Stylized Motion Blur Houdini Digital Asset

Omar Fayed

Bournemouth University

August 21, 2025

ABSTRACT

The design and development of FX simulations in Houdini is a time consuming process and can take up a big chunk of artists' and studios' time on a project. If a specific effect is assigned to a specific character or object throughout the project, it is logical to build this effect procedurally. This tool will offer flexibiltiy to apply the effect according to the corresponding animation of the shots. A well built procedural tool can save artists time and avoid repetitive work. For fast paced projects, a library or a procedural tool kits help save time and efforts and often does the job. The project's inspiration was to offer a procedural HDA (Houdini Digital Asset) to apply effects and simulations for animated objects and characters. The effects in this project are targeted for adding stylization for motion trails and blurs of the input animations. The project will allow the user to choose from various effects and customize it as desired. The main theme of the effects are 2D-style blur and particle effects.

ACKNOWLEDGEMENTS

I would like to first mention that I am grateful to have been part of the NCCA and be taught and supervised for this project by Jon Macey and Jian Chang. I would like to extend my thanks to the help and support I received from my friends and colleagues in this course. Finally, I would love to acknowledge the role and support of my family that allowed me to pursue the master's degree and all the success I have reached in my career so far.

TABLE OF CONTENTS

LIST OF FIGURES

1 INTRODUCTION

The main aim of the project is to create a procedural FX tool kit to provide easily customizable character effects in Houdini. The implementation follows the animation pipeline by using animated rigs/objects and then exporting them in FBX into Houdini to apply and customize the needed effects. For testing purposes, all the rigged characters were obtained from *Mixamo*. Various styles of motion blur effects would be available to be configured and rendered on Houdini by using the HDA. The project is heavily dependent on Houdini's VEX language for point processing and attributes creation. Also Python for Houdini was briefly used.

To give the user better use of the tool, the inputs are split into two. First, the user can use the first HDA which is targeted for regular shaped objects that doesn't include skeleton. A second option would be to import a rigged character as FBX and plug it in as input for the second HDA. Depending on the input, the animations go through different computations that are passed down as attributes. To avoid creating these effects from scratch especially in projects that do not need very sophisticated effects, this tool can provide good options for artists to choose from and use for their projects. The toolkit includes some of the widely needed effects like dust/smoke trails or having trailing copies of the object to give cartoonish-like style of motion. As Houdini is used for this project, the HDA settings provide fairly good cutomization settings and properties for the procedural effect approach. In addition, the solvers that Houdini provides upgraded the provided effects and the outcome possibilities

1.1 PREVIOUS WORK

Over the past few years, stylized animations have been getting more appreciation from viewers. Especially with the release of "Spiderman:Into The Spiderverse"(2018) and the positive impact it had on people and the animation industry. The second part of the movie, "Spider-Man: Across the Spider-Verse"(2023), showed further technologies concerning the technical direction and animation field. Sony Pictures Animation had to decide on hand drawing the motion blur lines on the animated characters to give it the astounding look they wanted. This had a huge impact on the final look of the film. A seemingly simple addition like motion blur to the animations can have a fairly impressive impact on a film. Watching both films definitely were one of the main sources of

inspirations for this project.

The initial search for previous work and research papers was motivated by the ”twelve basic principles of animation” that were firstly introduced by Disney animators Ollie Johnston and Frank Thomas in their book ”The Illusion of Life: Disney Animation” released in 1981 (Thomas and Johnston 1981). The research focused on getting an overview on previous approaches on applying 2D hand drawn animation effects into 3D animation. The book and the principles had an influence on 2D animations for years. Recently, that influence has expanded to include 3D animations and stylizing 3D animations in the same way. Therefore a few of the effects available in the HDAs would be relating to that theme and approach.

One of the papers that tackled this point and was a direct reference to the approach of this project was a paper published to SIGGRAPH 2024 titled “SMEAR: Stylized Motion Exaggeration with ARt-direction“ (Basset et al. 2024). Basset discussed in this paper different algorithms about stylizing the motion of animated characters with different art directions. Three differents motion exaggeration were introduced in the paper including elongated in-betweens, trail lines and multiple in-betweens. Vertex manipulation was reviewed across the paper to calculate the motion offset of each vertex of the input object for accurate computations and smooth smears. The paper applies the concept on a plug-in for Blender to test all the concepts discussed. A fair share of the base of this project is adopted from this papers approach and mathematical equations for point manipulation. Although their application was on Blender, refactoring everything to work on Houdini was not as straight forward as it might seem.

