Developing a Building Construction Timelapse Generator.

Will Harries

MSc Computer Animation and Visual Effects
Masters Project

Abstract

This paper details the creation of a Houdini Digital Asset with the ability to create an animation of a building under construction. The HDA also generates geometry surrounding the construction animation to replicate the trappings of a time-lapse building construction, including scaffolding, cranes and internal girding and structure that initially construct surrounding the object before being removed to leave the provided mesh. Various parameters can also be altered regarding the animation, including the control of timing, noise, direction and the redefinition of primitives as groups of cladding to be constructed. In order to allow for the generated animation to gain the speed and frame-by-frame changes of a given timelapse, features were also developed to distribute a given number of construction worker assets and vehicle assets across the internal structure and surrounding area. Finally, the paper describes the results of blackbox testing and efficiency testing of the HDA for distribution.

Contents

Intro

The intention of this project is to create a Houdini Digital Asset in which a piece of geometry could be provided and an animation of that geometry as if it were a building under construction would be generated procedurally. Similar projects have been attempted for the replication of other phenomena, such as the growth of plants and the workings of bacteria[6] (Muzic et al., 2015), leading to known benefits of recreating a time lapsed process digitally, in comparison to a direct recording both for entertainment and scientific purposes. These include a lack of dropped frames and a cleaner result in comparison to true footage, leading to a product that trades realism for recognisability. This aligns with the intention behind this project.

Influenced by research and previous work, the systems and functions of the project were defined (See Appendix A) and categorised into necessary functions, expected functions and additional functions. The necessary functions include the basic creation of an under-construction building, including parametrised internal structures and external cladding. The expected functions include the ability to redefine cladding size, noise variations on the placement of cladding and the generation of crane and scaffolding structures. Lastly, additional features were considered should time allow. These features included stylisation and animation of the placement of cladding onto the internal structures, the ability to swap internal structures with a solid internal structure made of bricks of equal sizes, and the ability to add assets of workers moving/building rapidly within the structure.

From this list of functions, a basic plan for completion was created, (See Appendix B), with the additional functions planned to follow the deadline of the project in order to allow for prioritisation of base functionality over potential additions beyond the base concept and project idea.

It should be noted following development of the asset that two weeks between B3 and C1 were spent blackbox testing the functionality of the existing features.

Previous Work

While this specific type of procedural animation lacks appropriate research, studies on the basics of procedural animation[2] (ACM SIGGRAPH 2018 Courses, 2018) and the most efficient manners in which they may be implemented[7] (NOVAES CANTÃO and RONALDO BEZERRA OLIVEIRA, n.d.) allowed for an implementation with several key points in alignment with modern procedural animation implementation.

The most significant point of this was that of the input of external information to the procedural animation to create an animation that is repeatable, lightweight and reactive to the current environment. This includes the ability to track lifespans within the animation (in this case the current frame of the animation in comparison to various inputted frame values) and information regarding material outside the animation, such as various colliders (in this case represented by the bounding box of the given geometry significantly influencing the animation).

In tandem with research on procedural animation significant time was spent dissecting timelapse constructions to ascertain their various features and quirks. From this, it was gleaned that significant control needed to be provided for the various steps of a construction, including delays in internal and external structures[3] (Cedar Woods Properties, 2020) both from specific examples and due to variety across each timelapse. These timed segments also need to be controlled in length and speed, with this speed difference applying throughout to emphasise the varying paces of construction in different areas[1] (3XN Architects, 2022), especially with external structures that often vary wildly in passing between timelapses[4] (Mena's Smart Home, 2021).

Secondary to basic timing functions, other features and attributes were influenced by reference footage. This includes using differing materials across the construction[12] (Work Zone Cam, 2014) and placing the assets of workers across the construction at both full opacity and various levels of transparency[11] (Watch a 57-Story Building Go Up in 19 Days, 2015). The use of reference footage also influenced the design principle to create a simpler depiction of a timelapse viewable and recognisable from a distance, as opposed to a full simulation with detailed and accurate architectural and construction methods[5] (MK timelapse, 2017).

Technical background

To ensure that the produced HDA is both of given technical pedigree and shares the design principles of the wider Houdini sphere, several HDA's were downloaded and analysed in advance of the project. Several things were learned as a consequence, influencing the design of the construction generator. Often assets lacked hidden features, choosing to instead disable features when not in use. Many assets also lacked certain controls that one would expect from their function, likely to prevent a user disabling the asset entirely, this prompted the creation of "advanced settings" where controls that would likely disable the asset if used carelessly are placed and properly labelled.

The assets within Houdini's environment that are most downloaded[9] (Orbolt.com, 2019) also contain well organized node diagrams, often splitting nodes into functionality groups or full subnetworks.

