CONTROLLING MELTING PHASE CHANGE SIMULATIONS

A Thesis

by

RUSHIL SHASHANK KEKRE

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

December 2017

Major Subject: Visualization

ABSTRACT

I have developed a set of Houdini Digital Assets (HDA) to control phase change in materials that melt. Examples of real world materials that exhibit these phenomena include melting candles, molten steel, etc. The purpose of this tool is to provide artistic user control when animating these materials. The user can provide an object model as an input to the HDA, which can then be split into multiple sections as per the user's needs. The user can then specify attributes such as temperature, viscosity, and a melting rate for each section. The object is then filled with particles to resemble a particle based fluid object, with each particle inheriting the attributes of the section it belongs to. When the simulation is run, two conditions control the behavior of the phase change at each timestep. First, the particles melt only at the rate specified for their section. Second, the particles from one section do not mix with those from another section. These conditions are implemented using custom digital assets in Houdini that I developed. Once the simulation is complete, the user is able to combine the deformed meshes of each section into a unified animated mesh and proceed with shading, lighting, and rendering.

DEDICATION

To my parents, without whom none of my successes would be possible. Their love, support, and belief in my wild and unrealistic dreams is the reason I am here today.

ACKNOWLEDGMENTS

I would like to thank my committee chair, Dr. Frederic Parke, for his constant support, patience, advice, and timely feedback in completing this thesis. I would also like to thank my committee members, Dr. John Keyser and Philip Galanter, for helping me understand key concepts and providing valuable insights into my research topic. Without the guidance of my committee, this work would not have been possible.

I would like to express my gratitude to Ryan Goldade and Luca Pataracchia of SideFX Software for their help in dealing with implementation issues, despite their busy schedules.

Finally, thank you to all my friends, old and new, who have shared in my highs and lows. Their support and encouragement has meant a lot.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a thesis committee consisting of Dr. Frederic Parke (chair) and Philip Galanter of the Department of Visualization and Dr. John Keyser of the Department of Computer Science.

All other work conducted for the thesis was completed independently by the student, with edits made to the paper by Dr. Frederic Parke.

Funding Sources

There are no funding sources to acknowledge related to the research and completion of this work.

NOMENCLATURE

3D	Three-Dimensional
CFD	Computational Fluid Dynamics
DOP	Dynamics Operators
FLIP	Fluid Implicit Particle
FX	Effects
HDA	Houdini Digital Asset
POP	Particle Operators
ROP	Render Operators
SOP	Surface Operators
SPH	Smoothed Particle Hydrodynamics
UI	User Interface
VEX	Vector Expression
VOP	VEX Operators

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Fluid animation has come a long way from its debut in the 1998 film *Antz*, which featured computer generated water, to the visual effects fueled movies of recent years such as *Life of Pi* (2012) and *The Hobbit: Desolation of Smaug* (2013). Over the decades, there have been many technological advances in rendering techniques, better simulation methods, and more robust 3D software, which have resulted in highly realistic simulation and rendering of fluids. Many fluid behaviors can be accurately modeled today. For example, *The Hobbit: Desolation of Smaug* (2013) features a scene where a giant gold statue melts, drowning the movie's main antagonist Smaug in a flood of molten gold, as seen in Figure 1.1. Disney's *Moana* (2016), also features extensive fluid simulations such as oceans and lava, as seen in Figure 1.2.

Figure 1.1: A gold statue of a dwarf king from The Hobbit: Desolation of Smaug (2013) before it melts

The area of fluid behavior I have focused on is *phase change*. Phase change, otherwise known as *phase transition*, is most commonly used to describe transitions between solid, liquid and gaseous states of matter. In this work, the term *fluid* is used to encompass the motion of liquids (water) and not gases (smoke, fire). The state change that I have focused on is *melting* (gold, chocolate, wax, etc.). I have developed a set of Houdini Digital Assets (HDA) that will help effects artists manipulate these changes in state, thus providing user control when animating objects that exhibit melting phenomena. In my work, a Houdini digital asset is also called an asset or a tool, for short. For this asset, I treat objects as particle fluids with variable viscosity [1] for the duration of the simulation.

Figure 1.2: Lava simulation from Disney's Moana (2016)

2. BACKGROUND AND PREVIOUS WORK

Fluid simulation has been handled in many ways in computer graphics literature and in Computational Fluid Dynamics (CFD) for engineering purposes. Abbott and Basco cover some basic principles without being too mathematically dense [2]. Phase transition represents a complex subset of CFD.

The basis for my tool is similar to the approach taken by Carlson et al. based on the incompressible viscous Navier-Stokes equations. By considering objects that undergo this phenomenon to initially be a fluid with high viscosity, a change in phase is achieved by altering the viscosity of the fluid in accordance with factors such as temperature and water content. A benefit of this method, the authors state, is the elimination of arbitrary threshold conditions needed to determine when the object should be treated as a solid or a liquid [1]. This approach is carried out at a particle level.

Miller and Pearce handle viscous fluids by treating each particle as a globule in a connected particle system. Each globule has characteristics such as mass, temperature, radius, position and velocity. By simulating soft collisions between these globules, the authors have eliminated spatial stacking issues that can arise from rigid collisions. The movement of the globules is a coarse approximation of molecular movement within the liquid. This connected particle system deforms over time which leads to a dynamically changing geometry [3]. The approach is based on the concept of flocks and their behavior as outlined by Reynolds [4] where each particle interacts with neighboring particles within a radius.

Terzopoulos et al. approach the simulation of melting deformable solids by modeling thermoelastic objects that conduct heat and exhibit thermodynamic phenomena. These thermoelastic simulations are based on the Lagrange equations of nonrigid motion coupled with a heat equation to describe the diffusion of heat within the objects. During the simulation, when objects come into contact with surfaces of higher temperatures, they conduct heat into their interiors. As the temperature increases, the objects deform, and as the temperature exceeds the melting point, the objects melt into liquids. A molecular dynamics approach, where each particle represents a molecule, is taken to simulate the liquid state, as pairs of particles interact with each other by exhibiting attractive and repulsive forces based on their distance from each other [5].

Based on this thermodynamic approach at a molecular level, Tonnesen constructed a single temperature dependent model to model changes in geometry between solids and liquids [6]. Depending on the temperature, the objects exhibit fluid-like behavior with changing geometry. Cooler objects are more rigid, thus exhibiting characteristics found in solids, whereas hot objects deform as the temperature increases.

Desbrun and Gascuel adapted the Smoothed Particle Hydrodynamics (SPH) approach, used by physicists, to simulate fluids in computer graphics [7]. By defining matter as a collection of sample points, or particles, in a restricted space, it was possible for the authors to carry out computations of physical quantities such as pressure, density, temperature, velocity, etc. in a stable manner. While traditional techniques used an Eulerian approach - which involved dividing space into a fixed grid of voxels and tracking the motion of the fluid in and out of the voxel - the authors considered these to be counterintuitive for flows. SPH relies on a Lagrangian approach, which involves tracking the fluid elements over space and time. It was also possible for the authors to define surfaces using a level set of the mass density function which indicate where and how mass is distributed in space.

Using the SPH concept, Stora et al. animated lava flows by varying viscosity with changes in temperature [8]. As the temperature of the lava cools, its viscosity increases. By linking the temperature to the viscosity for each particle and simulating heat transfers, the authors could generate realistic motion and rendering of the lava. During the simulation, heat transfers at the surface of the lava and the interior particles, resulted in the temperature decreasing, thus enabling a transition from a low viscous behavior to a highly viscous one.

Foster and Metaxas provide an approach to animate liquids to make them look realistic. It is based on the Navier-Stokes equations which couple momentum with mass conservation to completely describe fluid motion [9]. The authors achieve realistic motion through a finite difference approximation to the incompressible Navier-Stokes equations. This results in the creation of a pressure and velocity profile for the simulation environment. This profile is then used to determine the behavior of surfaces. The simulation region is divided into a Cartesian coordinate grid and at the center of each grid cell the velocities along all three axes as well as pressure are defined. The authors create the surface profile by assuming the walls of the object are always aligned with the grid cells. When the fluid particles pass through these cells, boundary velocity and pressure are used. The fluid surface is determined using a height field. This approach provided a physically accurate and stable method to develop a general fluid control tool for animators. The authors claim this allows animators to specify and control three-dimensional fluid flow without having to think about the underlying representation [10]. The authors also claim that by solving the three-dimensional Navier-Stokes equation over a voxel representation of the surrounding environment, boundary conditions could be calculated and applied much faster. This in turn makes the simulation faster compared to traditional methods, while preserving the accurate behavior of the Navier-Stokes equations.

Stam developed a modified approach to solve for fluid flow by replacing the finite difference scheme with a Lagrangian method [11]. While this method works well for fluids such as gas or smoke, it isn't a good model for simulating liquid flow as it results in mass dissipation.

Foster and Fedkiw improved this method by using a combination of inertia-less particles and level sets to track the motion of a liquid surface. The level set prevents mass dissipation while the particles allow splashing [12]. The authors also developed a technique to account for the movement of polygonal objects within the liquid. By focusing on modeling the liquid volume, the authors could achieve more realistic simulation and rendering of fluid surfaces.

Based on Stam's work, Enright, Marschner, and Fedkiw proposed a technique that relied on surface modeling instead of volume modeling [13]. By extrapolating surface velocities into the nearby regions occupied by air, the authors could generate more accurate results that were visually appealing and physically plausible. In doing so, the authors could add a degree of control to the behavior of the fluid surface such as dampening effects to reduce or increase splashing.

Goktekin, Bargteil, and O'Brien further extended this approach to demonstrate the behavior of viscoelastic fluids [14]. By introducing a new term to the Navier-Stokes equations to handle elastic strain, the authors were able to model viscoelastic fluids that would deform when subjected to external forces.

Rasmussen et. al. developed an efficient method for the directable animation of photo-realistic liquids in a visual effects production [15]. By coupling the particle level set technique of Enright, Marschner, Fedkiw with the control methodology introduced by Foster and Metaxas, they created a system that allowed directable control of fluids. Control particles are used to provide the desired degree of directed behavior. Associated with each control particle is a control shape which defines the region of influence of the particle, and within this region a falloff curve is used to determine the amount of control applied to the liquid. A *soft*, or blended, control mixes between the animators desired value and the value currently being used by the simulation. This control can be applied to any controllable liquid property, such as viscosity. A precise degree of control, known as a *hard* control, is used for values such as velocity that remain constant within the region of influence of the control particle, for the directed movement and visual appearance of the liquid surface.

3. DEVELOPMENT PLATFORM

3.1 Houdini

Houdini is a 3D animation software developed by SideFX Software [16]. It is an industry standard for FX development. It allows artists to construct node networks using operators to create a wide range of simulations including fluids, cloth, rigid bodies, and soft bodies. The variety of operators within Houdini allow artists to focus on the creative aspects of production rather than the technical aspects. Houdini has been used in various feature films, recently in Disney's *Zootopia* (2016), Pixar's *Finding Dory* (2016), and Disney's *Moana* (2016).

3.2 Houdini Operators

Houdini's tools are mostly implemented as operators. There are several benefits to this approach. It allows highly detailed objects to be constructed with very few steps, it encourages a non-linear workflow, and it allows the creation of new operators by combining existing operators [16]. This operator based structure is divided into several main groups:

1. OBJs - These are nodes that pass transform information, certain material and render settings, etc. and are traditionally composed of one or more SOPs. A collection of such nodes is known as an Object network, otherwise known as the "Scene level".

2. SOPs (Surface Operators) - can be used to deal with geometric structure by influencing surfaces, vertices, points, normals, etc. and provide capabilities to deal with static volumes, object velocities, etc. These operators play a big role in procedural modeling.

3. POPs (Particle Operators) - used to manipulate particle systems by providing functionality to introduce, or control, forces, noise, collisions, etc.

4. DOPs (Dynamic Operators) - for dynamic simulation of fluids, cloth, rigid body interaction, etc. A collection of DOPs forms a DOP simulation, which in turn provides controls such as caching options, timestep controls, etc.

5. CHOPs (Channel Operators) - for manipulating time-based channel data such as animation curves or audio information.

6. SHOPs (Shading Operators) - can be used to create shading networks to represent different shaders.

7. COPs (Composite operators) - used to manipulate 2D pixel data. Like other mainstream compositing packages, there is functionality to composite images such as render passes, depth maps, etc.

