In the Scene View pane, use the handle to drag up the polygons and set Distance to 0.05. Tumble around and press s to go into select mode then select the bottom four polygons of the box. Press q to repeat the PolyExtrude tool. In the Parameter pane, use the slider to set Inset to 0.025. Press q to repeat that tool and set a Distance of -0.175. When you finish, tumble back to see the top of the brick.

07 Press 3 to go to edge selection and press n to select all the Edges. In the Scene view press tab and start typing Group.... Select Group and in the Parameter pane, set the Group Name to bevel_edges. Next, set Enable to OFF under Base Group and then set Enable to ON in the Include by Edges section. Turn on Min Edge Angle and set it to 89 and then turn on Max Edge Angle and set it to 91.

08 Press s to go to the Select tool. Press 9 to turn on the "Select Groups" option. In the popup window, click on the bevel_edges group.

In the viewport, press c to bring up a radial menu. From this menu, choose Model > Polygons > PolyBevel. This adds a polybevel node and automatically fills in the Group field with bevel_edges.

Now set the bevel Offset to 0.006. Under Fillet, set Shape to Round and Divisions to 3.

09 Go to the object level and in the Network view, rename the object to single_brick. With the brick selected, press Shift + to turn on subdivision surface display for this shape. Deselect the brick to see the model subdivided. If you see wire lines on your object, press v and from the radial menu, choose Shading > Smooth Shading to hide them.

You are now going to create some points to copy the bricks to. Mouse over the node in the Network view and choose the Display flag to hide the brick for now.

Save your work.

**SUBDIVISION DISPLAY**

You can use Shift + and Shift - to turn subdivision display on and off on selected polygonal objects. This creates a viewport subdividing that lets you see what the shape would look like subdivided. These hotkeys set the Display As parameter which can be found at the object level on the object’s Render tab.

Your object will not render with subdivisions unless you turn on Render Polygons As Subdivision Surfaces on the same tab.
PART TWO: Copy Bricks to a Point Cloud

You are now going to create a cloud of points that match the shape of a particular piece of geometry. You are then going to instance the bricks to the 3D grid to create a brickified version. The instancing is generated by packing the brick geometry then instancing them to the points.

01 In the Scene View, press tab and start typing Test Geometry... then choose Test Geometry: Rubber Toy. Press Enter to place it at the origin.

Press spacebar b to bring up four views. Use spacebar h to home the orthographic views so you can see the rubber toy in all of them.

Press i to dive into the test geometry object. Set the following:
- Uniform Scale to 3
- Translate Y to 1.5

This raises the toy up so that it sits higher on the ground. Use spacebar-h to again home the views.

02 In the Network editor, RMB-click on the output of the rubber toy node and type Points from Volume... Select the Points from Volumes and place its node in the network. Now set its display flag to focus on the output of this new node.

Mouse over the view and press spacebar b to expand this view.

Press s to go to the select tool and press 2 to get point selection and then press n to select all the points which will highlight in yellow. You will copy the bricks to these points.

03 Press u to go back to the Object level and shorten the name of the rubber toy to rubber toy. In the Network view, display the single_brick object then press the shift key and click on the rubber toy and then the single_brick.

From the Modify tab, select Combine to bring these objects together. You are taken to the geometry level and the nodes are feeding into a merge node.

04 In the Network view, select the display_merge node and press the delete key. Now the polybevel node chain is displayed and the other chain is not. Houdini lets you choose which node you want to display which would be the node that will be visible when you go back to the object level.
Part Two: Copy Bricks to a Point Cloud

05 RMB-click on the output of the *polybevel* nod, start typing *copy to points...* and select *Copy to Points* node. Click to place the node below the two chains and set its *Display Flag*. Turn on the *Pack and Instance* option. This is important because it will display the copied bricks much faster than if this option is off. At this point there is an error on the node because we haven’t connected the second input.

06 Click on the dot under the *pointsfromvolume* node then connect it to the second input on the *copytopoints* node. The bricks appear to be overlapping. Go back to the *pointsfromvolume* node and set *Point Separation* to 0.2. Now you are copying bricks to the points of the grid.