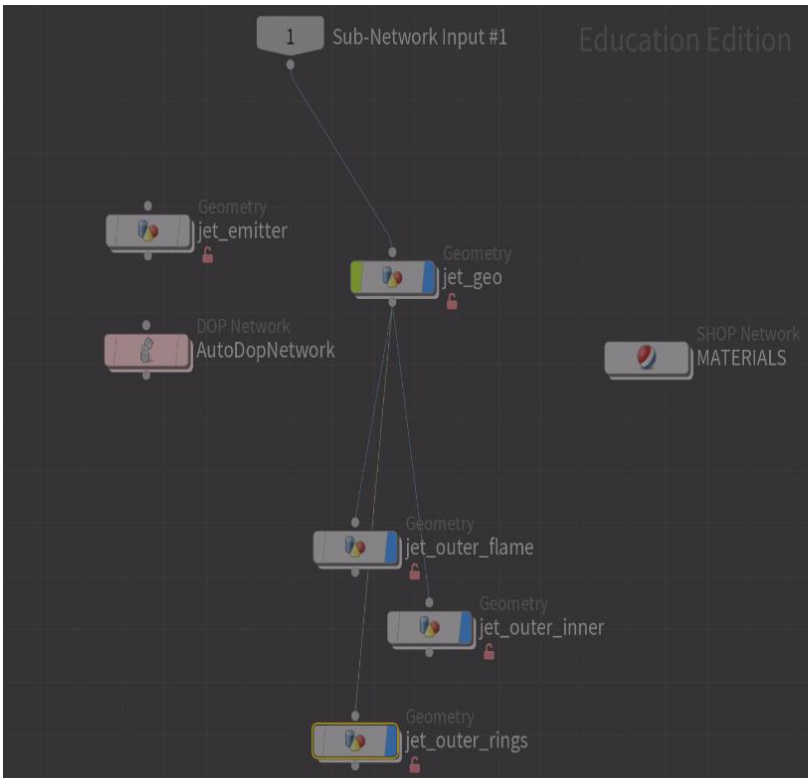

Figure 1: Orbolt multi-subnet division.

These assets will also make use of caching different sections of the final asset, contradicting the efficiency based requirement set for the project and possibly detailing a feature for future work. The intent to create a HDA of several subnets was also barred through the use of shared data within the implementation of the HDA, the ubiquity of which prevented the conversion of each separate function into a set of inputs and outputs.

Research was also done into previous attempts at replicating the nature of timelapsed footage in a digital environment, with this leading to several key principles followed in the design of the HDA. This includes (as a companion influence alongside procedural animation) an emphasis on using the tracking of time to directly influence the animation via tracking the frame number both as a method to track animation progress and to dictate the beginnings and ends of various phases of the animation. In the case of the studied paper[10] (Przemyslaw Prusinkiewicz, Hammel and Mjolsness, 1993), this took the form of a frame-based divergence in the animation, in much the same manner as the construction generator is intended to diverge between versions at different points and dependant on different geometry. Notably this requires the tracking of the creation of various primitives of the provided geometry, something difficult to achieve in Houdini but will have to be worked around.

Development and methodology

A1.1 Mesh Construction

The gradual construction of the mesh was initially solved relatively easily through the use of a single growing sphere that collected points within the provided geometry and added them to a group for display. This sphere would be controlled by a simple equation, eventually taking the following form after the implementation of several features:

$$S = \left(\frac{F - D - E}{I + L + O}\right)(1 + |d|)$$

Where:

F – The current animation frame.

D – The start of the construction animation.

E – A delay for the construction of external structures first.

O – A variable for time between internal structures and their external cladding.

L – A variable for the length of the cladding's construction.

I – A variable for the length of the construction of internal structures.

d – A vector for the current direction the animation should depict the construction building in.

This allows for parameters to delay the construction of parts of the animation through additions and subtractions to each end of the equation, and a construction speed to be set by the user. The ability to add positive and negative delays to the cladding construction also allows for a user created timeline of the construction's different layers, with elements being able to be placed before or after one another through this.

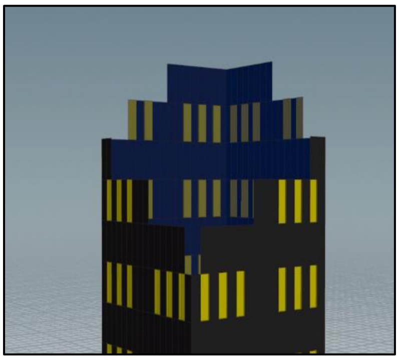

Figure 2: Cladding Construction.

A1.2 Removing disconnected primitives

Due to the shape of the construction's progress being that of a sphere, it was common to have unsupported geometry during the construction. In order to prevent this, a function was developed to group the largest piece of contiguous geometry, displaying that instead of the mesh's whole progress as a means to remove floating mesh elements. As this creates several issues for input geometry without contiguous pieces, this feature remains togglable and was developed alongside several other features to prevent cladding floating away from the construction.

A2.1 Internal Structure Construction

The girding of an internal structure was created through defining a set of points of a given density from the inputted mesh, before copying to each a box of exact dimensions to bridge the gaps between each point, creating a 3D grid internally restricted to the given mesh. The defined grid would be placed a density-based distance from the mesh, as to prevent excessive bleeding of the internal structure outwards. This would then be converted to a wireframe geometric structure, the "pipes" of which being of a user defined radius.

From there a floor of a height equal to each "pipe" and square dimensions matching the grid density is copied to the grid, allowing the wireframe structure to gain floors akin to the inner structure of a large construction. The same sphere technique is deployed on the grid of points prior to conversion into geometry to allow construction to avoid the placement of stray primitives without connection.

A later addition to the internal structures allows for the gap between the internal structures and cladding, with rays being cast from the given grid of points of a distant capped at the distance between each point in the grid. This allows for gaps in the internal structure otherwise not accounted for by the equally spaced grid of points to be filled in, greatly reducing the depiction of floating cladding.

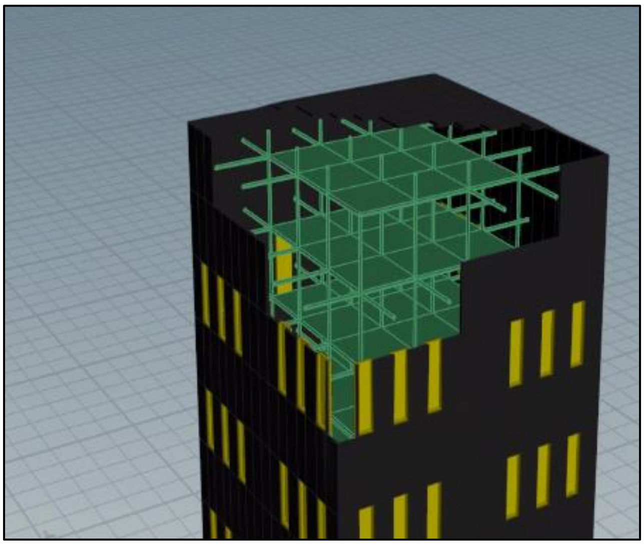

Figure 3: Internal Structures.