8. ROPs (Render Operators) - for building networks to represent different render passes and render dependencies. Different nodes can also represent different render jobs or launch different renderers.

9. VOPs - VEX operators - for building nodes of any of the above types using a graphical representation of VEX. (See Section 3.4)

3.3 Houdini Digital Asset (HDA)

A Houdini Digital Asset (HDA) is a collection of connected operators that perform specific functions packaged into a single node. A HDA is a reusable tool which can be used by artists in various scenes and simultaneously across various workstations in the Houdini environment. The interface for HDA's can also be customized by different artists as per their needs. HDA's are especially useful to FX Technical Directors since they can create complex operator networks and condense them into single nodes to be used by animators.

3.4 VEX

Vector Expression (VEX) is Houdini's internal language. It bears similarities to Renderman Shading Language (RSL) and its language syntax is C-like [16]. Using VEX, users can develop custom SOPs, POPs, shaders, etc.

3.5 Mantra

Mantra is the renderer bundled with Houdini. It has many similarities to Photorealistic Renderman in its scope and application. Shaders are scriptable and are composed using VEX, or by using VOPs. VOPs are a node based approach to VEX, and are similar to the visual scripting workflows found in other 3D packages and game engines.

4. GOALS AND OBJECTIVES

The main goal of this research was to create a set of Houdini Digital Assets to provide user control when performing melting simulations. These assets accomplish the following:

1. Provide a method for users to create sections from the object they intend to simulate, and to provide user control of section attributes.
2. Develop a method to group particles in accordance with the user defined sections and to inherit the section attributes.
3. Develop custom operators in Houdini to ensure that particles are simulated in accordance with their section attributes.
4. Provide a controlled animated deforming mesh of the melting object as the result of the simulation.

5. METHODOLOGY

The methodology of this research is to develop a Houdini Digital Asset that demonstrates the proposed control approach to simulating and animating melting phase change of 3D objects. The focus has been on creating a prototype using Houdini that allows the user to create a melting simulation and animation in a controlled manner.

6. SIMULATION APPROACH

A two-step approach is employed to achieve controllable melting in an object model. The first step involves preparing the model for simulation and assigning section attributes. The second step involves specifying simulation parameters for the heating and cooling of the fluid. The workflow for achieving controllable phase change is shown in Figure 6.1.

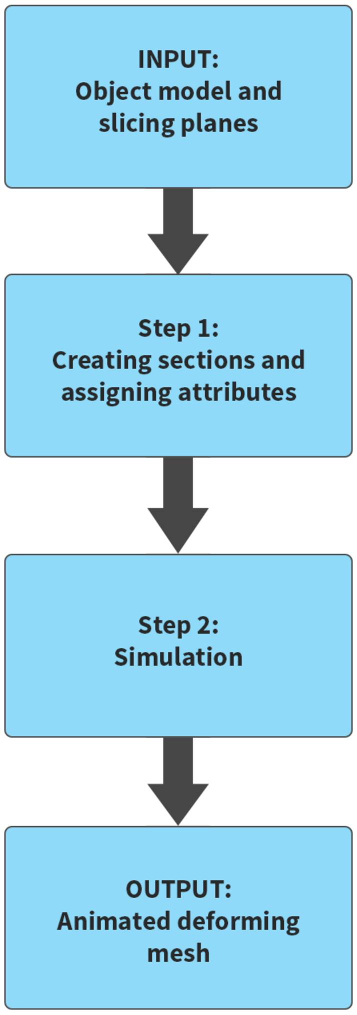

INPUT:
Object model and slicing planes
Step 1:
Creating sections and assigning attributes
Step 2:
Simulation
OUTPUT:
Animated deforming mesh

Figure 6.1: Workflow for achieving controllable phase change

6.1 Model Preparation

There are many ways in which an object model may be split into sections, such as Voronoi fracturing, using noise patterns, painting parts of the model, etc. The method I chose to create sections is by *slicing*. This eliminates the randomness of Voronoi and noise patterns and allows the user to slice the model accurately as it offers more control. A user can position multiple planes to create boundaries for each desired section of the model.

Once the model has been sliced into sections, the user can specify section attribute values such as temperature and melting rate. If the user chooses not to specify attribute values for a particular section, default values will be used. The default value for viscosity is set to 5000. This value was determined by many test simulations. This allows the section to melt quickly when subjected to heat. If a section is not subjected to heat, it maintains its shape. The default values for temperature and melting rate are set to zero, to ensure that there is no heat transfer within the section.

Figure 6.2 shows the general workflow of this stage.

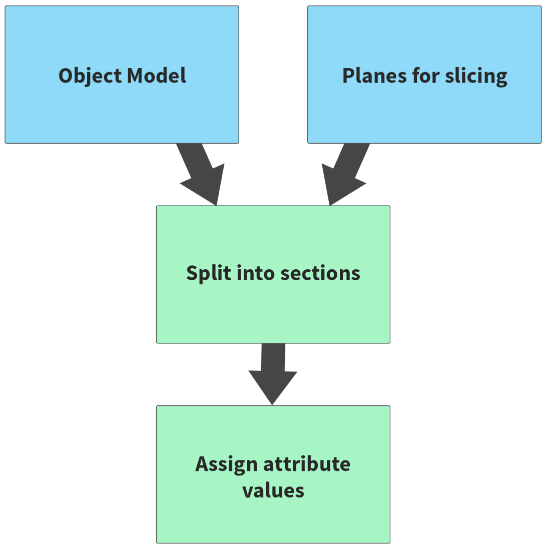


Figure 6.2: Model preparation stages

6.2 Simulation

The simulation is run using the attributes from the model preparation stage. At this stage, the user can specify attributes such as threshold temperature, cooling rate, minimum-maximum viscosity, and control temperature flow through the fluid. At each timestep, the HDA ensures that particles are simulated in accordance with their section values only.

This HDA is connected to Houdini's in-built FLIP fluid solver which is the basis of the simulation. This is shown in Figure 6.3

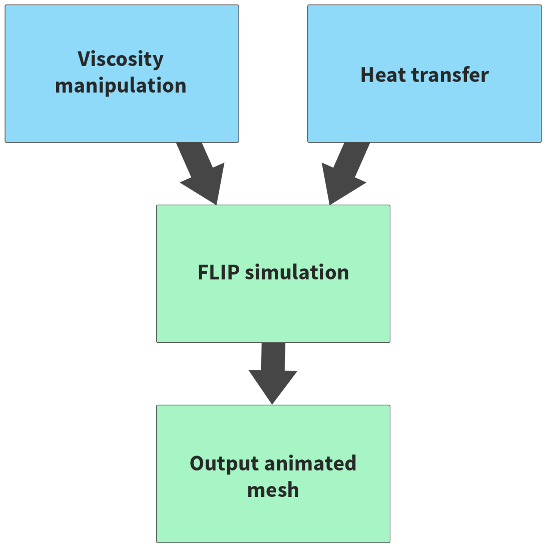


Figure 6.3: Simulation approach

6.2.1 FLIP Solver

The FLIP solver is a DOP that allows the user to control liquid simulations. It is a hybrid between particle based and volume based fluid simulation. The advantage of the FLIP solver, when compared to SPH, is that fewer timesteps per frame are required to run the simulation. FLIP requires 2 to 4 timesteps per frame, whereas SPH requires from 7 to

20 timesteps per frame for stable simulation results.

The FLIP solver is responsible for other simulation aspects such as collision behavior, and handling multiple fluid sources and sinks.

6.3 Input and Output

The user provides a polygonal object model for simulation. It is the user's responsibility to make sure the model has no holes in the surface geometry, and that all surface normals are pointing outward. Once the simulation is complete, the output will be an animated, deforming surface mesh which can then be used for lighting, shading, and rendering.

7. IMPLEMENTATION

Multiple HDA's have been created, each with its own function in controlling melting phase change. These are:

1. Section Create - Allows the user to create model sections using slicing planes
2. Heat Sources - Allows the user to specify attributes such as viscosity, temperature, and melting rate for each object section. Also specifies starting points on the model for heat propagation.
3. Temperature Control - Allows the user to control the viscosity based on temperature values.
4. Temperature Blend - Provides user control of heat propagation through the fluid.

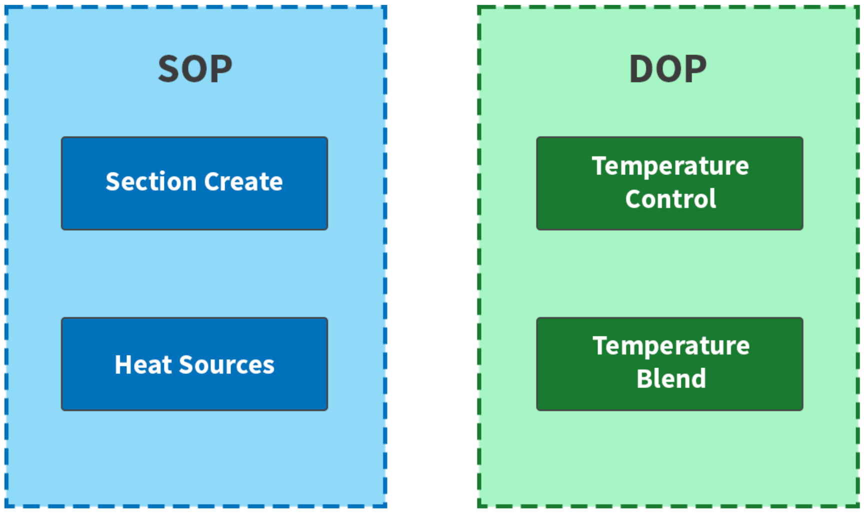


Figure 7.1: Collection of SOPs and DOPs created

The *Section Create* and *Heat Sources* digital assets work at the SOP level as this is where operations responsible for manipulating geometry are carried out. At the DOP level, dynamic operations such as simulation of particles are performed. *Temperature Control* and *Temperature Blend* work at the DOP level. This is shown in Figure 7.1.

7.1 Section Create

The first step of the implementation involves creating sections from the input object model and assigning attributes. A custom digital asset, called *Section Create*, was developed to handle the creation of sections on the model. It requires two inputs - a polygonal object model and the slicing geometry. As mentioned in the *Simulation Approach* chapter, it is the user's responsibility to make sure the model being used has no holes in its surface geometry, and that all surface normals are pointing outward.

A variety of sectioning geometry might be used to slice the object. Geometry such as spheres, cubes, or geometry of higher complexity could have been implemented as slicing geometry.

However, for this thesis, I have used planes as the slicing geometry as they are easy to place in the scene, do not obstruct the view of the input model when creating additional sections, and reduce the risk of creating unwanted sections when intersecting with other slicing geometry. The slicing planes can be positioned as the user wishes to create sections of the model. An example sliced object is shown in Figure 7.2.

Figure 7.3 shows the operator node network for the Section Create HDA and its input.