1.2 TECHNICAL BACKGROUND

This section will discuss all the required scientific and technical information that will be used throughout the paper.

1.2.1 Catmull-Rom Splines

This concept will be used for interpolating point positions of a geometry. Catmull-Rom interpolation interpolates a spline curve to get the next point on a curve using the previous point and one consecutive to the point needed. So if the position of point p_1 is known and the position of point p_2 is needed, both points' p_0 and p_3 are used. It uses

the following Hermite matrix

$$M_{\text{Catmull-Rom}} = \frac{1}{2}\begin{bmatrix} -1 & 3 & -3 & 1 \\ 2 & -5 & 4 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix}$$

.

To get the position of a point along the curve the following rule is used:

$$\mathbf{P}(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \cdot \frac{1}{2}\begin{bmatrix} -1 & 3 & -3 & 1 \\ 2 & -5 & 4 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}$$

. The point position equation will be implemented into VEX code to interpolate the point positions which will be discussed later in this paper.

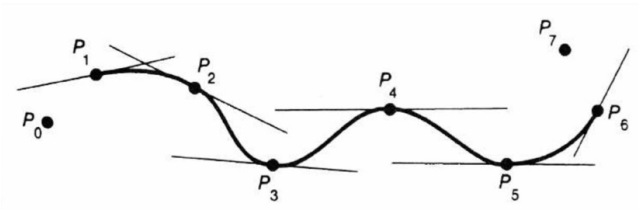

Figure 1.1: Figure with Catmull-Rom spline points (Ren, Bezier Curves Surfaces)

1.2.2 Smoothstep Function

As this function will be used a few times in the project a prior understanding is ideal. In his book about Mathematics for 3D Game Programming and Computer Graphics (3rd ed.), Lengyel introduces on of the cases of the cubic hermite interpolation called the smoothstep function(Lengyel, 2012). He uses the function for a smooth interpolation between two values over an interval. The interval being $[0,1]$ while ensuring that both the value and the first derivative are zero at the endpoints. This would result in a polynomial:

$$\text{smoothstep}(x) = \begin{cases} 0, & x \leq 0 \\ 3x^2 - 2x^3, & 0 < x < 1 \\ 1, & x \geq 1 \end{cases}$$

which is initially at 0 with a smooth raise till it reaches 1 and then a flat line after that at the value 1. Lengyel points out the use of the smoothstep function in computer graphics for blending, fading, and procedural texturing for its smooth transition properties(Lengyel, 2012). For the purpose of this project, it will be adopted to help with having smooth interpolation of points and point transformation for a smooth mesh.

1.2.3 Heaviside Function

The Heaviside function which is also often called unit-step function is often used for functions or applications that require a binary output(Kreyszig, 2006). Its notation is :

$$H(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases}$$

For the implementaion of this project it will be used to filter attribute values to fit certain criteria and provide a boolean 0/1 output accordingly.

2 PROJECT IMPLEMENTATION

As mentioned before that the solution is divided into two HDAs both for objects and rigged characters. The paper will discuss first how VEX was used in each solution to compute required attributes and then go through each effect in details.

2.1 POINTS MOTION OFFSET

To start developing the effects procedurally, each point on the input geometry will be assigned a motion offset delta. It is a value indicating the direction and magnitude of the motion of each point in the animation. The value is then adopted in subsequent applications in the tool.

2.1.1 Regular Objects

When it comes to a regularly shaped objects or geometry, the point are split in half. The first half is the trailing half, having a value for the motion offset δ: $-1.0 < \delta < 0.0$ depending on the velocity of the point at each frame. The second half would be the leading half with motion offset deltas equal $0.0 < \delta < 1.0$.

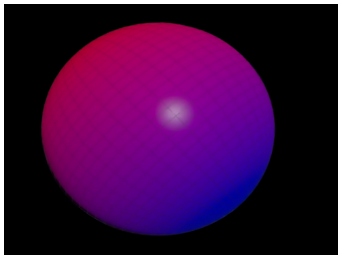

Figure 2.1: Visualization of the motion offset on an animated sphere

The node tree in Houdini starts by a ”Trail” node that is set to compute the velocity of each point on the input geometry. This node saves the velocity in a point attribute ”v” which is then fed into a Python node. In the Python node two numpy arrays are initialized for the position and velocity of every point. The centroid of the geometry is calculated using the mean of the positions of the points which is then used to get the relative position by subtracting the position from the centroid. Another numpy array is then filled with the normalized velocities. By using all the previously computed variables, the delta of every point is calculated by multiplying the normalized velocity

by the relative position. To normalize the deltas, every point's delta is divided by the maximum delta value. This equation was adopted from Basset's paper (2024) in the section where motion offsets of simple objects were discussed and the equation below was introduced.

$$\delta_i = \frac{\delta_i}{\max_j |\delta_j|}, \quad \text{with} \quad \delta_i = (\mathbf{p}_i - \mathbf{c}) \cdot \hat{\mathbf{v}}$$

The delta offset is saved as a point attribute which is then passed to the effects implementation which will be discussed in later sections.