07 Click on the *copytopoints* node. Some of the bricks may appear dark grey and don’t display the proper brick. This is because of geometry culling in the viewport. You can fix this with a display setting. In the Scene View, press *spacebar-d* to bring up the *Display Options*. Click on the *Optimize* tab and either set *Scene Polygon Limit* to 50 million or set *Distance-based Packed Geometry Culling* to OFF. Now you can see all of the bricks being copied to the points as Packed Instances.

08 Before saving your work, let’s Organize the network. Select the nodes that make up the single brick and press *Shift-O* to create a network box around them. Click on the title bar of the box and enter *single brick*. You can then collapse the box and move it down to de clutter the network a bit. Save your work so far.

Packed Instances

If you leave *Pack and Instance* setting OFF on the copy to points node then you will end up with a large model with over a million points and primitives. This would make it very slow to manipulate in the Viewport because instancing is not being used.

If you turn it ON then the 338 points on the brick model are packed up and instanced which leaves a much more efficient point count for the copy to points node.
PART THREE: Add Color

You are now going to add color to the points which will then be picked up by the instanced bricks. At first this transfer of color only applies in the viewport but by setting up a proper material which has a Use Packed Color option, you can set up the point colors for rendering the bricks.

01 Add a color node between the pointsfromvolume node and the copytopoints node. This will add color to the points which will get copied to the bricks. Change the color to red to help the bricks stand out against the background.

Click on the Render Region tool and drag a bounding box over the rubber toy. This kicks off a test Render and you can see that the bricks are not rendering as red. You need a material assigned that picks up the color.

02 Press tab in the Network view and type Material Network... Choose the Material Network tool and click to place the node down in the network.

Go to the Material Palette and close /mat and open up /matnet. Drag a principled shader down into the matnet. Name it brick_material.

03 Go back to the geometry network using the back button in the Network view. RMB-click on the output of the copytopoints node and type Material... Select material and place it’s node in the network and set its display flag.

Click on the Operator Chooser button on the far right of the Material parameter. From the pop-up window, navigate to and highlight brick-material. Turn on the Export Relative Path Option and click Accept.

The material is assigned but the bricks still don’t render.

04 Go back to the Material Palette and select the brick_material. Click on the Surface tab then set the Base Color to 0.5, 0.5, 0.5 and turn ON the Use Packed Color option. The rendered bricks should now be red.

If not then click Render in the top bar of the Scene view just to make sure the changes have been applied. Now the bricks are rendering red to match the colors assigned to the points.

Save your work so far. Click the x button in the top right of the render region to close it.
PART FOUR: Switch over to a Teapot

You are now going to add a switch node that switches between the Rubber Toy and a Teapot. This will show you how the network of nodes can handle different shapes and still give us the desired result. This shows that this network’s procedural “recipe” is doing its job.

01 Press s to go to the select tool. In the Network view, press tab and type the word Switch then from the Sourcing folder drag it into the Network view. Next, drag it over the line connecting the rubbertoy node to the pointsfromvolume node. This inserts it into the network between the two nodes. This node will make it easier to switch between different incoming shapes.

02 In the tool shelf, go to the Create menu and drag the Platonic tool down to the network view. This places a platonic node at the geometry level. Tools can be dragged into the network view as long as the node type makes sense for your current network level.

Set the platonic node’s Solid Type to Utah Teapot. Set Radius to 4 and Position Y to 1.9. The volume from points node updates the points to match the volume and we have a new configuration of bricks.

03 Connect the output of the platonic node to the input of the switch node. Now you can select the switch node and change its Select Input to 1. The platonic shape has been cubified using the same setup as the rubber toy.

This is one of the big benefits of a procedural system. Later you will package up this network into a custom tool called a digital asset. As a digital asset, it will be easier to share the network with others and to manage the tool as it gets deployed in a studio environment where changes and updates are inevitable as the tool evolves.