A2.2 Building Direction

During this time, the possibility of changing the construction direction was considered, due to both the “dull” image of uniform internal structure construction and the issue of a ceiling often spawning somewhat simultaneously should progress be made from the bottom up. As a consequence, a user defined “build direction” was created as a vector, allowing a given direction to be defined for the building to be constructed in the direction of.

The speed of the construction was also partially altered to consider the magnitude of the vector, preventing issues should the user provide a vector that is not normalised, something that would otherwise increase the duration of the construction.

B1. Segmentation

In order to allow for more complex meshes to have alterable amounts of “cladding” as constructions, a function was developed to break down a given mesh into segments along cube grid of user defined density. First the mesh is converted to a cuboid bounding box to allow for even segments. From there the process mirrors creating the internal structure grid, starting with the placement of points at equal distances throughout the provided volume, converting them into cubes of equivalent measurements.

Again, similar to the construction of the internal structures the points within the volume are placed in cuboids, using the same grouping technique to then gather the mesh points in each cuboid and display them.

Later, to allow for re-defined cladding to be animated and stylised, it was necessary to divide the redefined cladding into groups, assigning each grouping cuboid a value to be transferred to its corresponding geometry. These values then allowed the stylisation to apply to these primitives in a group rather than as individuals primitives and radically improved this feature’s resilience to stranger geometry.

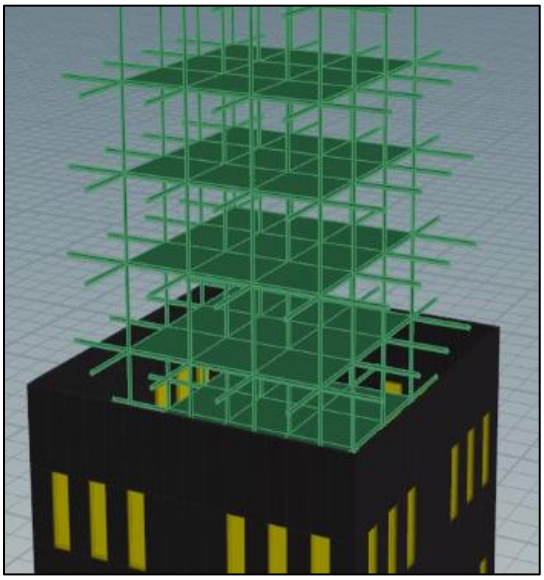

Figure 4: Segmentation of mesh into groups.

B2. Noise

Noise was determined to be a key concern early on within planning of the HDA to allow for natural variance in construction speed both between different instances of the asset and in the construction itself. Two types of noise placement were considered for this, noise at the edge of construction and "islands" of empty space scattered across the entire construction, eventually filling themselves out as the building finishes construction.

This is achieved by creating a cross section of a given width and applying noise of a given density, removing more or less geometry to create a gradient of noise towards the fully constructed area. The "islands" are then created by removing a given percentage of points in advance from the geometry, decreasing said percentage of points over time until the otherwise constructed cladding is unblemished. The percentage of points is notably defined by the above algorithm for cladding progress, in order to allow for the number of islands to be reflective of all other variables within the animation.

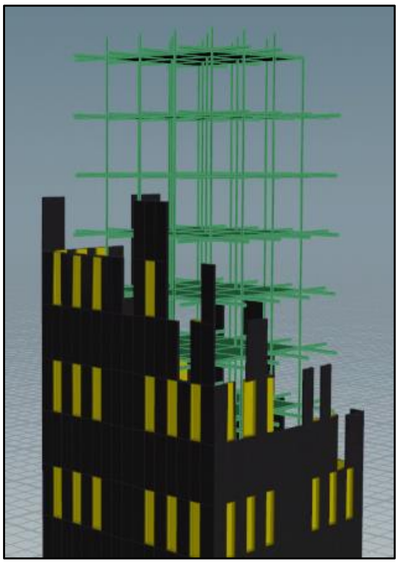

Figure 5: Noise variation.

B3. External Structures

Two external pieces were considered for the structures outside of the building, including cranes placed at a random points at the base of the construction and pole structures similar to internal construction to represent basic scaffolding.

Scaffolding structures were achieved by scaling up the width and depth of the provided geometry to create a ring of points surrounding the original mesh. From there similar functions to that of the internal structures are provided to convert the ring into a scaffold that can surround the construction. This method allows for a set of points at the base of the scaffolding to place cranes, however it creates wildly different amounts of scaffolding depending on the provided geometry. Therefore, the ability to scale the number of points subtracted to create the ring was provided.

Eventually, the ability to define scaffolding structure orientations was considered, prompting the creating of separate code to detect the placement of scaffolding in relation to overhangs and complex geometry as well as to redefine the proportions of the cuboids. This required the use of bespoke code to create a number of points to fit within a volume and scatter them, the code accounting for different distances between points should the scaffolding contain non-cubic proportions.

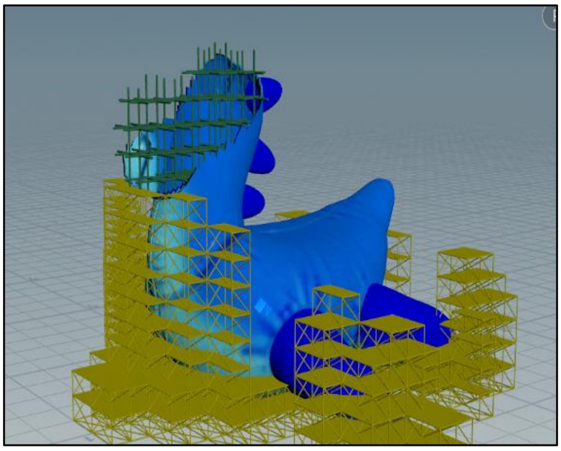

Figure 6: Scaffolding on overhanging geometry.