2.1.2 Rigged Characters

Rigged characters were the most challenging part of the project as it was completely original and it did not follow any prior implementation. Following the same approach or adopting the design of the previous work mad in Blender did not apply for this part as Houdini deals with bones in a different way. When a FBX file is imported in Houdini using the "FBX Character Import" node it automatically turns it into a KineFX rig. That was one of the main reasons the project's pipeline was based on using FBX files. There are three outputs for this node: Capture Pose, Animated Pose, Rest Geometry. Each output went through a few pre-computations before the major VEX node.

Basset's implementation concerning bones was specific and sophisticated, therefore some modifications had to be done to be applicable to Houdini. In his implementation the bones were first split just like the normal object but using *ribbons*.

The animated pose which is the the skeleton with the input animation applied to it. It was passed through a primitive wrangle to extract the required attributes that will be used later on. Each bone entry in the skeleton (primitive) was assigned a root position c_r and tip position c_t and an axis vector b from the root to the tip. Another input is used by the primitive wrangle which comes from a Trail node that calculates the velocity of each point, and so the velocity of the root v_r and tip v_t point are mapped to the corresponding bone.

The animation was applied to the rest geometry using a Bone Deform node and got it's capture attributes unpacked. When the capture attributes are unpacked they turn into two arrays with bone weight data. As each bone from the unpacked capture attributes and the capture pose (skeleton) have different primitive indexing a mapping needed to be done for accurate computation. Using attribute wrangle for VEX programming

the correct index of each bone was raised as an attribute by assigning each point on the geometry its assigned bone or the bone with the highest weight. Now that all the pre-required attributes are calculated a final Point wrangle is used to manipulate each point of the geometry. Using mathematical equations from Basset's paper the wrangle computes a motion coefficient u by:

$$u_i = S\left(\frac{(p_i - c_r)\cdot \hat{b}}{\|b\|}\right)$$

while S being the smoothstep function and p_i being the point location. It is then used to get the direction of motion using

$$\hat{v}(u) = \frac{\sin((1-u)\omega)}{\sin(\omega)}\hat{v}_r + \frac{\sin(u\omega)}{\sin(\omega)}\hat{v}_t$$

with ω denoting $arcos(\hat{v}_r \cdot \hat{v}_t)$ only if ω is > 0.1, else it would calculated using linear interpolation. The next step is to calculate the normals of the ribbon n splitting the bone:

$$n(u) = \hat{v}(u) - \left(\hat{v}(u)\cdot\hat{b}\right)\hat{b}, \quad \hat{n}(u) = \frac{n(u)}{\|n(u)\|}$$

All the previous equations build up to get the motion offset of each point that we need to calculate using :

$$\delta_i = (p_i - c_r)\cdot\hat{n}(u_i)$$

After raising the motion offsets as a point attribute, each bone needed to be assigned a maximum value of its points' motion offset for normalization purposes. In a separate VEX wrangle that looped over the bones and checked all the points motion offsets to assign its maximum value. In the final VEX wrangle of the implementation, it computes the normalized motion offset $\bar{\delta}$ by

$$\bar{\delta}_i = w_{\text{coll.}}(u_i)\cdot\frac{\delta_i}{\max_j|\delta_j|}$$

according to the paper, the w_{coll} is a weight function that is chosen to control the magnitude of the motion offsets by decreasing it and is computed as $w_{coll}(u) = 1 - (\hat{v}(u)\cdot\hat{b})^2$. As the skeleton is turned into a KineFX rig and it has the capture attributes unpacked, it provies two arrays per point showing each index of the bones affecting the

point and its corresponding weight in the next array. The node uses these two arrays to calculate how the weight of each bone contributes to the final motion offset by:

$$\bar{\delta}_i = \sum_k w_{ik}\,\bar{\delta}_k$$

Most of the procedural effects that are implemented in this project will depend on the motion offsets calculated both for regular objects and skeleton that is why it was somehow the base of the project.