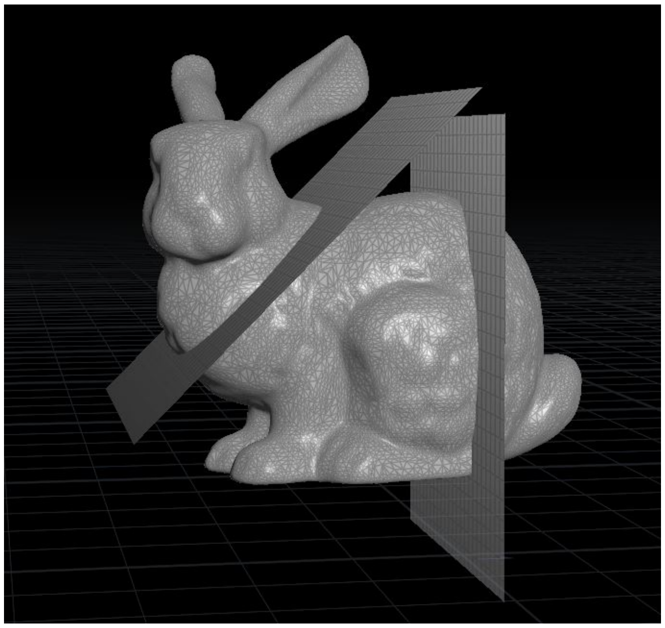


Figure 7.2: Input model with slicing planes

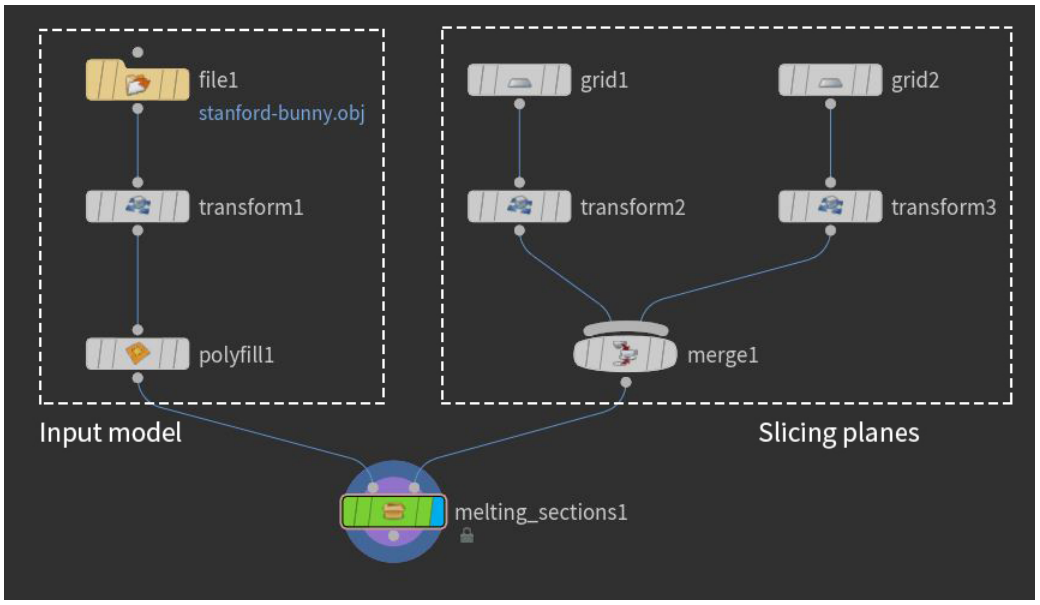
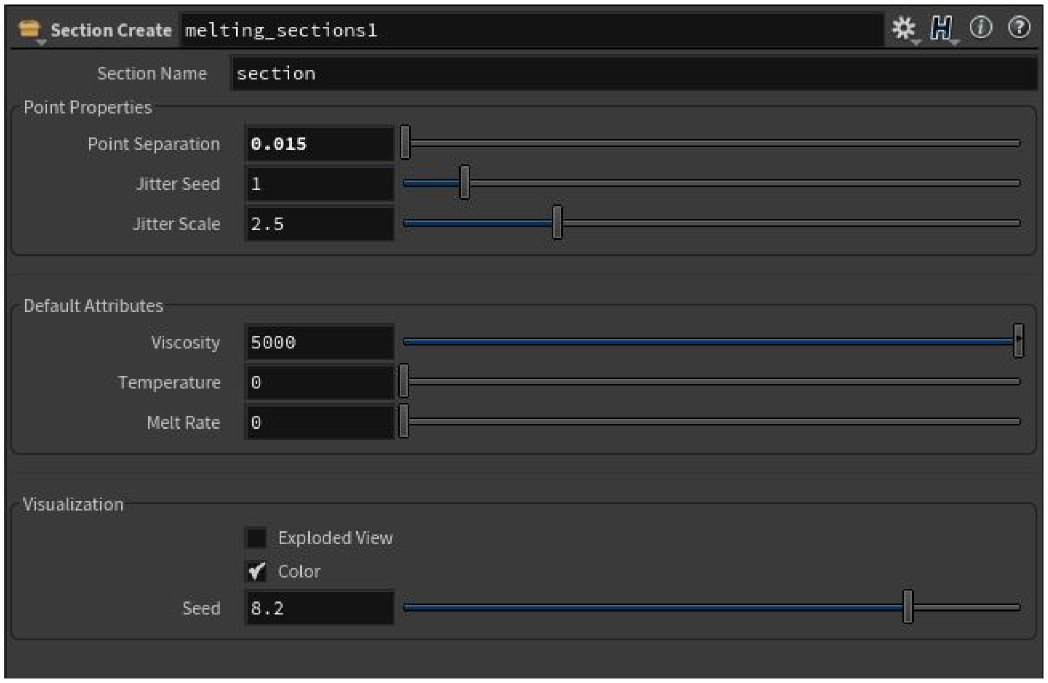


Figure 7.3: Section Create HDA (green) in the node network

The parameters and attributes used by Section Create are made accessible to the user, and these collectively form the user interface as shown in Figure 7.4. The user can set various parameter values such as particle separation, assign seed values to randomize color, and set attribute values for viscosity, temperature, and melting rate. The values assigned to these parameters are applied to the particles as a whole, and not to individual sections.

The particle separation describes the density of points packed within the input model. A lower particle separation value results in a higher number of particles packed in the object model. If temperature is assigned a default value greater than zero, this will act as the initial value for temperature for the whole input model during simulation. Color is used to visualize the sections created.

The output of this HDA is a particle filled representation of the model split into sections, as shown in Figure 7.5. Each section has its own identifier to be used in later stages.


Section Create melting_sections1

Section Name section

Point Properties

Point Separation 0.015

Jitter Seed 1

Jitter Scale 2.5

Default Attributes

Viscosity 5000

Temperature 0

Melt Rate 0

Visualization

Exploded View

✓ Color

Seed 8.2

Figure 7.4: Section Create user controls

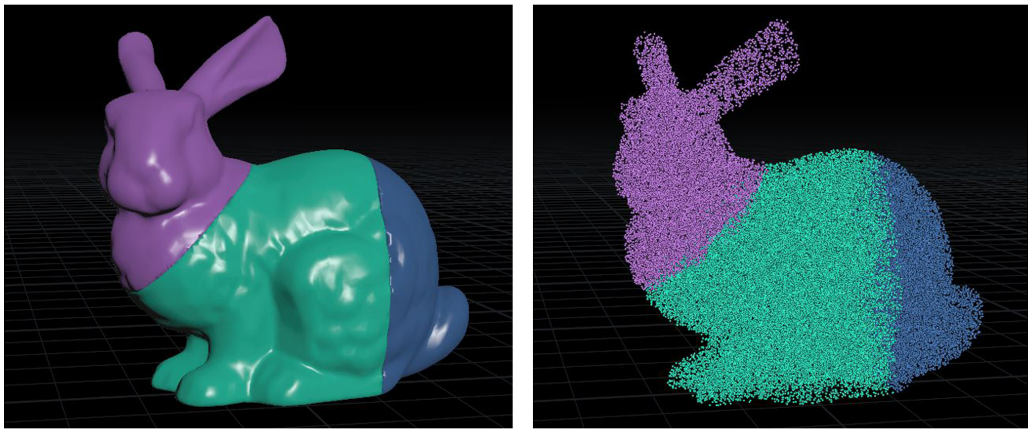


Figure 7.5: The sections created by the Section Create HDA (left) and the resulting particle filled representation (right)

7.1.1 Inside the Section Create HDA

The detailed node structure used within Section Create is as shown in Figure 7.6. Attributes required for simulation such as temperature, viscosity, and melting rate are created and initialized to their default values as specified by the user. This ensures that all particles within the geometry possess these attributes. This is done using an *Attribute Create* SOP. The values for these attributes are set by the user using the Section Create UI as shown in Figure 7.4.

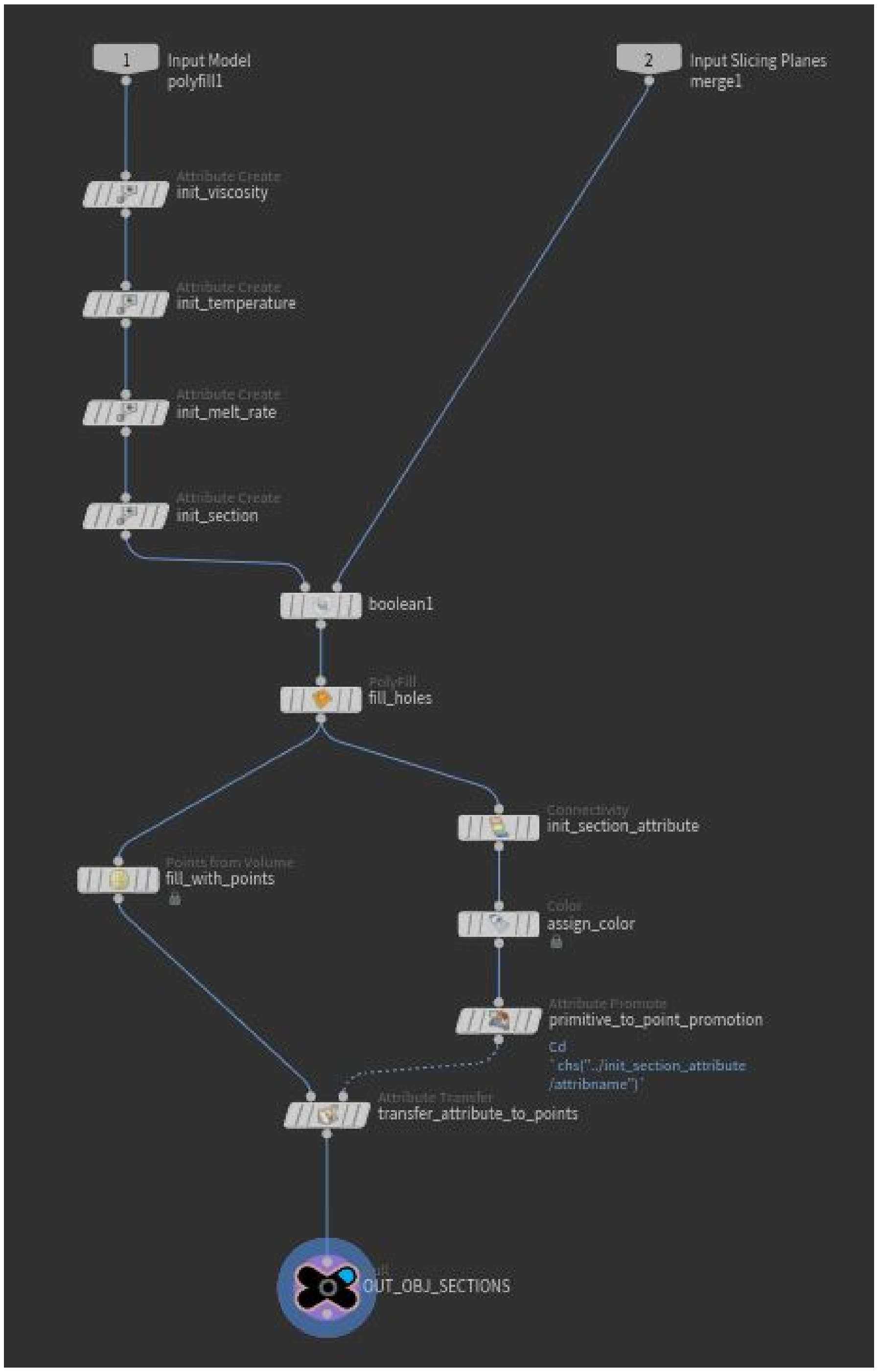


Figure 7.6: Collection of SOP nodes used to build the Section Create HDA

The polygon model is one of the inputs for the *Boolean* SOP. The other input is the planes used for slicing. The Boolean SOP settings are set to *Shatter*. This node is responsible for creating sections on the model. While it was originally used in destruction simulation to create fractured objects, it can be used to perform a variety of operations such as finding the Intersection or Union of objects, seams, etc. This operation will fail, or lead to inaccurate results, if the input model has holes or does not have outward pointing normals. A *Polyfill* SOP is used to fill any holes in the geometry once the sections are created.

There are two output streams from the Polyfill operation. The first output stream is an input to the *Points from Volume* SOP. This operator is used to fill the object model with particles. The particle separation, jitter values, and seed value for noise can be adjusted using its UI.

The second output stream is passed to a *Connectivity* SOP which assigns a reference name to each section, along with an integer ID value. Each section is assigned a random color to distinguish it from the other.

With the exception of the Points from Volume SOP, these SOPs have been created at a *Primitive* class level. A Primitive in Houdini is a unit of geometry that is lower in level than objects, but higher than points and vertices. For these attributes to be accessible at a Point class level, we use the *Attribute Promote* SOP to change the class from Primitive to Points.