04 Use the switch node and set the Select Input to 0 to go back to the rubbertoy. You can now go back and forth between these two shapes and even add more shapes to see them brickified.

Save your work.
PART FIVE:  
Color the Points using a Texture

Earlier you added a color attribute to the points which affects the coloring of the bricks instances. Instead of using a single color, you are now going to use a texture map to create a more interesting look for the bricks. This will involve some special nodes to turn the texture into point colors.

01 Turn on the Display Flag on the testgeometry_rubbertoy. Select the rubbertoy node and in the Parameter pane turn Off the Add Shader parameter.

You will bring back the color on the bricks using a different method that involves adding color to the bricks by pulling the color from a Texture Map on disk.

02 From the Asset menu, choose Edit Asset Properties > Rubber Toy. In the Properties window, click on the Extra Files tab and select toylowres.jpg.

Click the Save as File button and save it into the tex folder. The texture was stored in the digital asset so that it could be shared along with the asset. You will use the texture on disk to add color to the bricks.

03 In the Network view, press tab and start typing Attribute... Select the Attribute VOP node and place it into the network beside the pointsfromvolume node.

Feed the output of the switch node into the first input of the attributevop node. Set the Display Flag on this new node. In the Parameter pane, set Run Over to vertices.

04 Double-click on the attributevop node to dive down and use the tab key to add a Texture VOP. Feed it into the Cd input of the geometryvopoutput node.

Add a UV coordinate node and feed it into the UV input on the texture node.
When you want to get attributes from one piece of geometry to another, you can use Attribute Transfer which uses a Distance Threshold along with other parameters to get the attributes copied over.

In the case of the rubber toy, you are transferring the Cd attribute from the points on the geometry to the point cloud that is used to copy the bricks. You could also copy attributes such as UVs or Capture Weights.

05 Select the Texture node. Click on the Gear icon on the far right of the Texture Map parameter and from the menu, select Promote Parameter. This will add this parameter to the upper level of this node.

Click on the little knob that appears next to map. In the parameter pane, change the Label to Texture Map.

06 Press u to go up one level. You can see the Texture Map parameter. Leave it set to the default Mandril.pic texture for now. You will add the toyowres texture map at the end.

Add an Attribute Promote to the end of this chain. Set Original Name to Cd and Original Class to vertex. Leave New class set to Point.

07 Add an Attribute Transfer node. Wire the pointsfromvolume into the first input and attribpromote into the second. In the Attributes tab click on arrow on the right side of the Points field and choose Cd.

Add a switch node after the color node and wire the attribtransfer node into the switch. Rename this node texture_switch.

Set the Switch to 1. Set the Display Flag on the copytopoints node.

Now you can see the colors from the texture map being transferred to the copied points and therefore to the copied bricks.

08 Select the attribvop node and click on the file button on the far right of the Texture Map parameter and navigate to the toyowres.jpg texture and click Accept. You can see that the texture map colors are now being assigned to the vertices.

Save your work.

PART FIVE: COLOR THE POINTS USING A TEXTURE
PART SIX: Create a Brickify Digital Asset

Now that the brickify recipe is working and the nodes are wired together properly, you are going to wrap up some of the nodes to create a single Houdini Digital Asset [HDA] node. Now you can share the network with parameters from inside the asset promoted to the top level to create an interface that can generate unique results each time the asset is used.

01 In the Network view, drag the platonic node off to the side. Select all the other nodes in the network and from the Assets menu, select New Digital Asset From Selection.... This will collapse them into a single node.

Set the Operator Name to brickify which in turn changes the Operator Label to Brickify. Click on the button on the far right of Save to Library. Select $HIP in the Location sidebar and then double click on the HDA directory. Press Accept then click Accept again in the Create New Digital Asset dialog.

02 This brings up the Type Properties window and make sure the Basic tab is visible. Set Minimum Input to 0 so that you don’t “require” an input for the asset to work. The Maximum Inputs parameter is set to 1 which defines how many inputs nodes you will allow.