2.1.3 HDA Settings

The HDA's have general settings that allows the users to have more control over the output of the tools. These settings are as follows. A frame range for the effect to take place, as an artist may choose to have a specific effect at a specific duration of the animation. The output can be either an effect only to be rendered alone or to be accompanied with the input animated geometry. The user has the choice of multiple effects to be applied (will be discussed throughout the rest of the paper). The whole UI was based on the available HDA parameters and UI settings that Houdini provide.

Figure 2.2: HDA settings for the tools.

2.2 MOTION DEPENDENT EFFECTS

For the effects, the choice was mainly dependent on widely used or needed examples in VFX. The type of effects were divided into two major groups. The first one being the three effects provided in the research paper. Other effects are particle effects using houdini's solvers.

2.2.1 Trail Copies

The first effect that utilises the motion offset value assigned to each point is the trail copies. This effect simply creates copies of the animated geometry behind and infront

the original position of the geometry. As the motion offset splits the geometry into two parts, the first half being the trailing part of the geometry and the other being the leading part. That feature help in also splitting the copies to make the fazed motion blur effect.

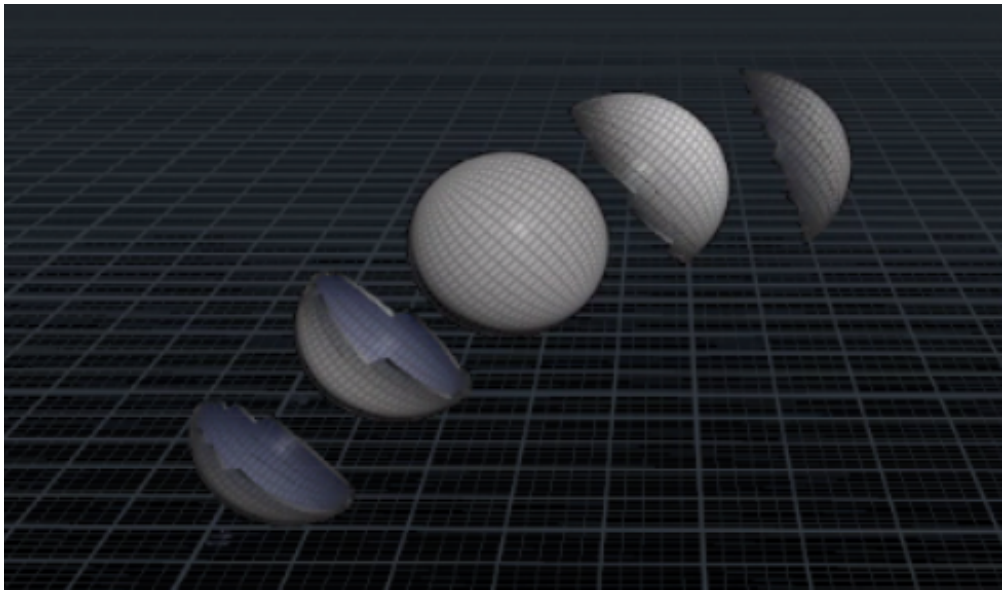

Figure 2.3: The figure showing the trail copies effect by making two copies in each direction.

A for loop nodes are established and runs depending on the number of copies needed by the user. In each iteration, a Time Shift node shifts the animation by the number of frames the user indicated and calculates the opacity depending on the motion offset, gradient value and the opacity multiplier. The final alpha value of each point goes through various computations to get its final value to control the opacity. The intial value of the alpha follows the equation provided by Basset in his research paper by:

$$S_M(\delta, m; m_{\max}) = H(\delta_m)\, S\left(\frac{|\delta| - \delta_m}{1 - \delta_m}\right), \quad \text{with} \quad \delta_m = \frac{|m| - 1}{m_{\max}}$$

H and S are the heaviside and and the smoothstep functions, m is the copy number and m_{max} is the max number of copies while δ is the motion offset. So the opacity of every point of every copy is calculated and then assigned to the alpha channel. This project introduces more flexibility and options including

- Number of copies
- Number of frame difference between each copy
- The direction of the copies, either trailing or leading or both

- The opacity of the copies
- Choose whether they want an increasing or decreasing gradient of opacity in the copies

The skeleton HDA expands the options to applying the copied parts to specific bones chosen by the user. As an example figure made copies of only the right leg and feet related bones with an increasing gradient of 3 trailing copies

Figure 2.4: The figure showing the trail copies effect on a character by making three leading copies.