The *Attribute Transfer* SOP is used to transfer attributes from one geometry to another. The first input is the destination geometry to which attributes are transferred. The second input is the source geometry which provides the attributes to be transferred. The attributes transferred are the *Section ID* and color. These attributes are transferred to the points created using Points from Volume SOP. This ensures that all points created possess a Section ID and color attribute.

7.2 Heat Source HDA

The approach used here is one way to locate heat sources. Other method to locate heat sources could also be used. This asset is used with the Section Create HDA. While Section Create is used to assign attribute values to the particles as a whole, Heat Source assets are used to assign attribute values to individual sections. The other purpose of this asset is to determine points on the input model to be used as heat sources. Since there are no external heat sources, it is important to find points from where heat propagates into the fluid. These points can be selected automatically by the asset or manually by the user. For manual selection, the user selects a point and enters the corresponding point number into the UI. When the automatic method is used, the point in each section that is furthest away from the centroid of the input model is used as a heat source. This selection usually allows parts of the input model that have less mass and are structurally weak to start melting first.

A heat source point is created for each section and assigned a temperature and melting rate for simulation. Each heat source points is also assigned a heat source ID. To mimic heat wave fronts passing through the liquid, polygonal spheres are placed at each heat source point. These spheres are assigned the temperature and melting rate specified by the user. The spheres grow in size over time in accordance with their assigned melting rate. These spheres are the output of the Heat Sources asset, and are used in the simulation to propagate heat through the particle fluid.

Figure 7.7 shows the usage of the Heat Source digital asset in the node network.

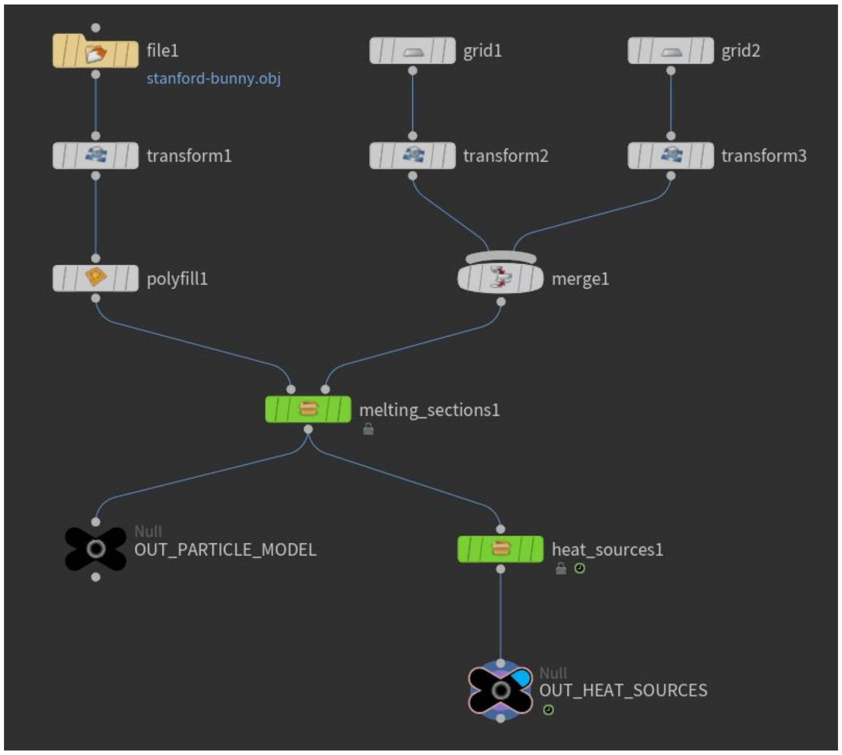
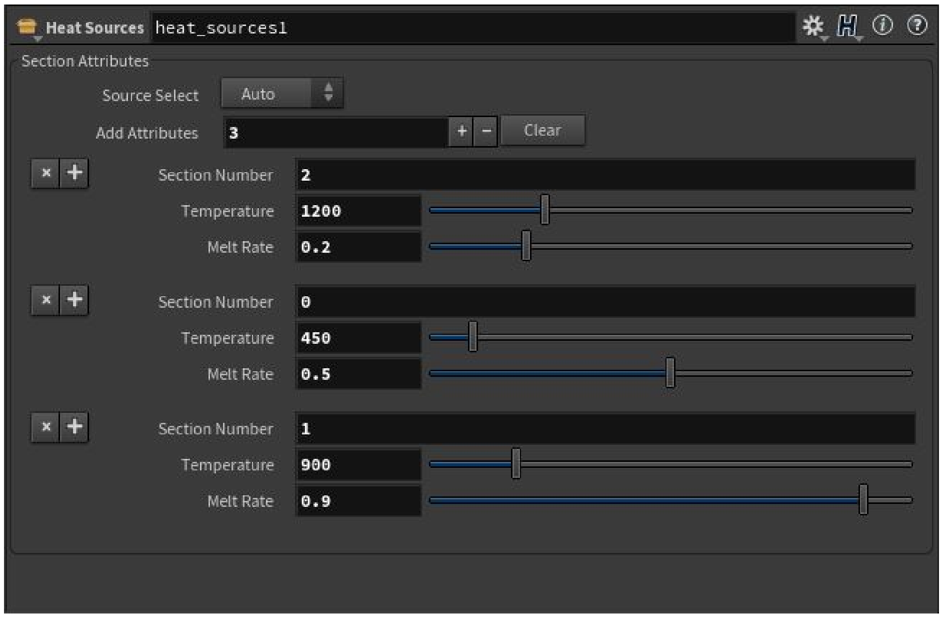


Figure 7.7: Heat Source HDA in the node network


Heat Sources heat_sources1

Section Attributes

Source Select: Auto

Add Attributes: 3

Section Number: 2
Temperature: 1200
Melt Rate: 0.2

Section Number: 0
Temperature: 450
Melt Rate: 0.5

Section Number: 1
Temperature: 900
Melt Rate: 0.9

Figure 7.8: Heat Source HDA controls

As seen in Figure 7.8, the user can assign values for temperature and melting rate to each section. The user adds attributes to a section by specifying its section number. The user may assign attribute values to sections as needed. When attribute values for a section are not explicitly assigned, default values are used. The user can assign values for sections in any order.

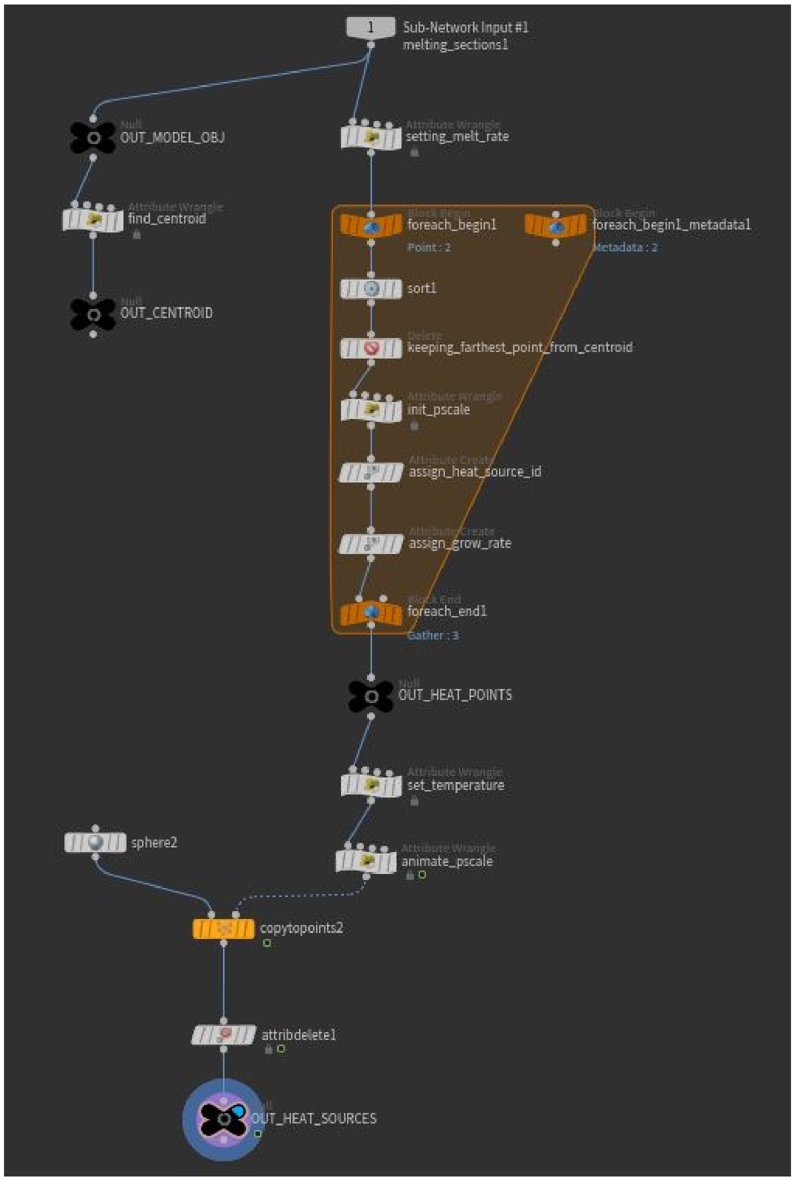
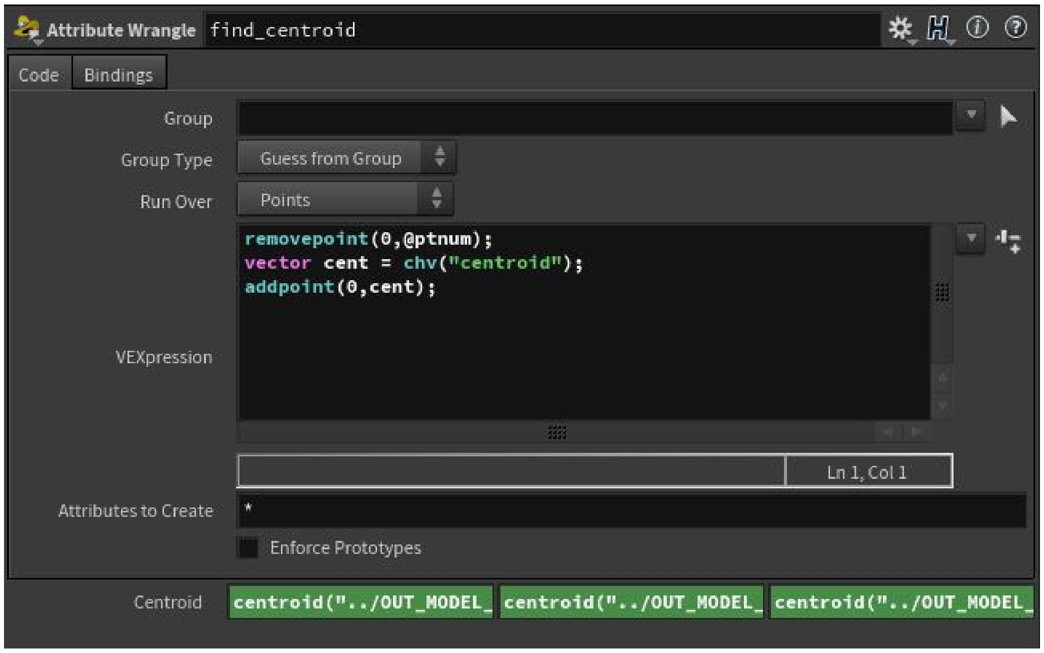


Figure 7.9: The collection of SOP nodes used to build the Heat Source HDA

7.2.1 Inside the Heat Source HDA

A detailed node network for the Heat Source HDA is shown in Figure 7.9.