Crane structures were achieved via taking a cuboid of defined width and depth, while matching height to the provided geometry. These then had points scatted within them separated by the same depth and height values, creating a column of equally separated points. These points were then converted to a polywire frame similar to the internal structures to create crane towers, copied to a randomly selected set of points across the ring.

From there a basic tower crane top was modelled as several separate elements, each then placed in relative to the height of the given geometry. These crane head pieces were then placed at random rotations on the crane towers. Initially, this caused several collisions between cranes, and therefore an algorithm was developed to calculate the intersection point of the cranes and rotate them away from intersecting where the crane head extends to. The fomulae to do so can be found below (Appendix D).

Later, additional modelling was added to the crane's top, allowing for the depiction of wiring and triangular support structures visible on some cranes. This is done through the detection of either end of the crane arm, as well as several points on an additional column at the crane's centre, before generating three polywires between the selected points.

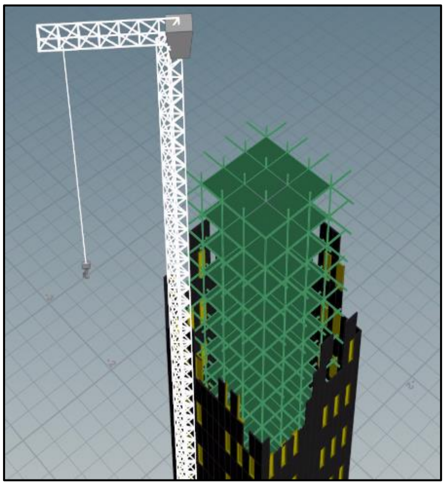

Figure 8: Crane generation.

C1.1 Complex assembly

Two types of complex, animated assembly were considered for the construction of given geometry. The first was a stylisation of construction that produced a flipping animation for each piece of cladding. The other was a simple depiction of each piece of cladding extruding outward smoothly instead of each cladding piece appearing individually.

Due to algorithmic nature of the Houdini engine, defining an efficient method in which all defined cladding animates individually in a way related but independent to the construction of the wider geometry was quite a challenge. Eventually it was decided that rotations were required to be pre-calculated through a similar methodology as the above scaling function, instead relating to distance from the furthest point in the inputted geometry rather than current frame. This allows the primitives to be ordered in "construction time" and rotated accordingly. From there, the code can be tuned in order to produce a rotation of 90˚at a given point in the animation, starting at a rotation and decreasing over time until reach that point. Then cladding above 90˚rotation can be removed and those under fixed to an unrotated state, producing a gradual appearance of defined cladding, each rotating 90˚into place before being fixed in place. The full breakdown of these equations can be found below (Appendix E).

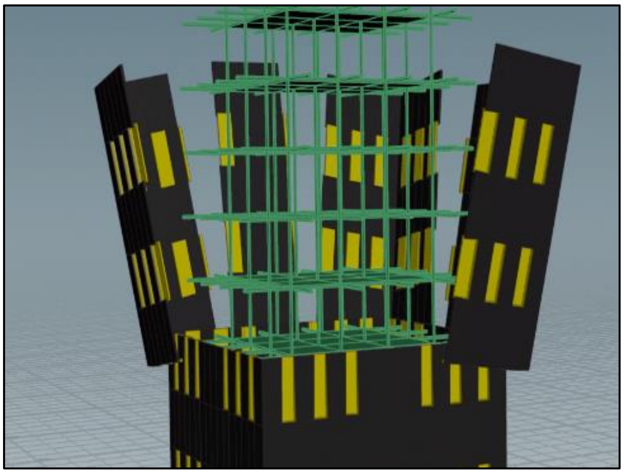

Figure 9: Flip stylisation.

In comparison the extruding method of stylisation was much easier to implement, simply taking the initial model and using a Boolean function to remove the sphere of constructed primitives used throughout the HDA. This allowed the cladding to be created gradually, growing smoothly over the internal structures. The addition of cladding features, such as noise, was also considered through both stylisations, requiring in the smooth extruding of the cladding the addition of pre-calculated randomly defined primitives to be ahead of extruding construction at a user defined rate before being "booleaned" with the noise sphere as with normal noise creation.

C1.2 Cladding Thickness

To ensure cladding's appropriate look during animation, it was decided a thickness variable should be implemented to ensure cladding appears realistic. This was done simply, through a minute extrusion of the cladding backwards to generate the appropriate geometry, before scaling all generated primitives inwards to a user defined thickness.

C2. Internal Construction

While initially the internal construction feature was mostly copied from the initial generation of the internal structure's grid but without the conversion into a wireframe grid, it was noted this left large gaps between the internal blocks and cladding. As a consequence, similar functionality was deployed as to the struts placed along the internal structure to depict it supporting the cladding. From there, primitives were created instead of struts so that no gaps were present or visible between internal structure and cladding. It was necessary to flip the primitives inside-out in order to have the additional primitives shade correctly.

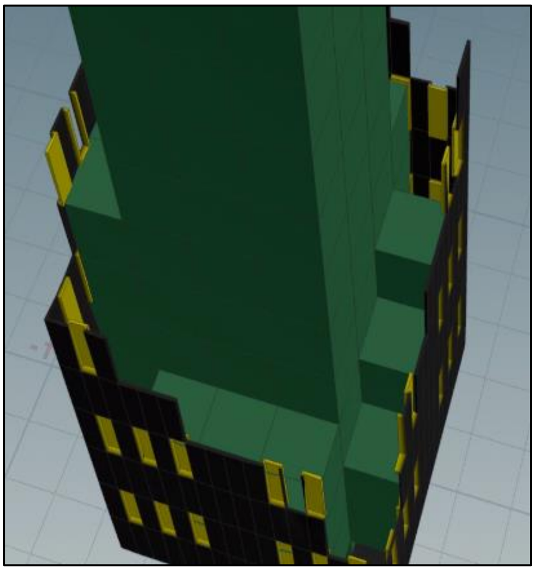

Figure 10: Filled in Block Structure.