2.2.2 Smear

The elongated smear effect also depends on the motion offset delta to apply point computations to the geometry. As discussed in section 1.2.1 the position of each point on the geometry is calculated using Catmull-Rom interpolation. The input geometry is time shifted three time getting frames $f-1$, $f+1$ and lastly $f+2$. All these frames along with the current frame are put in as input for an attribute wrangle node. Each point manipulated by the equation:

$$p_i'(f) = Q_{i,f(\beta)}\big(t(\beta)\big)$$

while

$$f(\beta) = f + \lfloor \beta \rfloor, \quad t(\beta) = \beta - \lfloor \beta \rfloor.$$

the $Q_{i,f(\beta)}$ is how each point i is trajected between frames f and $f + 1$, with $t \in [0, 1]$ (Basset et al 2024). First the constant input from the user (controls the smear size) is multiplied by the motion offset which is the β value. The floor of that multiple is then subtracted from the β to get the t. As Catmull-Rom interpolation was introduced in section 1.2.1 and the motion offsets coefficient in section 2.1, they were both used in a point wrangle to compute the transformation of each point. Therefore the t value is then used for interpolation using Catmull-Rom.

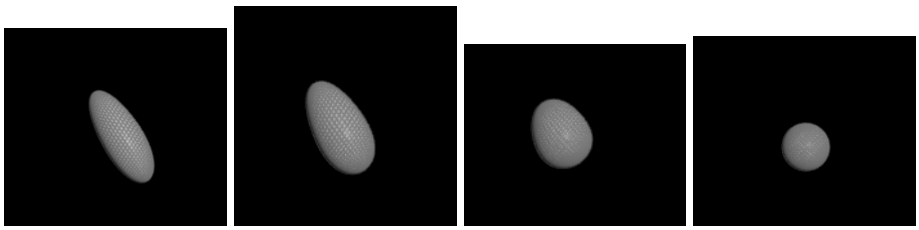

Figure 2.5: Four consecutive frames from a smeared sphere.

The skeleton approach for this effect has an extra feature of selecting the bones to be smeared to give more freedom for artists.

Figure 2.6: The figure showing a smeared character with the legs bones selected.

2.2.3 Trail Lines

This effect is basically lines produced from random point on the geometry that follows the path of the animation to give it the 2D animation look.

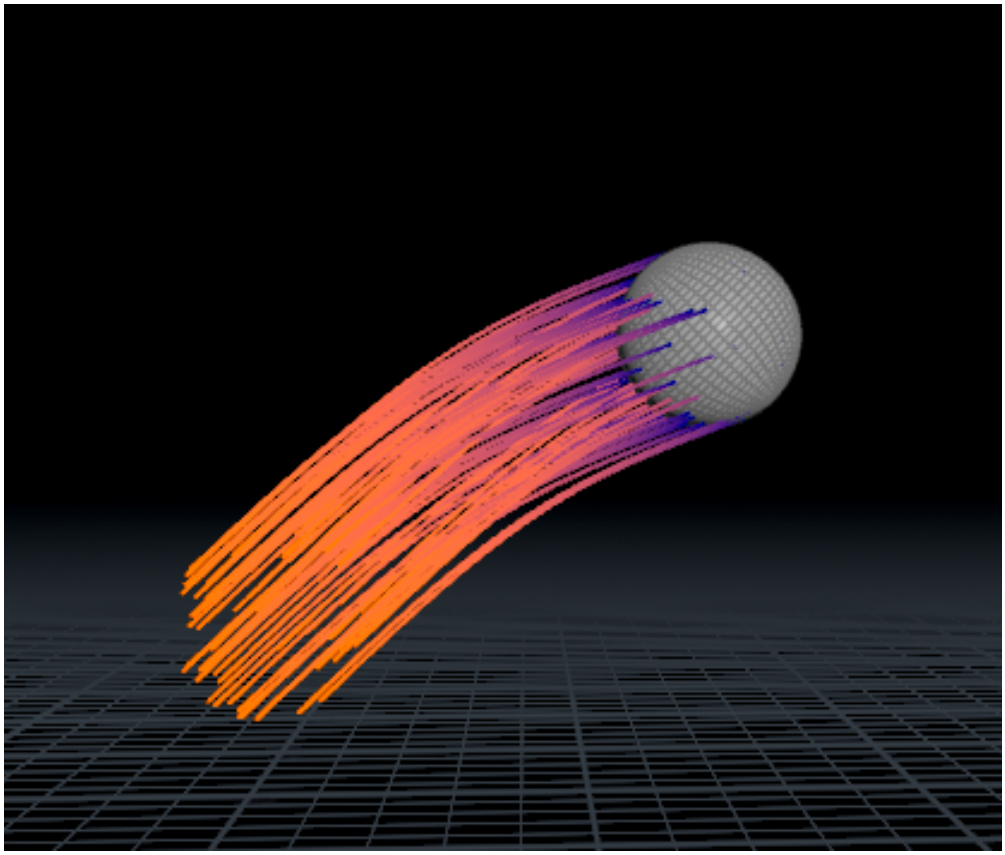

Figure 2.7: The figure showing trail lines for a sphere.