The *Attribute Wrangle* SOP is a powerful, versatile, low-level node that allows the user to manipulate geometry using VEX code. This SOP is used to add a single point at the centroid of the object model, as shown in Figure 7.10.



find_centroid

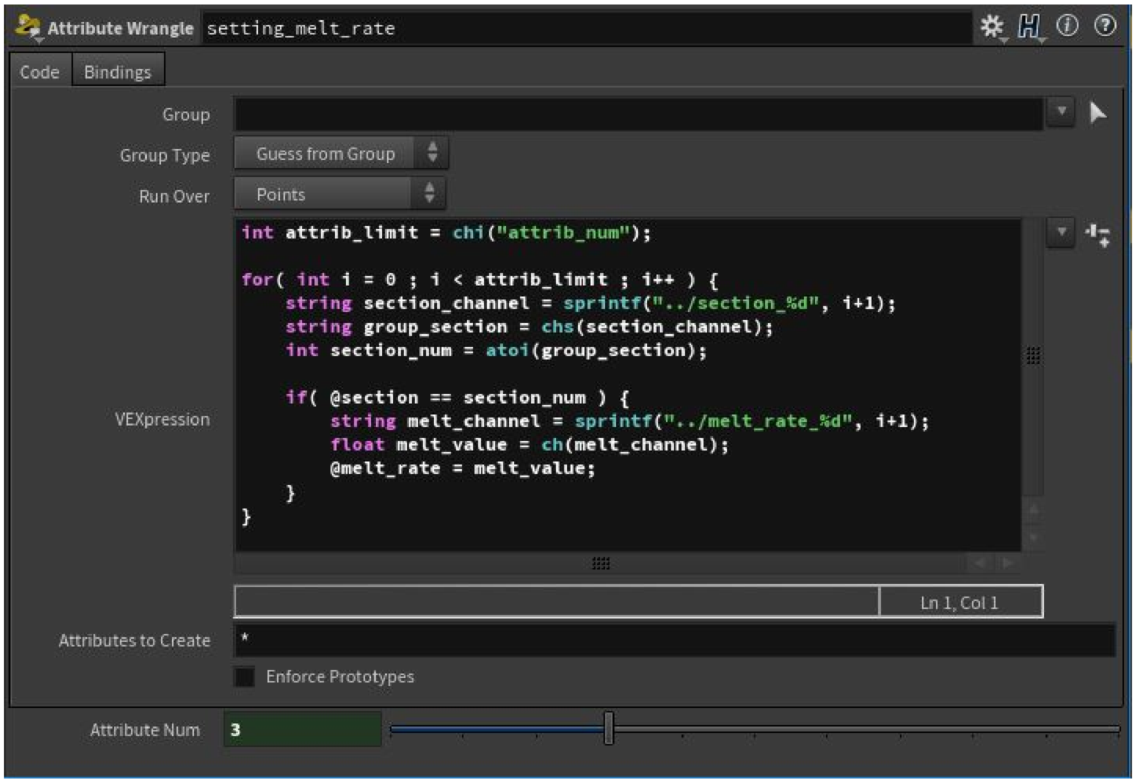
Group:
Group Type: Guess from Group
Run Over: Points
VEXpression:

```
removepoint(0,@ptnum);
vector cent = chv("centroid");
addpoint(0,cent);
```

Attributes to Create: *
Enforce Prototypes
Centroid: centroid("../OUT_MODEL_ centroid("../OUT_MODEL_ centroid("../OUT_MODEL_

Figure 7.10: Finding the centroid of the input object model

The Attribute Wrangle SOP is used again to set the melting rates for each section using the values provided by the user, as shown in Figure 7.11. The user input is parsed from the UI by the Attribute Wrangle and assigned to all points belonging to that particular section.



```
setting_melt_rate

int attrib_limit = chi("attrib_num");

for( int i = 0 ; i < attrib_limit ; i++ ) {
    string section_channel = sprintf("../section_%d", i+1);
    string group_section = chs(section_channel);
    int section_num = atoi(group_section);

    if( @section == section_num ) {
        string melt_channel = sprintf("../melt_rate_%d", i+1);
        float melt_value = ch(melt_channel);
        @melt_rate = melt_value;
    }
}
```

Figure 7.11: VEX code to set the melting rate for each section

Once the melting rate for each section is assigned to all the points, the data is then run through a for-each loop. This loop iterates through each section based on the section number. The operations performed in the for-each loop are as shown below in Figure 7.12.

The first task performed within the for-each loop is to sort the points of each section based on their distance from the centroid of the object model. The sorting is performed by the *Sort* SOP. This sorting is done in descending order; points further away from the centroid have lower point numbers. This ensures that the point furthest from the centroid always has a point number of zero. This is essential in finding the points to use as heat sources.

The *Delete* SOP isolates points from each section that have a point number equal to zero. These points are then assigned a new attribute known as a *Heat Source ID*.

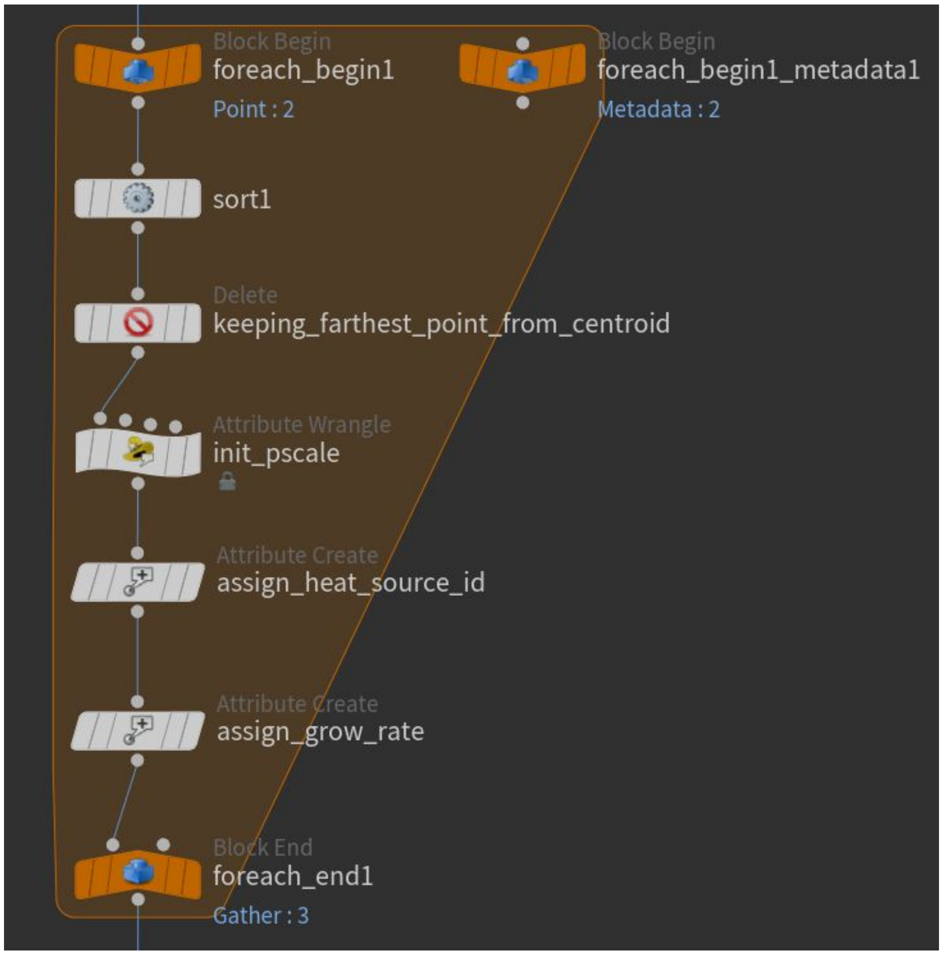
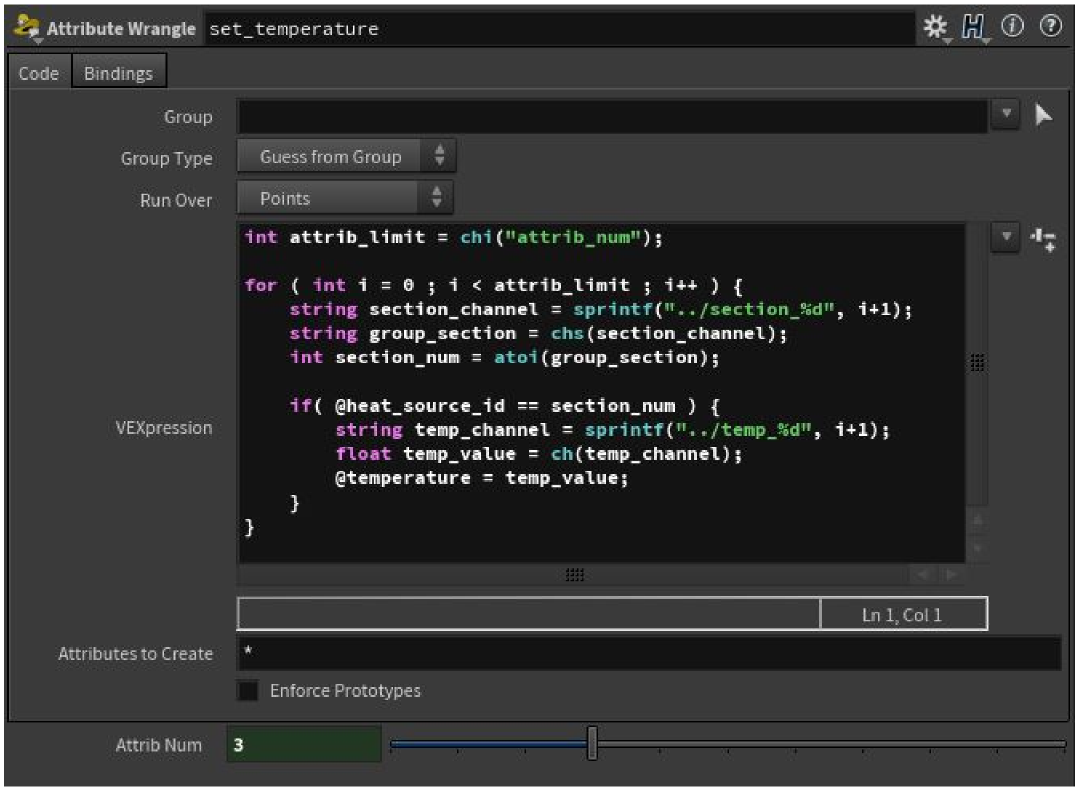


Figure 7.12: The For-Each loop responsible for performing operations on each section



set_temperature

Group
Group Type: Guess from Group
Run Over: Points
VEXpression:

```
int attrib_limit = chi("attrib_num");

for ( int i = 0 ; i < attrib_limit ; i++ ) {
    string section_channel = sprintf("../section_%d", i+1);
    string group_section = chs(section_channel);
    int section_num = atoi(group_section);

    if( @heat_source_id == section_num ) {
        string temp_channel = sprintf("../temp_%d", i+1);
        float temp_value = ch(temp_channel);
        @temperature = temp_value;
    }
}
```

Ln 1, Col 1
Attributes to Create: *
Enforce Prototypes
Attrib Num: 3

Figure 7.13: VEX code to set the temperature value for each heat source

Upon completion of the for-each loop, each section will have a dedicated heat source point. Using an Attribute Wrangle, the temperature values are then assigned to the corresponding heat sources using a method similar to setting the melting rates for all the points. This is shown in Figure 7.13. Polygonal spheres are copied to each of these heat source points using the *Copy to Points* SOP, as shown in Figure 7.14. As these spheres grow for each simulation frame, they will act as heat wavefronts to simulate the propagation of heat through the section.

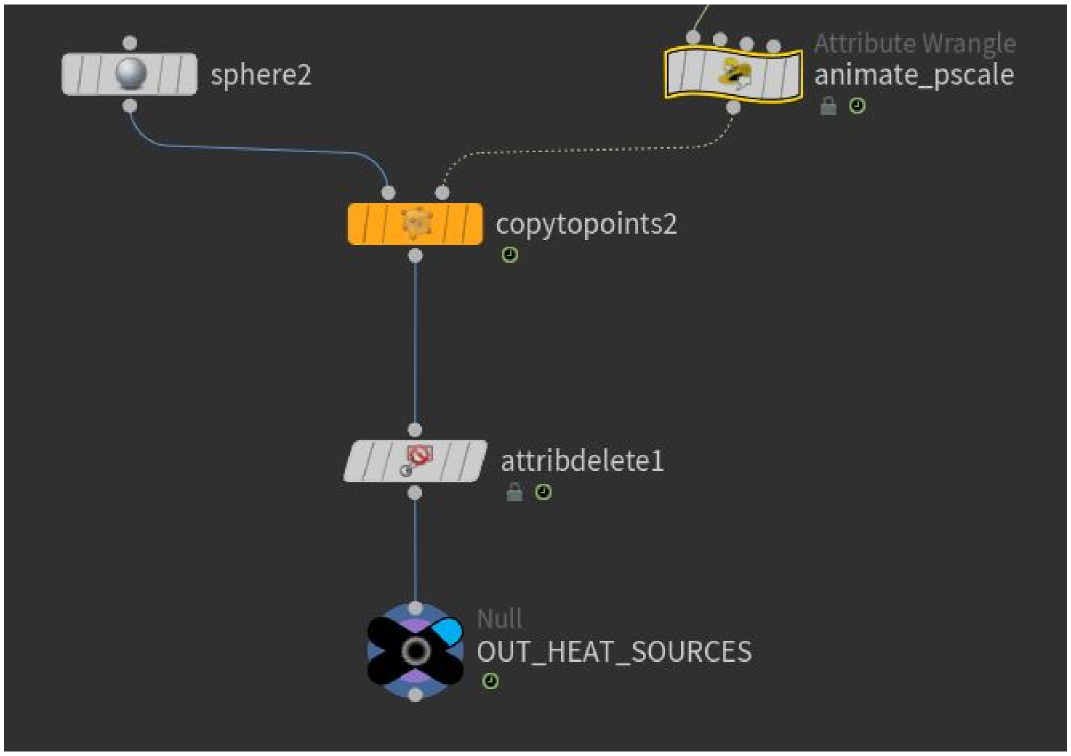


Figure 7.14: Copying polygonal spheres onto the heat source points

7.3 Simulation Setup

The built-in Houdini FLIP solver is used as the basis for the melting simulation. Two DOP digital assets have been created to work with this solver.