C3. Internal Animation

Several systems were considered to allow a set of assets to be added to have those assets be distributed across the inside of a given structure. However, after several different attempts it was decided that a system similar to a connectivity SOP would be included. This allows the user to input a geometry node and supply the name of a class attribute. Using this class attribute, the generator then splits each set of primitives with a varying class attribute into individual assets and places them onto the internal structure of the animation.

In collaboration with a detection system that allows for the distinction of suitable primitives for the assets to stand upon using angle detection, the assets are then randomly scattered across the construction, changing position every frame as to replicate a timelapses' random capturing of mobile variables, the rotation of each copied asset would also be randomised every frame.

A duplicate system was also created for the scattering of objects along the ground plane at a given distance around the constructing geometry. This allowed for the distinction between provided construction assets for scattering within the internal structure of the building and assets that would remain outside the building (construction workers and construction vehicles respectively).

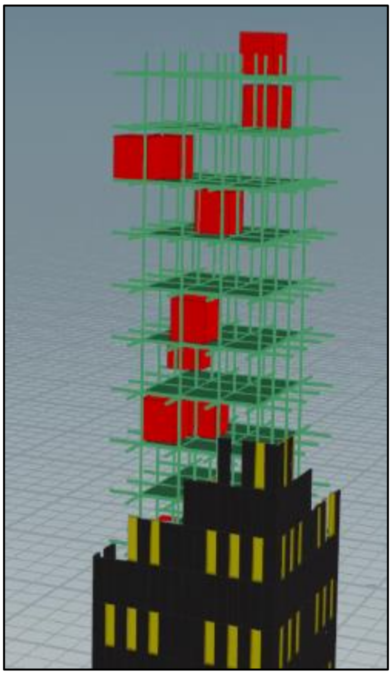

Figure 11: Life placement inside construction.

Working of tool

Without the ability to use modular testing, it was decided that the best way to ensure the HDA's quality was to produce blackbox tests for all inputs within the HDA, including tests for interactions between different functions such as stylised animation working with cladding recalculation.

Within the below table (see Appendix C) are the results of each test on each control over time, initially testing prior to the development of features categorised as "Additional", before testing following the development of stylised construction and finally following internal asset animation.

To test the stated system requirements of the asset, Houdini's internal performance metrics were deployed in order to discern nodes of high demand (such as the wireframe node initially used to create internal structures). From there, a cache of the whole animation would be stored and compared so that it can be confirmed to meet requirements, and nodes replaced or made more efficient if this is not the case.

Rendering tests

While the rendering of the animated geometry created by the construction generator can be found within the stage layer of Houdini, only the colouring within the object and geometry layer displays. This is seemingly in line with the general design philosophies of the Houdini environment[8] (Orbolt.com, 2025), this approach replicated other HDAs of buildings with customisable features.

Due to moving away from the initial concept of building a tool that covers the full pipeline of the assets production and instead pairing back the features to primarily focus on geometry, certain features observed during the research phase were also dropped. This includes the application of differing materials (instead represented by allowing different sections of the construction to be given materials independently of the HDA) and the varying transparencies of internally animating assets (something that would be required to be provided by the user).

However, the outputting of the final geometry to allow for it to have material attachments has been adequately considered, with the node outputting all the geometry as a singular object while simultaneously outputting the different layers of the animation for potential texture application and reconstruction. Notably, any material work present on the object prior to animation remain on the cladding of the object following the animation's generation.

Conclusion

As a summary of what has been achieved, the final project converts the provided geometry into an "under construction" building that can then be altered and edited depending on the preferences of the user. The generated structure can then progress from unfinished to duplicating the original geometry with its materials and texturing. The timing of each aspect of the construction can then be altered to match user preferences, as can the addition of noise and external structures for a more realistic construction. Other features have also been implemented to fully overhaul the aesthetics of the construction to stylise the resulting timelapse style animation, to allow for less realism-based applications, and asset placement has been developed to allow the user to replicate workers within their timelapses for even greater realism.

All these features have been tested as to both their functionality and their ability to work with each other in different circumstances and combinations. The entire HDA has also been efficiency tested, notably coming under a stated target of 30 seconds for generating the animation in most circumstances.

Reflection

While work within the project can on the whole, be considered valuable, and the project overall can be considered a success, several alterations to development methodology could be made to improve the project in hindsight or moving forward. As a significant portion of development had to be reallocated to the testing of previous functions as they relate to the user and to each other, more significant time for testing and more efficient testing methodologies would be an easy improvement to the current development cycle. Testing with animators and within industry would also benefit the usability of the HDA greatly.

As well as this, several assumptions regarding the ease of various functionalities and the existence of certain standards within the Houdini ecosystem would be better revised and discarded as needed. Because of a distinct lack of standardisation across HDAs, it would likely improve the project to research into possible markers of quality to which this project could be judged, with these deriving themselves from interviews with users of the Houdini engine.

Future work

Several features could be added to benefit the project in the future. This included the addition of possible foundations to the building during its construction as well as other initial constructions to replicate the start of a building site prior to the creation of any geometry to represent the construction itself. On analysis of conventional Houdini HDA approaches, it may be also worthwhile to remove the requirement to avoid caching data within the HDA, developing methods to cache parts of the construction animation within the node to better improve generation speed through the reuse of unchanged data when the asset's appearance is being customized.