Unlike Basset's approach in the paper, this project rely on Houdini's range of node to implement this effect. The Trail node provides the required criteria to achieve the look in figure 2.4. The user has the following options to customize the effect

- Trail length
- Lines frequency
- Colour for the lines
- The direction for the lines whether its trailing, leading or both

As the trail lines can be trailing or leading the motion offsets were reused to group points on the geometry accordingly. The points were grouped and then scattered to get random positioning of the lines. As the user chooses the length of the line, the animation is frame shifted accordingly to produce the trail lines. Two color approaches are available, one being a chosen base colour or gradient. The other one would be inheriting the material of the geometry itself. For regular objects, the user can choose a

texture file for the base colour of the object for the lines to inherit. On the other hand, the characters HDA use the texture embedded in the FBX file.

Figure 2.8: The figure showing trail lines that is extended in both directions of motion.

2.3 PARTICLE EFFECTS

As mentioned above, the toolkit's effect were expanded to include interesting effects using Houdini's powerful solvers. These effects will achieve one of the projects main aims and target audience by being a quick plugin to create procedural effects. Most of the ideas and initial look of the effects follows tutorials and learning paths from the SideFX website, however the effort made was mainly applying the procedural setting and modifications for the HDA. The time exerted was to turn the static effects or particle system into a procedural tool that can have the same look on every input possible, while being heavily customizable. The particle effects have a common parameter in the HDA which is the particles birth setting. The user can choose from either have the frame range specified to be for the particles birth in general (the particles will only start with the frame start). The other option that the particles birth starts from the timelines beginning but will only be visible starting the specified frame start.

2.3.1 Dust

To integrate the tool with Houdini's solver, this effect simulates a dust trail effect using POP networks and Pyrosolvers. The dust gives the exaggerated effect of motion for objects and characters. The original dust look and settings are following one of the Houdini tutorials (FX Guru 2024). Unlike the tutorial, the tool expands the effect to be procedural and suits the users input geometry and settings. At first, particles are generated from the input geometry using POP network while adding suitable noise and velocity. As the amount of particles birthed affects the final outcome, the user has the ability to control it through the HDA. The particles are then turned into a Pyrosource and a volume with the required attributes to be then fed into the Pyro solver. The HDA has a default dust color, however the user has the ability to change the color. The available settings is sufficient to explore smoke/dust options possible with Houdini and pyro solvers. Adjusting colors, particle count, particle velocity and life expectancy gives infinite amount of possibilities for this effect.