This is the input that is currently connected to the platonic node. When you use this digital asset later, you will use this input to point it to a different shape.

Press Apply. DON’T press Accept because it will close the window.

03 In the Network view, rename the new asset node brickify_asset and double-click to dive into it. Click on the switch node that is switching between the testgeometry_rubbertoy and the Subnetwork Input node. The input node is bringing in the platonic teapot shape from the level above.

In the Type Properties window, click on the Parameters tab. Click on the Select Input parameter name and LMB-drag it to the Existing Parameters list in the Type Properties window. Drop it on the root to add it to the UI. Click Apply which adds the parameter but doesn’t close the Type Properties window.

WHAT IS AN .HDA FILE?

When you saved your asset, it creates a .hda file on disk. HDA stands for Houdini Digital Asset and the asset definition is stored in this file then referenced into your scene. This file contains information about the nodes, the promoted parameters, UI elements and more. This file can be referenced by multiple people into different shots for a shared experience.

The assets being referenced into your scene can be managed by going to the Assets menu, selecting Asset Manager... and opening Current HIP File.
RMB-click on brickify_asset in the Network View’s path bar and choose Parameter and Channels > Parameters. Now you have a floating Parameter window with the two brickify asset parameters. These are the parameters that will be available to anyone who uses this asset. Let’s add some more.

The brickify node has a new parameter. Change its value from 0 to 1 and back to see how it affects your scene. The problem is the name isn’t very appropriate and a menu would work much better than a slider in this case, so you are going to refine the UI using the Type Properties.

In the parameter list, click on the Select Input parameter. To the left are options for refining how people see it. Change its Name to shape and its Label to Shape.

Now click on the Menu tab and turn on Use Menu. Now under menu items, type 0 under Token and Rubber Toy under Label then press enter. Next type 1 under Token and Custom Shape under Label. Press Apply.

Now in the floating parameter pane, you see a Shape parameter with a menu. Try it out to see how it works.

Select the second switch node that sits just under the color and attributetransfer nodes and promote the Select Input parameter to the parameter list. Change its Name to look and its Label to Look.

Now click on the Menu tab and turn on Use Menu. Now under menu items, type 0 under Token and Color under Label then press enter. Next type 1 under Token and Texture Map under Label. Press Apply.

Now select the color node in the network and promote the Color parameter to the parameter list. This brings over the color widget and the color wheel as well as the RGB fields. Press Apply in the Type Properties window to see what this parameter looks like in the brickify_asset parameter interface.

Note: If you press Accept by mistake the new parameter is saved to the asset but you will need to use the Asset menu and choose Open Edit Asset Properties... > brickify to reopen it.

Now select the attributevop node and promote the Texture Map parameter to the parameter list.

In the Parameter description section, click on the Channels tab and change the default value to Mandril.pic. This is a default texture map that doesn’t require a path to load and is a more reliable default. Later you will reload the toylowres.jpg texture map. Press Apply in the Type Properties window and you will see this parameter in the brickify_asset parameter interface and the Mandril texture map is now coloring the bricks.
To make it clear which parameters are associated with which shape, you can disable and enable them based on the menu choice. Click on the **Color** parameter and in the **Disable When** field, enter `{ look != 0 }.

This tells this parameter to disable whenever **Color** has not been chosen in the Look menu. Next, click on **Texture Map** and in the **Disable When** field enter `{ look != 1 }.

Press **Apply** and test the results using the **Shape** menu. You can see the parameters disabling when they are not needed. You could also hide them with the **Hide When** option but disabling is fine for now.

The **brickify_asset** texture map parameter now defaults to **Mandril.pic** which is a more versatile default because it is a texture map that will always be available. To go back to the toy map, you can click on the file selector next to **Texture Map** and again click on `$HIP` then dive into the tex directory and select the **toylowres.jpg** file.

The asset is now using this texture map while **Mandril.pic** is still the default. You can tell it isn't the default because the text is bolded. The Mandril is still the default but now you are using a different map for this instance of the asset.

Now press **Accept** to save the changes to your asset and close the Type Properties panel.