7.3.1 FLIP Simulation

When a FLIP fluid is created using the FLIP tool, a default DOP network is created with the FLIP simulation operators as shown in Figure 7.15.

The *FLIP Solver* is responsible for collecting all the data from the input nodes and simulating the particles accordingly. The FLIP solver has four nodes which accept input. These are:

1. Fluid To Solve: Objects to undergo simulation are connected here.

2. Particle Velocity: DOPs that manipulate particle attributes are connected to this input.
3. Volume Velocity: Field affecting operators are added to this node.
4. Sourcing: This input is for adding or removing particles, and adding forces after the main simulation steps are completed.

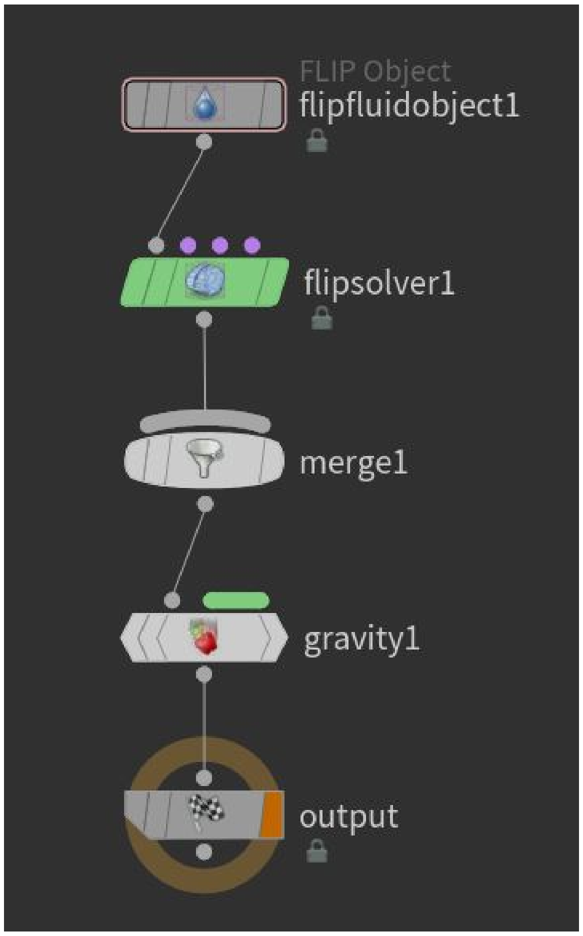
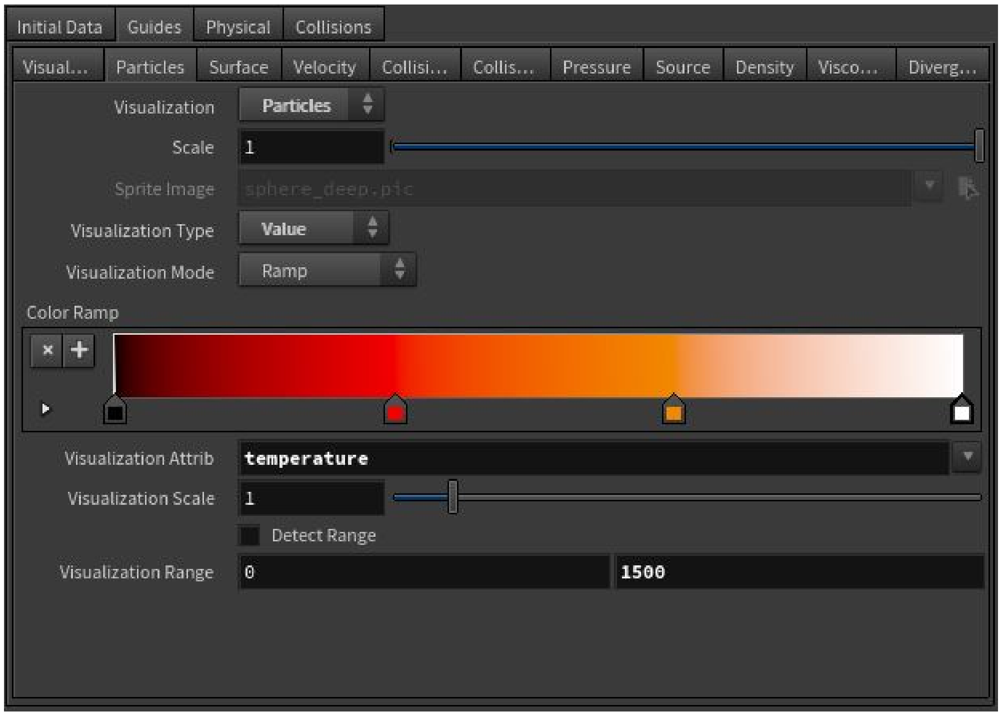


Figure 7.15: FLIP simulation setup using Houdini's preset tools

The user can configure the *Flip Object* DOP by setting the SOP path to the output of the Section Create DOP and changing the input type from *Surface SOP* to *Particle Field*. This is needed since we are passing point data to the Flip Object, not Surface data. Figure 7.16 shows the visualization settings used.



Visualization: Particles
Scale: 1
Sprite Image: sphere_deep.pic
Visualization Type: Value
Visualization Mode: Ramp
Visualization Attrib: temperature
Visualization Scale: 1
Detect Range
Visualization Range: 0 1500

Figure 7.16: Visualization settings for the FLIP object to show changes in temperature during simulation

The user can modify the visualization parameters to display changes in temperature or viscosity rather than velocity, if desired. For this simulation, the settings have been changed to display temperature in a range of 0 to 1500. The visualization type has been changed from *Speed* to *Value* to represent changes in temperature. A color ramp is used to visualize the temperature values. These settings are shown in Figure 7.16

The viscosity attribute needs to be enabled in the FLIP Solver, as it is disabled by default. Finally, the *Gravity* DOP is deleted, as gravity is handled by the Temperature Blend HDA. If this DOP were left intact, gravity would act on all the particles and cause the object particles to collapse. The Temperature Blend HDA implements a gravity force taking into account changes in temperature and viscosity.

7.3.2 Collision Objects

For this simulation, a ground plane has been created to act as collision geometry. This simulation network is shown in Figure 7.17. Using additional *Static Object* DOPs, the user could add additional collision geometry.

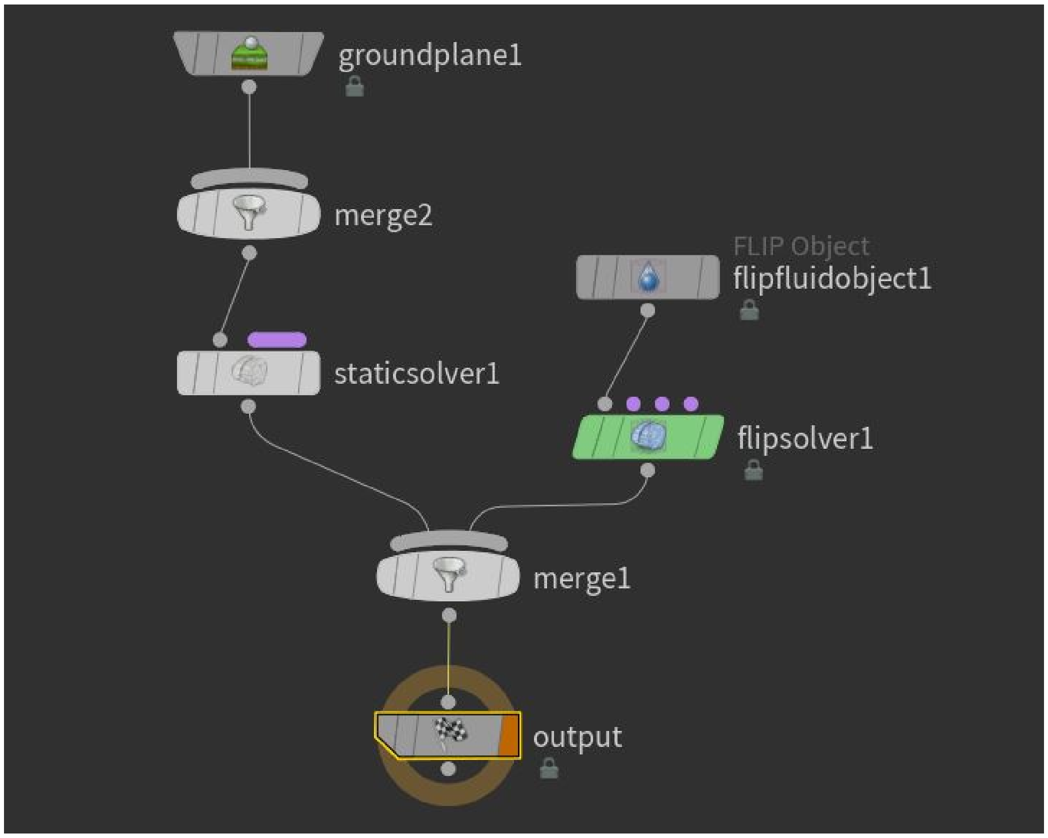


Figure 7.17: Updated network with ground plane collision geometry.

7.3.3 Custom Digital Assets Developed

The two custom digital assets created to control melting simulations are Temperature Control and Temperature Blend. The Temperature Control HDA deals with particle attributes such as temperature and viscosity. It is connected to the second input on the FLIP Solver. The Temperature Blend is connected to the sourcing FLIP input. It imports heat source spheres as surface geometry and applies a gravity force. Figure 7.18 shows the simulation network with these two custom digital assets added.

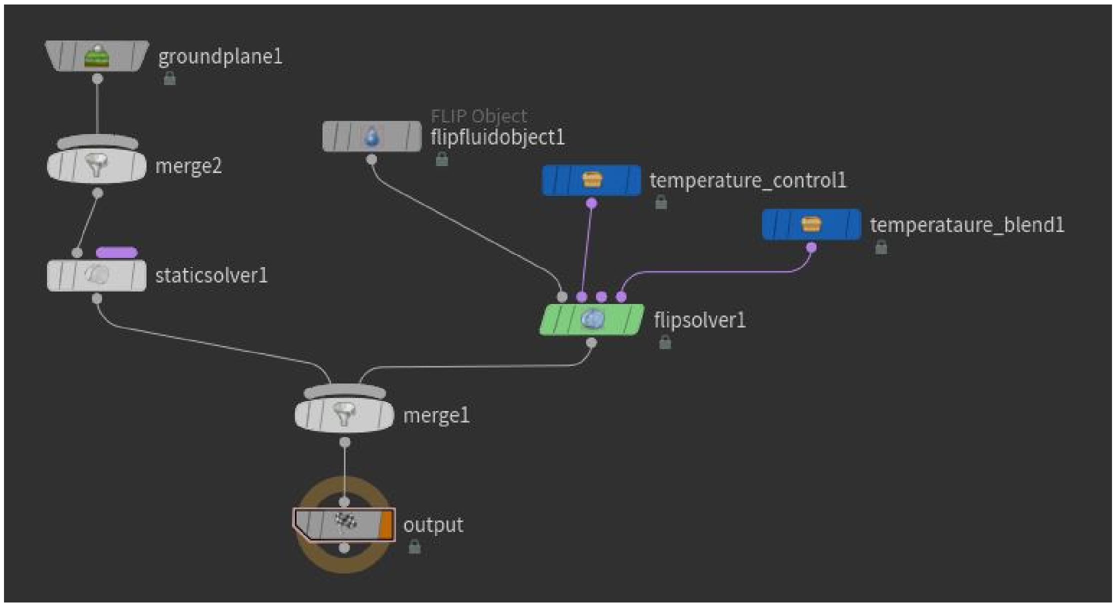


Figure 7.18: Simulation network with custom digital assets (blue)

7.4 Temperature Control

This asset is responsible for performing two tasks - checking simulation conditions and also manipulating viscosity based on temperature. If a heat source wavefront passes through particles from another section, their temperature remains unchanged. If the heat source passes through particles from its own section, its temperature is transferred to those particles. The viscosity of the fluid changes in accordance with the temperature value of its particles.