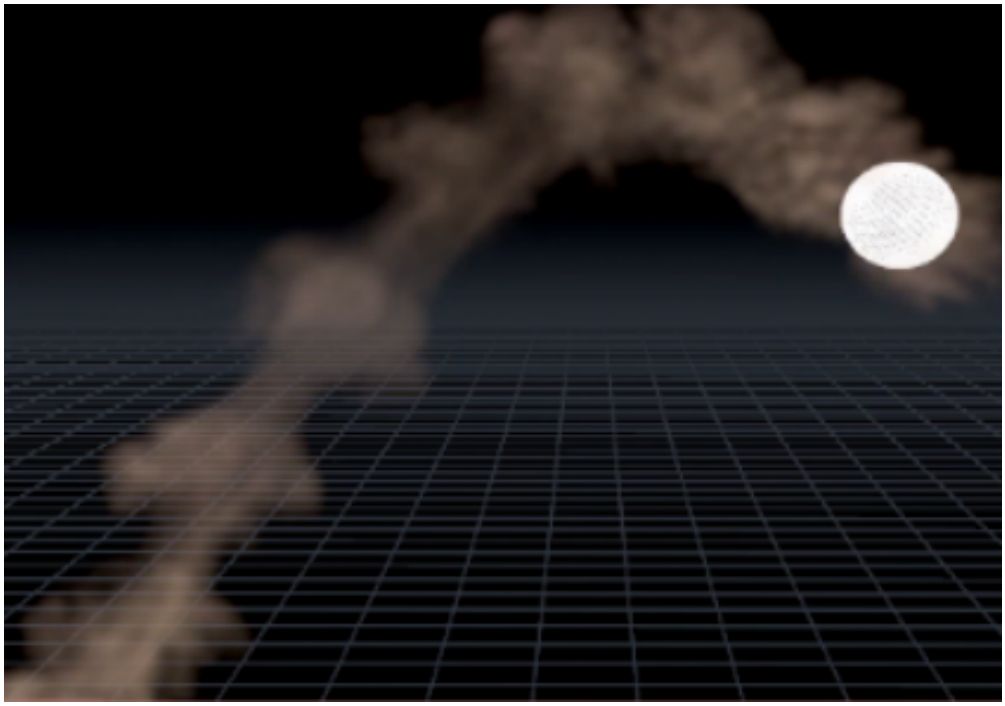

Figure 2.9: The figure showing the dust effect on a regular object.

Figure 2.10: The figure showing the dust effect when selected to be applied on the sword.

2.3.2 Energy Particles

To extend the usability of the tool, an effect depending on POP networks and particles was needed. Houdini's particle solvers has various settings for infinite looks of the particles flowing. This project follows the initial implementation of a shockwave energy effect (Indian VFX School 2022). As the work done previously was a static shockwave, to suit this project it had to be changed. Instead of sourcing the particles from a static object, it was changed to work with the input animation procedurally. Some attributes were raised to be changed by the HDA like the particle count and color. The HDA includes the predefined material for the particles which can then be changed by the user. To render this effect the HDA has an embedded material network with a material for this effect. The user needs to assign this material to the ”geometry” node or in the render settings.

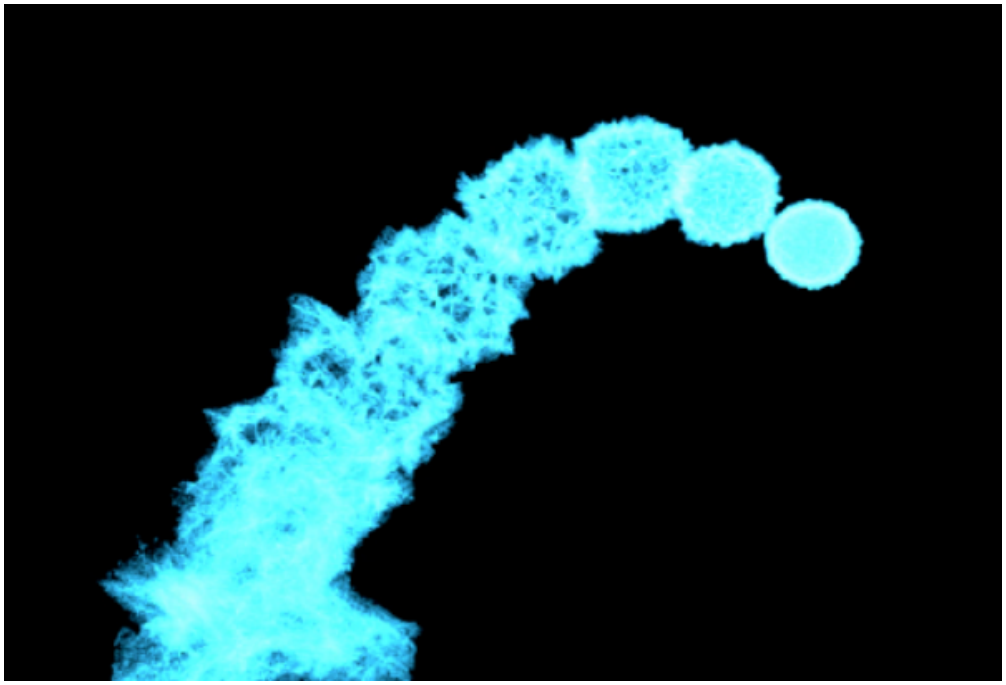

Figure 2.11: The figure showing the energy particles effect.

Figure 2.12: The figure showing energy particles from the hand bones of a character.

CONCLUSION

To wrap up the project, the base algorithms used and developed were influenced and adapted from the previous work in the paper "SMEAR: Stylized Motion Exaggeration with ARt-direction" (Basset et al. 2024). The main work of this paper was to adapt the algorithms and maths behind the vertex manipulation into Houdini. On top of that base was the research and development to create the effects introduced with the Houdini twist to it. Trials for these effects were to get better outcomes with Houdini's render engines. Moreover, the procedural effects expanded furtherly the tool. Finally, the HDA settings set-up was completely based on testing and previous experience taking into consideration the artists needs.