In the Network view, press **u** to go back up one level. With the **brickify_asset** node selected, choose **Assets > Lock Asset > Brickify** from the main menu. Press **Save Changes if prompted**.

If you **double-click** on the **brickify_asset** node to dive down, you can see that the network is greyed out and these nodes are protected. You can only manipulate this asset using its parameters. You will have to unlock the asset to make changes to its inner workings.

Now **Save** your scene file to preserve the work you have done so far. You now have the scene file and the .hda file which is being referenced into your scene to create the asset. You can use this library to create other instances of the asset in this scene or to add the asset to another scene.

In the next section of this lesson, you will test out your asset and find out if it is working properly. It is always good to have one working asset and one for testing to make sure it is doing what you want it to do.

**LOCKING AND UNLOCKING ASSETS**

You can lock and unlock selected assets using the **Assets** menu. When the asset is **locked**, it references the HDA file to determine how the asset behaves. If the asset is **unlocked** the active definition is in your scene file. When you lock it you will be prompted to save if there are changes.

If you **RMB-click** on the asset node you can **Allow Editing of Contents** to unlock the asset or **Match Current Definition** to lock the asset but be careful because there is no prompt to save and changes might be lost.
PART SEVEN: Test the Digital Asset

A Digital Asset can be instantiated more than once in a single scene file. You are going to use this asset on a different piece of geometry to test how it works. It is always good to have a test version available so that changes set to the first asset can be quickly verified. The asset can also be used in other scene files once you have it working properly.

01 Go back to the object level and in the Scene View, use the tab key to get the Squab Test Geometry. Press Enter to place it at the origin then use the handle to move it to the side away from the rubber toy.

Double-click on the new object node to dive down to the geometry level. Select the node and set Scale to 3. This will make the Squab a bit bigger than the rubber toy.

02 RMB-click on the output of the test geometry node and start typing brickify... then select the brickify asset from the menu. This places the asset into this new network.

Set its display flag and you will see another Rubber Toy colored by the Mandril texture map. This is because these are the defaults for this asset.

03 On the brickify asset node, set the Shape parameter to Custom Shape and how the squab is being brickified. The new shape is running through the node network inside the asset to create a unique result.

This is how digital assets become tools that you can put into your pipeline to package up multiple actions into a single node. This is an approach that speeds up your workflow and helps you achieve more consistent results.

04 Select the testgeometry_squab node. From the Asset menu, choose Edit Asset Properties > Squab. In the Properties window, click on the Extra Files tab and select squab_diffuse.jpg.

Click the Save as File button and save it into the tex folder. The texture was stored in the digital asset so that it could be shared along with the asset.

Now use this texture on disk to add color to the bricks using the brickify node’s Texture Map parameter.
PART EIGHT: Animating the Bricks

It is possible to continue adding features to the asset and you will create an automating build-up animation for the bricks. This will involve adding more nodes to our network to make sure the asset has this new functionality. Once the results are saved into the .hda file, the features will be available for use by anyone using this asset in their work.

01 Select the brickify_asset node and choose Assets > Unlock Asset > Brickify from the main menu. Double-click on brickify_asset node then RMB-click on the output of texture_switch and type Group... then select Group by Range. Place the node and set its display flag then set the following parameters:
- Group Name to hide_points
- Group Type to Points
- Method to Start and Length
- Length to (SF-1) * 20
- Invert Range to ON
- Under Range Filter, leave Select to 1 and Of to 1

02 RMB-click on the output of the group node and select Polygon > Blast. Place this node down then using the arrow next to Group, select the hide_points group. Now turn on Delete Non Selected to delete points outside the group. Set the display flag on the blast node.
Press play to watch as the points grow with every frame. Now set the display flag back to the material node at the end of the chain and watch the bricks grow over time.

03 Right now the bricks are coming in from one side instead of from the ground. This is because the points are appearing based on their point numbers. To control this, you need to reorder the points to create the look you want.
RMB-click on the output of the texture_switch node and type Sort... then select the Sort tool. Place this node down and change Point Sort to Along Vector. With this set to 0, 1, 0, the points start at the bottom and go up.
Playback to see this result. Test out different vectors to see how it affects the animation.