As mentioned above, more efficient methodologies for testing could well be created in order to ease the development of this project and other HDAs, however the development of proper module testing for Houdini is likely beyond the scope of this project and likely to be a significant project in its own right. As part of this, the HDA would benefit from a stress test using several versions of the asset on differing geometries, possibly to point to the eventual development of TOP multi-threading between the different sections of the HDA and (as used within previous work[10] (Przemyslaw Prusinkiewicz, Hammel and Mjolsness, 1993), possibly altering divergence in the animation on occasions where a change between frames stresses the system beyond acceptable levels.

References

1. 3XN Architects (2022). *Quay Quarter Tower - Construction Timelapse*. [online] YouTube. Available at: https://www.youtube.com/watch?v=IHDR1s54jPo [Accessed 24 Oct. 2024].

2. ACM SIGGRAPH 2018 Courses. (2018). doi:https://doi.org/10.1145/3214834.

3. Cedar Woods Properties (2020). *Huntington Apartments Time Lapse*. [online] YouTube. Available at: https://www.youtube.com/watch?v=KV3wcnMOs30.

4. Mena's Smart Home (2021). *Building Construction Time laps*. [online] YouTube. Available at: https://www.youtube.com/watch?v=FgFQYLC38Pc [Accessed 15 Jul. 2025].

5. MK timelapse (2017). Construction of the breathtaking TORRE REFORMA in MEXICO CITY. - CINEMATIC TIMELAPSE 4K. [online] YouTube. Available at: https://www.youtube.com/watch?v=LAmkFK5BsAo [Accessed 1 Aug. 2025].

6. Muzic, M.L., Waldner, M., Parulek, J. and Viola, I. (2015). Illustrative Timelapse: A technique for illustrative visualization of particle-based simulations. pp.247–254. doi:https://doi.org/10.1109/pacificvis.2015.7156384.

7. NOVAES CANTÃO, V. and RONALDO BEZERRA OLIVEIRA, S. (n.d.). AN IMPLEMENTATION GUIDE TO PROCEDURAL ANIMATION. *CONTECSI International Conference on Information Systems and Technology Management*. [online] doi:https://doi.org/10.5748/18contecsi/pse/esd/6726.

8. Orbolt.com. (2025). Point Cloud Volume. [online] Available at: https://www.orbolt.com/asset/SideFX::pointcloudvolume [Accessed 2 Aug. 2025].

9. Orbolt.com. (2019). Pyro Jet Exhaust. [online] Available at: https://www.orbolt.com/asset/SideFX::pyrojetexhaust [Accessed 1 Aug. 2025].

10. Przemyslaw Prusinkiewicz, Hammel, M. and Mjolsness, E. (1993). Animation of plant development. International Conference on Computer Graphics and Interactive Techniques. doi:https://doi.org/10.1145/166117.166161.

11. Watch a 57-Story Building Go Up in 19 Days. (2015). *YouTube*. Available at: https://www.youtube.com/watch?v=N6f_sayw0mM.

12. Work Zone Cam (2014). *Construction Time-Lapse: Single Family Home Built in 5 Months*. [online] YouTube. Available at: https://www.youtube.com/watch?v=CMrpeNaNh84 [Accessed 15 Jul. 2025].

Appendixes

Appendix A: Requirement breakdown

Type	Requirement	Necesity	Fufilled
Function	As a user, I must be able to input geometry, and have the output be that geometry growing over time gradually by connected primitives.	Must (A)	✓
Function	As a user, I must be able to see scaffolding be created at the centre of the geometry in the shape of the geometry's eventual mesh, before the scaffolding is covered by the primitives.	**Must (A)**	✓
Function	As a user, I expect to be able to customise how the geometry is split into individual growing primitives.	**Expect (B)**	✓
Function	As a user, I expect the placement of panels to be irregular, with the level of irregularity at a user defined level.	**Expect (B)**	✓
Function	As a user, I expect the animation to include the external structures and scaffolds necessary for the object's construction t o be visually recongisable, with their number being customisable.	**Expect (B)**	✓
Function	As a user, I could customise the method in which each primitive is placed onto the construction out of a set of defined presets.	**Additional (C)**	✓
Function	As a user, I could choose between the outputted animation being that of a hollow structure, or that of a solid brick-based structure.	**Additional (C)**	✓
Function	As a user, I could toggle the placement of blurred figures within the geometry, to replicate the capture of workers on the constructing object.	**Additional (C)**	✓
System	The asset must remain fully documented with appropriate tool information and an appropriate help-card following the implementation of a new requirement.	**Must (A)**	✓
System	The asset must be streamlined for a user, being installed without use of Houdini's underlying systems.	**Must (A)**	✓
System	The asset must have a clear user interface, with each function of the asset clearly demonstrated by its user interface controls.	**Must (A)**	✓
System	The asset must be able to render the full output animation to the quality displayed in the viewport, with appropriate material assignments for each section of the animation.	**Must (A)**	✓
System	The asset should generate its animation without the need for intermediate file caching or wait times of over 30 seconds for an iteration of the animation.	**Expect (B)**	✓

System	The asset could have an appropriate store page or online download link with usage examples.	**Additional (C)**	

Appendix B: Feature Planning

Requirement	Planned Completion	Completion Date
A1	24/06	17/06
A2	03/07	24/06
B1	15/07	27/06
B2	18/07	30/06
B3	25/07	08/07
C1	05/08	03/08
C2	12/08	05/08
C3	21/08 (overestimation)	16/08

Appendix C: Blackbox Testing

Out of 196 tests, 2 failed and 4 were not implemented, with 2 tests causing more slowdown than acceptable should certain UI conditions be met.