After mentioning the implementation details of the project it best to mention the possible future work, limitations and improvements.To start with the limitations, The major limitation for this tool that it has limited number of effects which cannot be applied to majority or a big number of projects or user. However, it provides somehow usable effects while showing a proof of concept. The project's research and development would have been upgraded with the help of animators and FX artists in the industry. Getting feedback and suggestion would have a huge impact and improvement in the UI specially with the HDA settings and attributes.

Building a procedural tool can be expanded in many ways to suit various project requirements. For this specific project, a basic expansion would be first to develop and include more effects. Including more simulations with RBD/Bullet solvers would be an interesting options. Houdini provides wide range of nodes that affect the usability of the HDA. To have a more complete tool, the HDA or tool can also include rendering nodes or techniques to match the theme or artistic approach of the effects available. To isolate the motion offset calculation as a feature, a seperate HDA may provide the ability for the user to input the effect or particle network as input. For a smoother pipeline in development, the tool can be expanded to have live feed or real-time changes with Unreal Engine 5. Similarly the same feature can be integrate with animation in Maya and seeing instant feedback on how the animation would look concurrently with the effect.

In conclusion, the tool and HDAs serve their initial purpose of making an MVP of a procedural effects toolkit for stylized motion blur while tackling different aspects

of technical direction such as mathematics/physics, particle simulations and animation pipeline.

REFERENCES

[1] Basset, J., Bénard, P. and Barla, P., 2024. SMEAR: Stylized Motion Exaggeration with ARt-direction. In: Burbano, A., Zorin, D. and Jarosz, W., eds. ACM SIGGRAPH 2024 Conference Papers, Denver, CO, USA, 27 July–1 August 2024, New York, NY: ACM, pp.107. [online] Available from: https://doi.org/10.1145/3641519.3657457.

[2] SideFX, 2024. Dust Interaction with Rhino Animation. [online] Available from: *https://www.sidefx.com/tutorials/dust-interaction-with-rhino-animation/*

[3] S.-T. and Chi, M.-T., 2024. 3D-to-2D Animation Smear Effect Technique Based on Japanese Hand-Drawn Animation Style. [online] Conference paper, 3 December 2024. Available from: DOI:10.1145/3681756.3697891.

[4] Drury, M. R., 2016. Creating 3D Smear Frames for Animation. [online] Undergraduate honors thesis (East Tennessee State University). Available from: https://dc.etsu.edu/honors/348/

[5] DeRose, T. D. and Barsky, B. A., 1988. Geometric Continuity, Shape Parameters, and Geometric Constructions for Catmull-Rom Splines. ACM Transactions on Graphics, 7(1), pp. 1–41. [online] Available from: graphics.pixar.com/people/derose/publications/GcCatmullRom/paper.pdf

[6] Entagma, 2021. No VEX Houdini Tutorial: Dancing Mocap Trails. [online video] Available from: https://www.youtube.com/watch?v=U0GXGtdq1Zg

[7] Entagma, 2021. Guest Tutorial: Simon Fiedler – Controlling Swirly Particles. [online video] Available from: https://www.youtube.com/watch?v=yqM_3goH4J8&t=1096s

[8] Hughes, J. F., van Dam, A., McGuire, M., Sklar, D. F., Foley, J. D., Feiner, S. K. and Akeley, K., 2014. Chapter 22: Splines and Subdivision. In: J. F. Hughes, A. van Dam, M. McGuire, D. F. Sklar, J. D. Foley, S. K. Feiner and K. Akeley, eds. 2014. Computer Graphics: Principles and Practice. 3rd ed. Boston: Addison-Wesley, pp. 1033–1120.

[9] Indian VFX School, 2022. *Houdini - Particles Energy Shockwave Tutorial*. YouTube. Available at: https://www.youtube.com/watch?v=mQemf9WctFI.

[10] Chang, J. Curves and Surfaces. CGI Techniques. Bournemouth University.

[11] NG, Ren. Bezier Curves Surfaces. CS184/284A. University of California, Berkeley.

[12] Kreyszig, E. (2006) Advanced engineering mathematics. 9th edn. Hoboken, NJ: John Wiley Sons.

[13] Lengyel, E. (2012) Mathematics for 3D game programming and computer graphics. 3rd edn. Boston, MA: Cengage Learning.

[14] Thomas, F. and Johnston, O., 1981. The Illusion of Life: Disney Animation. New York: Abbeville Press.