Figure 7.19 shows the controllable parameters of this asset. The user can specify minimum and maximum values for viscosity. Increasing the minimum viscosity will result in a "thicker" fluid at its highest temperature.

Temperature parameters such as a threshold temperature and cooling rate can also be controlled by the user. The threshold temperature is the value at which the high viscosity fluid begins to melt. The cooling rate specifies the amount by which temperature will be reduced at each time step. A value of 0.99 reduces the temperature by 1% at each time step. Similarly, a value of 0.9 reduces the temperature by 10%, and so on.

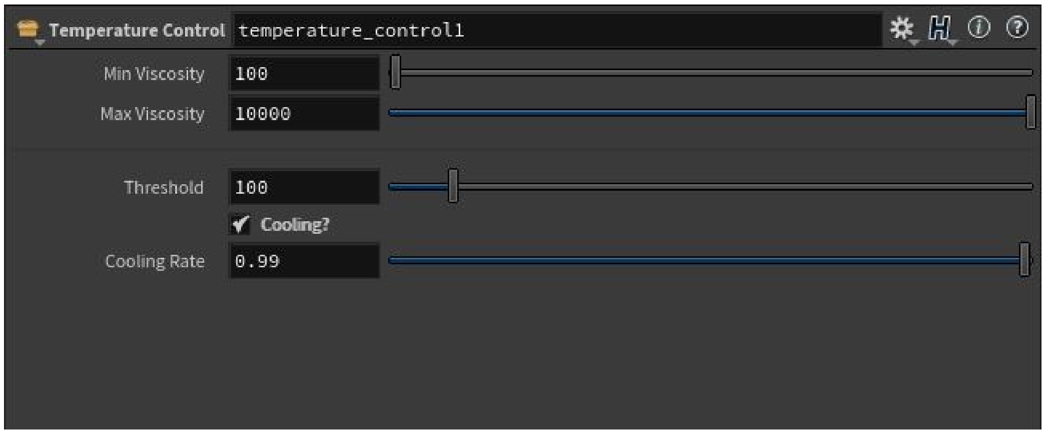

Temperature Control temperature_control1
Min Viscosity 100
Max Viscosity 10000
Threshold 100
✓ Cooling?
Cooling Rate 0.99

Figure 7.19: Temperature Control HDA controls

7.4.1 Inside the Temperature Control HDA

As seen in Figure 7.20, two Attribute Wrangle operators control the particle attributes responsible for simulating melting.

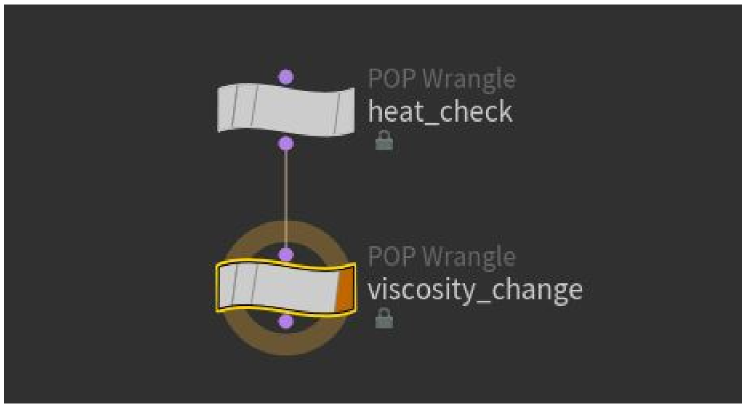
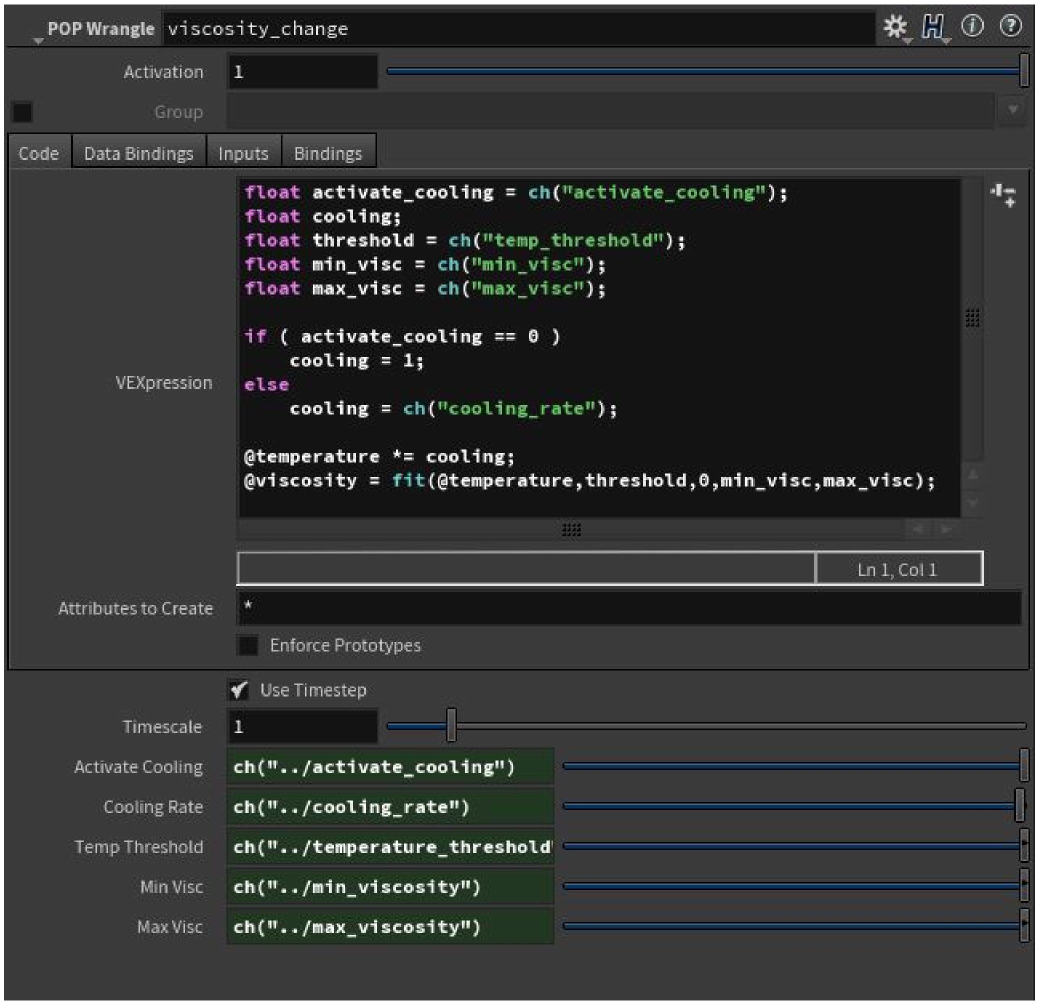


Figure 7.20: Nodes responsible for controlling the simulation

As heat propagates through the fluid, *Heat Check* compares each heat source ID to the section number of each particle at each time step to make sure that only particles from that particular section are being assigned a temperature value.

The *Viscosity Change* wrangle checks if cooling is enabled and then reduces the temperature by the cooling rate at each time step. This temperature is then mapped to a viscosity scale ranging from a minimum and maximum values to set the viscosity of the particles at that time step.



```
float activate_cooling = ch("activate_cooling");
float cooling;
float threshold = ch("temp_threshold");
float min_visc = ch("min_visc");
float max_visc = ch("max_visc");

if ( activate_cooling == 0 )
    cooling = 1;
else
    cooling = ch("cooling_rate");

@temperature *= cooling;
@viscosity = fit(@temperature,threshold,0,min_visc,max_visc);
```

```
Activate Cooling   ch("../activate_cooling")
Cooling Rate       ch("../cooling_rate")
Temp Threshold     ch("../temperature_threshold
Min Visc           ch("../min_viscosity")
Max Visc           ch("../max_viscosity")
```

Figure 7.21: VEX code to change viscosity according to temperature

Figure 7.21 shows the code snippet used to change viscosity based on the temperature at that time step. At the bottom of the figure, the values for cooling rate, minimum and maximum viscosity, temperature threshold, and cooling activation are shown referencing the values from the Temperature Control UI shown in Figure 7.19.

7.5 Temperature Blend

This digital asset is responsible for importing the temperature values from the Heat Sources asset to the FLIP simulation and transferring them to the points to be simulated.

It provides users with two controls on heat propagation from the heat sources. These are shown in Figure 7.22 as the *Distance Threshold* and the *Blend Width*. The Distance Threshold parameter sets the offset ahead of the heat sources where the temperature being applied to the particles will be constant. All particles within this range will have constant temperature, which in this case is the maximum temperature of that heat source. The Blend Width parameter is an additional offset parameter ahead of the Distance Threshold that applies the temperature gradually as the heat propagates through the fluid.

A second task of this HDA is to apply a gravity force to the particles. Gravity force only acts on particles that have temperature transferred to them during the simulation.

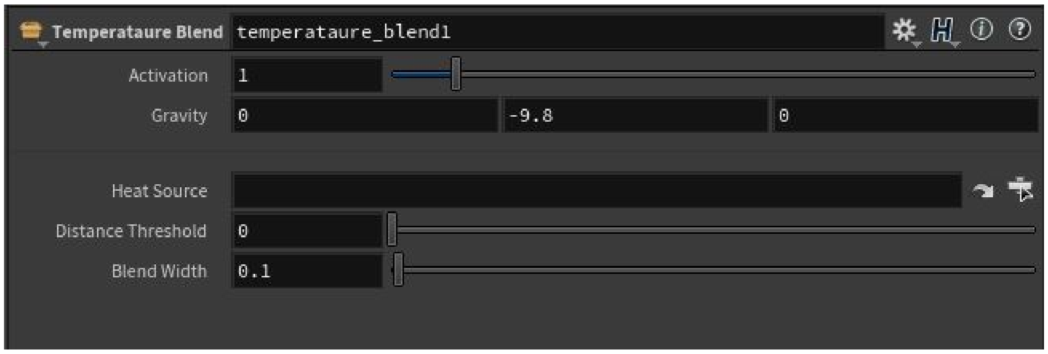

Temperataure Blend temperataure_blend1
Activation 1
Gravity 0 -9.8 0
Heat Source
Distance Threshold 0
Blend Width 0.1

Figure 7.22: Temperature Blend controls

7.5.1 Inside the Temperature Blend HDA

SOP Solvers, as shown in Figure 7.23, allows the user to import SOP geometry data to be used in a DOP simulation. Inside the SOP Solver the *Object Merge* operator is responsible for importing the heat source geometry from the Heat Source HDA. The Attribute Transfer SOP transfers the temperature attribute from the imported SOP geometry to the DOP simulation. This setup is shown in Figure 7.24