Feature	**Control**	**Type**	**Test**	**Pre-C**	**C1**	**C3**
General	Delay	Value	Function	🟩	🟩	🟩
			Max	🟧	🟩	🟩
			Min	🟩	🟩	🟩
			Negative	🟩	🟩	🟩
Mesh	Animation Length	Value	Function	🟩	🟩	🟩
			Max	🟩	🟩	🟩
			Min	🟩	🟩	🟩
			Negative	🟩	🟩	🟩
	Cladding Delay	Value	Function	🟧	🟩	🟩
			Max	🟧	🟩	🟩
			Min	🟧	🟩	🟩
			Negative	🟩	🟧	🟩
Internal structure construction	Animation Length	Value	Function	🟩	🟩	🟩
			Max	🟧	🟩	🟩
			Min	🟩	🟩	🟩
			Negative	🟩	🟩	🟩
	Density	Value	Function	🟩	🟩	🟩
			Max	🟩	🟩	🟩
			Min	🟩	🟩	🟩
			Negative	🟩	🟩	🟩
	Build Direction	Vector	Function	🟩	🟩	🟩
			Max	🟩	🟩	🟩
			Min	🟧	🟩	🟩
			Negative	🟧	🟩	🟩
	Internal Colour	Vector	Function	⬛	🟩	🟩
			Max	⬛	🟩	🟩
			Min	⬛	🟩	🟩
			Negative	⬛	🟩	🟩
	Solid Internals	Boolean	True	⬛	🟩	🟩
			False	⬛	🟩	🟩
	Solid Internals Density*	Value	Function	⬛	⬛	🟩
			Max	⬛	⬛	🟧
			Min	⬛	⬛	🟧
			Negative	⬛	⬛	🟩
Segmentation	Redefine toggle	Boolean	True	🟩	🟩	🟩
			False	🟩	🟩	🟩
	Panel Split	Value	Function	🟩	🟩	🟩
			Max	🟧	🟩	🟩
			Min	🟩	🟩	🟩
			Negative	🟩	🟩	🟩
	Connected only toggle	Boolean	True	🟩	🟩	🟩
			False	🟩	🟩	🟩
Noise	Noise toggle	Boolean	True	🟩	🟧	🟩

			False	🟩	🟩	🟩
	Noise size	Value	Function	🟩	🟩	🟩
			Max	🟩	🟩	🟩
			Min	🟩	🟩	🟩
			Negative	🟩	🟩	🟩
	Noise density	Value	Function	🟩	🟩	🟩
			Max	🟩	🟩	🟩
			Min	🟩	🟩	🟩
			Negative	🟩	🟩	🟩
	Island toggle	Boolean	True	🟩	🟩	🟩
			False	🟩	🟩	🟩
	Island size	Value	Function	🟩	🟩	🟩
			Max	🟩	🟩	🟩
			Min	🟩	🟩	🟩
			Negative	🟩	🟩	🟩
External Construction	Animation length	Value	Function	🟩	🟩	🟩
			Max	🟩	🟩	🟩
			Min	🟩	🟩	🟩
			Negative	🟩	🟩	🟩
	Delay length	Value	Function	🟧	🟩	🟩
			Max	🟧	🟩	🟩
			Min	🟧	🟩	🟩
			Negative	🟧	🟩	🟩
	Deconstruction start	Value	Function	🟩	🟩	🟩
			Max	🟩	🟩	🟩
			Min	🟩	🟩	🟩
			Negative	🟩	🟩	🟩
	Deconstruction length	Value	Function	🟩	🟩	🟩
			Max	🟧	🟩	🟩
			Min	🟩	🟩	🟩
			Negative	🟩	🟩	🟩
	Scaffold toggle	Boolean	True	🟩	🟩	🟩
			False	🟩	🟩	🟩
	Scaffold height	Value	Function	🟩	⬛	⬛
			Max	🟧	⬛	⬛
			Min	🟩	⬛	⬛
			Negative	🟩	⬛	⬛
	Scaffold density	Value	Function	🟩	🟩	🟩
			Max**	🟩	🟩	🟩
			Min	🟩	🟩	🟩
			Negative	🟩	🟩	🟩
	Scaffold Dimensions	Vector	Function	⬛	🟩	🟩
			Max	⬛	🟧	🟩
			Min	⬛	🟧	🟩
			Negative	⬛	🟩	🟩
	Scaffold Colour	Vector	Function	⬛	🟩	🟩
			Max	⬛	🟩	🟩
			Min	⬛	🟩	🟩
			Negative	⬛	🟩	🟩
	Scaffold Gap	Value	Function	⬛	🟩	🟩

			Max	⬛	🟩	🟩
			Min	⬛	🟩	🟩
			Negative	⬛	🟩	🟩
	Beam Type	Dropdown	Rows	⬛	🟩	🟩
			Columns	⬛	🟩	🟩
			Rows and Columns	⬛	🟩	🟩
			Triangles	⬛	🟩	🟩
			Quadrilaterals	⬛	🟩	🟩
			Alternating Triangles	⬛	🟩	🟩
			Reverse Triangles	⬛	🟩	🟩
	Scaffold Floor	Boolean	True	⬛	🟩	🟩
			False	⬛	🟩	🟩
	Crane toggle	Boolean	True	🟩	🟩	🟩
			False	🟩	🟩	🟩
	Number of cranes	Value	Function	🟩	🟩	🟩
			Max	🟩	🟩	🟩
			Min	🟩	🟩	🟩
			Negative	🟩	🟩	🟩
	Crane arm length	Value	Function	🟩	🟩	🟩
			Max	🟩	🟩	🟩
			Min	🟩	🟩	🟩
			Negative	🟩	🟩	🟩
	Crane Density	Value	Function	🟩	🟩	🟩
			Max	🟩	🟩	🟩
			Min	🟩	🟩	🟩
			Negative	🟩	🟩	🟩
	Crane Colour	Vector	Function	⬛	🟩	🟩
			Max	⬛	🟩	🟩
			Min	⬛	🟩	🟩
			Negative	⬛	🟩	🟩
	Crane Slope	Value	Function	⬛	⬛	🟩
			Max	⬛	⬛	🟩
			Min	⬛	⬛	🟩
			Negative	⬛	⬛	🟩
	Crane Wire Height	Value	Function	⬛	⬛	🟩
			Max	⬛	⬛	🟩
			Min	⬛	⬛	🟩
			Negative	⬛	⬛	🟩
	Add Crane Wire	Boolean	True	⬛	⬛	🟩
			False	⬛	⬛	🟩
	Crane Animation	Boolean	True	⬛	⬛	🟩
			False	⬛	⬛	🟩
	Crane Animation Rate	Value	Function	⬛	⬛	🟩
			Max	⬛	⬛	🟩
			Min	⬛	⬛	🟩
			Negative	⬛	⬛	🟩
Stylisation	Flip Stylise	Boolean	True	⬛	🟧	🟩