04 To let you choose whether you want to animate the brickify effect, you can add another switch node. In the Network view, Press tab and start typing Switch... Choose Switch and place the node down. Rename it animation_switch.
Click on the output of the texture_switch node and feed it into the input of the switch node. Repeat for the blast node. This makes the original shape the first option and the animated effect the second option. Changing Select Input to 1 will reveal the animated bricks but for now keep it at 0.
05 RMB-click on brickify_asset in the path bar and choose Type Properties... Click on the Parameters tab then from the Create Parameters by Type tab, add a separator just after Color. Next, drag the switch node’s Select Input from the Parameter pane to just under the new separator. Set its Name to animate_bricks and its Label to Animate Bricks. Next, change its Type to Toggle which limits to you an on[0]/off[1] setting.

In the Parameter Description section, click on the Channels tab and set the default value to 0 [off].

Click Apply to save changes.

06 From the Create Parameters section in Type Properties, drag an Integer parameter under the animate_bricks parameter. Set its Name to build_speed and its Label to Build Speed. Turn on the Range option then set the first value to 1 and the second value to 20. Click on the lock next to 1 to make sure the number never gets smaller than 1.

From the Create Parameters section, drag a separator to divide the new animation parameters from the other parameters.

Press Accept to save and close the window.

07 This parameter isn’t attached to anything yet but you will now use it to drive the grouprange node’s Length expression. Select the new grouprange node and change the Length expression to: 

\[(F-1)*ch(../build_speed)\]

Now you can change the speed of the animation using this parameter. With the brickify_asset node selected, choose Assets > Save Asset > Brickify from the main menu. This lets you save this expression to the .hda file without re-opening the Type Properties window. Don’t forget this step or else the expression won’t be part of the asset’s definition.

08 With the brickify_asset node selected, choose Assets > Lock Asset > Brickify from the main menu. You now have the brickify effect wrapped up into a custom tool which can be used on different shots by different artists.

Go to the Squab network where the new features are available on the brickify node. Turn on the Animate Bricks toggle and set the Build Speed which might need to be set to around 100 because of the large number of bricks in this shape.

Playback to see the results.

CONCLUSION

You have now created a shareable tool which was built without writing scripts using a node-based workflow which is easily accessible to artists. By saving the HDA to disk, it becomes an asset on disk that can be referencing into multiple shots.

Houdini Digital Assets offer a powerful way for artists to share these kinds of tools in support of a studio-level production. These procedural assets make it easy to automate repetitive tasks and to stay focused on the creative needs of your project.
To use the Houdini Engine in any of these applications, you need to first install the plug-ins using the Houdini installer. This will make the plug-ins available but you may need to take further steps to make the Houdini Engine available within your session.

Visit the following page for details:
SideFX.com/engine
Click on the Engine Plug-ins tab and then click on the desired plug-in for more information.

Once you have the plug-in installed you can load the asset using either the Houdini Engine menu or in UE4, the Import button. This will bring the brickify asset into the viewport while the asset parameters become available for manipulation.

You can also set Shape to Custom Shape and connect the asset to geometry within the host application and the brickification will be applied to that object.

You can also turn on animation for Maya or 3ds Max and play it the sequence using the timeline.

WHAT HAPPENED TO THE BRICK COLORS?
You may notice that none of the brickified shapes are showing the brick colors within the various host applications. This is because the plug-ins don’t always process information the same way as Houdini would. The point colors are still part of the asset but the host application is not receiving this information.

And while the animation works in Maya and 3ds Max, it would not make sense in Unity and Unreal Engine because Houdini assets can not become part of the runtime experience of a game therefore the built-in animation would be ignored. It is important to tailor your assets for the capabilities of the host application. Check out the PROCEDURAL GAME ASSETS FOR UE4 lesson to learn how to create assets specifically for that app.