			False	▨	🟩	🟩
	Stylise Portion	Value	Function	▨	🟩	🟩
			Max	▨	🟩	🟩
			Min	▨	🟩	🟩
			Negative	▨	🟩	🟩
	Extrusion stylise	Boolean	True	▨	🟧	🟩
			False	▨	🟩	🟩
	Cladding Width	Value	Function	▨	🟩	🟩
			Max	▨	🟩	🟩
			Min	▨	🟧	🟩
			Negative	▨	🟧	🟩
Advanced	Internal structure scale	Value	Function	🟧	🟩	🟩
			Max	🟧	🟩	🟩
			Min	🟧	🟩	🟩
			Negative	🟧	🟩	🟩
	Internal structure width	Value	Function	🟩	🟩	🟩
			Max	🟩	🟩	🟩
			Min	🟧	🟩	🟩
			Negative	🟩	🟩	🟩
	External width	Value	Function	🟧	🟩	🟩
			Max	🟧	🟩	🟩
			Min	🟧	🟩	🟩
			Negative	🟧	🟩	🟩
	Flip Multiplier	Value	Function	▨	🟩	🟩
			Max	▨	🟧	🟩
			Min	▨	🟩	🟩
			Negative	▨	🟧	🟩
	Solid Internal Gap Fill	Value	Function	▨	▨	🟩
			Max	▨	▨	🟩
			Min	▨	▨	🟩
			Negative	▨	▨	🟩
	Add life asset	Boolean	True	▨	▨	🟩
			False	▨	▨	🟩
	Scale to grid	Boolean	True	▨	▨	🟩
			False	▨	▨	🟩
	Asset Class	String	Function	▨	▨	🟩
	Points for Distribution	Value	Function	▨	▨	🟩
			Max	▨	▨	🟩
			Min	▨	▨	🟩
			Negative	▨	▨	🟩
	Toggle Ground Life	Boolean	True	▨	▨	🟩
			False	▨	▨	🟩
	Ground Class	String	Function	▨	▨	🟩
	Ground Points	Value	Function	▨	▨	🟩
			Max	▨	▨	🟩
			Min	▨	▨	🟩
			Negative	▨	▨	🟩
	Life Seed	Random Seed	Function	▨	▨	🟩
			Negative	▨	▨	🟩
	Crane Seed		Function	▨	▨	🟩

		Random Seed	Negative	⬛	⬛	🟩
	Noise Seed	Random Seed	Function	⬛	⬛	🟩
			Negative	⬛	⬛	🟩
	Island Seed	Random Seed	Function	⬛	⬛	🟩
			Negative	⬛	⬛	🟩
	Ground Life Seed	Random Seed	Function	⬛	⬛	🟩
			Negative	⬛	⬛	🟩

Key:

🟧	**Test failed.**
🟩	**Test Succeeded.**
⬛	**Control non-existence at time of testing.**

***It should be noted here “failure” is due to the corners of square geometry reacting poorly to this specific feature.**

****Aformentioned issue in which scaling scaffolding seemingly causes lag due to UI updates on version of Houdini that have been previously windowed.**

Appendix D: B3 Fomulae

The intercept point of a given crane's current angle and another crane's current rotation is considered a position vector in which Y position is not considered.

The X value of the interception point is defined as:

$$i_x = \frac{(O_z - (\tan(v) * O_x)) - (P_z - (\tan(a) * P_x))}{\tan(a) - \tan(v)}$$

$$i_z = (\tan(a) * i_x) + (P_z - (\tan(a) * P_x)$$

Where:

i – The point of eventual intecept.

P – The position of a given crane.

O – The position of the intercepting crane.

a – The current rotation of a given crane.

v – The current rotation of the intercepting crane.

Crane rotation is altered should the distance of the intercept between crane arms be less than the distance of the actual modelled crane arm, the distance being defined as:

$$d = |i - v| * 2$$

Where:

v – the position (Y position not considered) of another crane.

d – calculated intercept distance.

Appendix E: C1 Equations

$$d = |D| * (1 + f)$$

Where:

d - The distance a given piece of cladding is from the final point of construction. This allowing cladding to be iterated through in creation order.

D – A ray cast in the direction of the animation’s construction at a size equal to the final radius of the animation’s bounding sphere.

f – A decimal multiplier given to the distance to allow for customization of animation severity.

$$W = \frac{90}{p * (F * 2)}$$

Where:

W – Rotation within the distinguished user defined section.

P – User defined section of stylisation.

F – Defined as the total frame in the whole animation.

$$Fr = \frac{d}{R} * 2F$$

Where:

R – Radius of the geometry’s bounding sphere.

Fr – rotation in terms of Frames.

$$Fin = \left(\left(90 * \frac{Fr}{p * (F * 2)}\right) + W\right) - (90 * \frac{cf}{p * (F * 2)})$$

Where:

cf – Current frame.

Fin – The final rotation of a primitive or cladding.