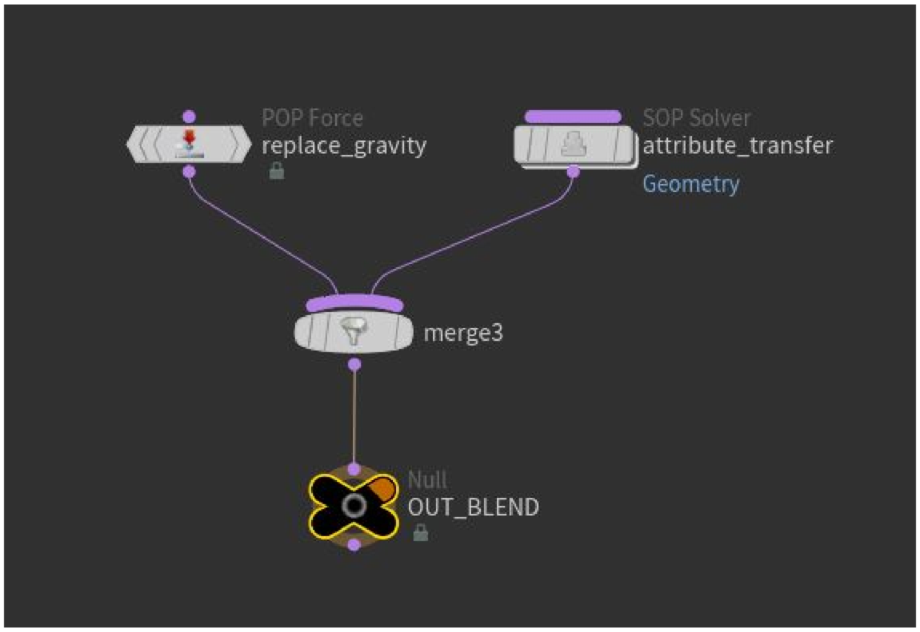


Figure 7.23: Inside the Temperature Blend HDA

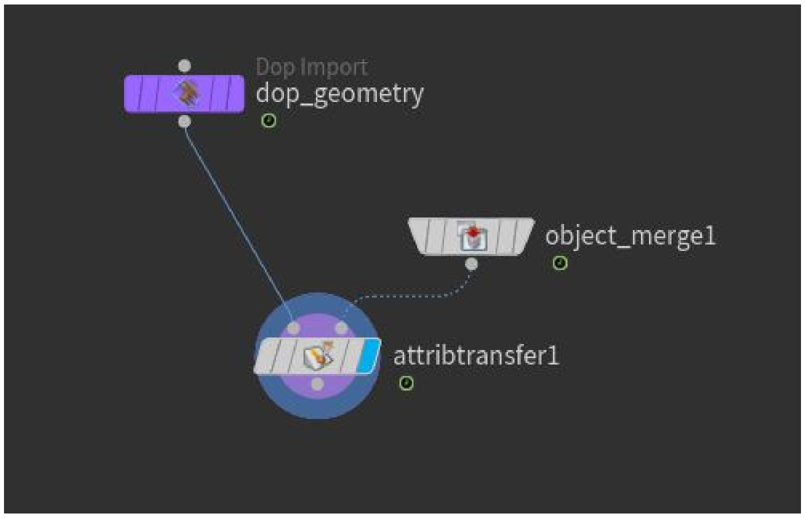
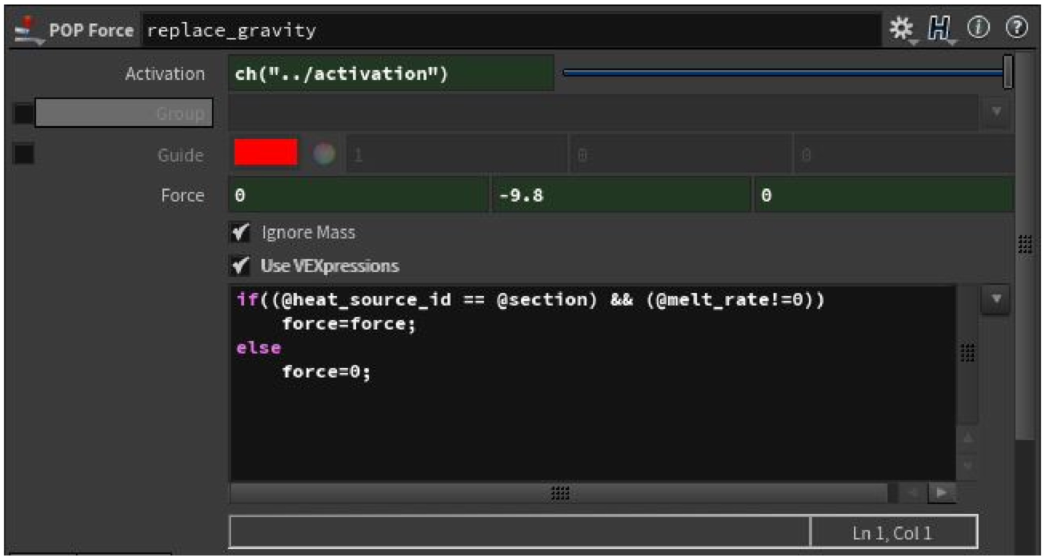


Figure 7.24: Inside the "attribute_transfer" SOP Solver shown in Figure 7.23

The *POP Force* operator, as shown in Figure 7.25, is responsible for simulating gravity. For each particle, if its Section ID matches the Heat Source ID, it experiences a gravitational force. If not, the particle does not experience gravity.


POP Force replace_gravity

Activation ch("../activation")

Group

Guide 1 0 0

Force 0 -9.8 0

✓ Ignore Mass

✓ Use VEXpressions

```
if((@heat_source_id == @section) && (@melt_rate!=0))
    force=force;
else
    force=0;
```

Ln 1, Col 1

Figure 7.25: Gravity force and its conditions

7.6 Running The Simulation

Once the custom digital assets are set up in the simulation network, the user can change parameters within the FLIP Object and FLIP Solver nodes as desired. Changes can be made to the parameters of the four custom digital assets to achieve different results. The user can then run the simulation.

When the FLIP tool is used to create a fluid simulation, a node network of operators to generate a deforming polygonal mesh from the simulation is also created. This mesh can be used for lighting, shading, rendering, or exporting to other 3D animation packages.

8. RESULTS

The control approach was tested on three object models - the Stanford Bunny, a statue of Venus, and a toy soldier. Multiple simulations were carried out with variations in base viscosity, minimum-maximum viscosity, temperature, cooling rates, melting rates, the number of sections created and the number of sections set to melt.

For each test case, images of the simulation were rendered using Mantra. Surface materials such as chocolate, plastic, and clay from Houdini's built-in library were used.

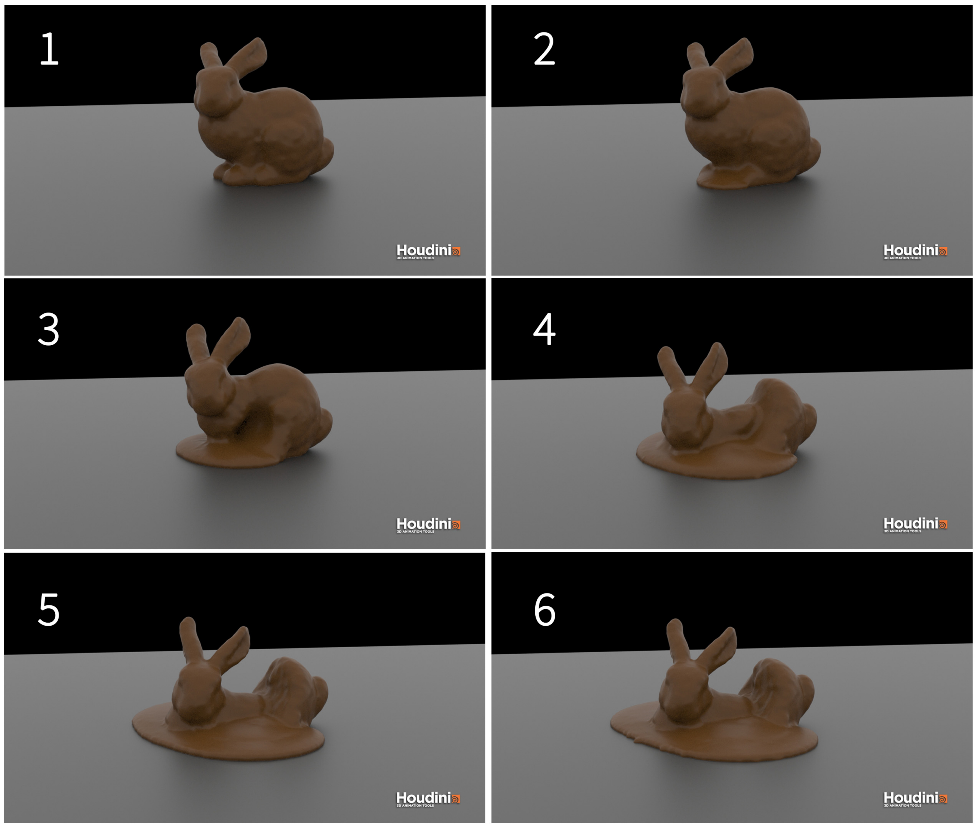


Figure 8.1: Melting a section of the Stanford Bunny

To achieve the melting shown in Figure 8.1, the Stanford Bunny is split into three sections - the head, the middle, and the rear. Default values are used for the head and rear sections so they would not melt. The middle was simulated using the values shown in Table 8.1. The head and the rear don't melt as they are not subjected to any heat transfer from the middle section heat source. As a result their viscosity values remains constant and they maintain their shape.

Property	**Value**
Section	0 (middle)
Point Separation	0.01
Viscosity	5000
Temperature	1000
Melting Rate	0.4
Min-Max Viscosity	10 - 10000
Threshold Temp	100
Cooling Rate	0.98
Frame Range	1 - 240

Table 8.1: Simulation settings

To melt the entire model, section attribute values were set for all sections. Figure 8.2 shows the resulting melting when the sections are assigned varying temperature and melting rate values. The varying values used by each section are shown in Table 8.2. Remaining values such as point separation, min-max viscosity, threshold temperature, cooling rate are the same as in Table 8.1.

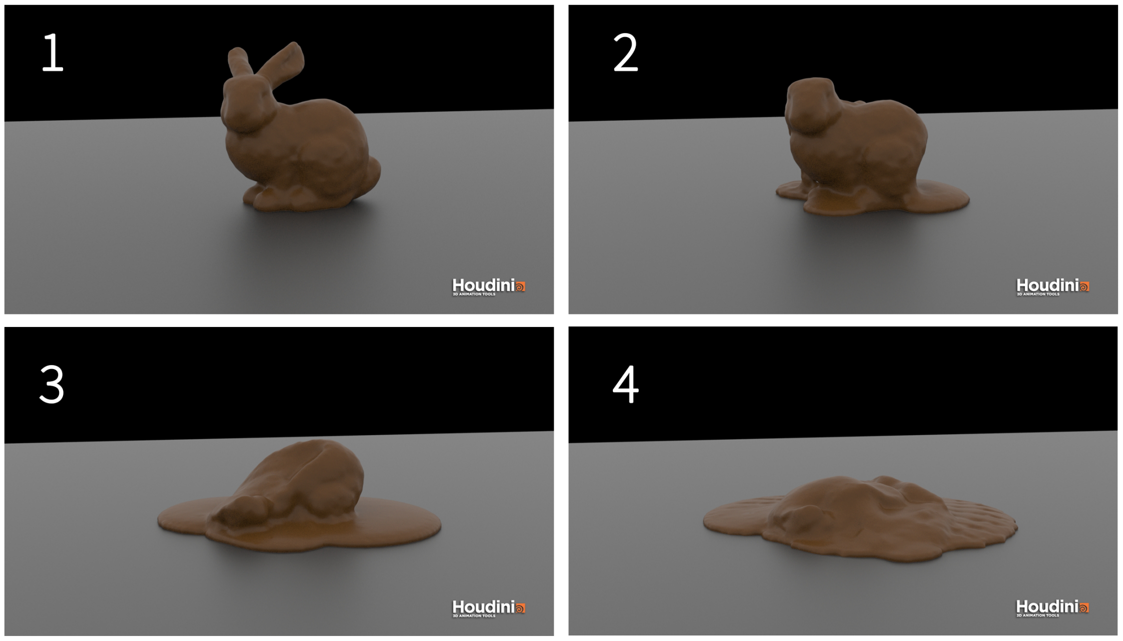


Figure 8.2: Melting all sections of the Stanford Bunny

Section	**Viscosity**	**Temperature**	**Melting Rate**
middle	5000	800	0.25
head	5000	1000	0.5
rear	5000	1200	0.9

Table 8.2: Simulation settings for Figure 8.2

Melting was also carried out on a 3D model of a toy soldier. This model was split into four sections - the head, the middle, and two legs. Similar to the previous test, each section was assigned a different temperature and melting rate. Figure 8.3 shows the resulting simulation and Table 8.3 shows the settings used.

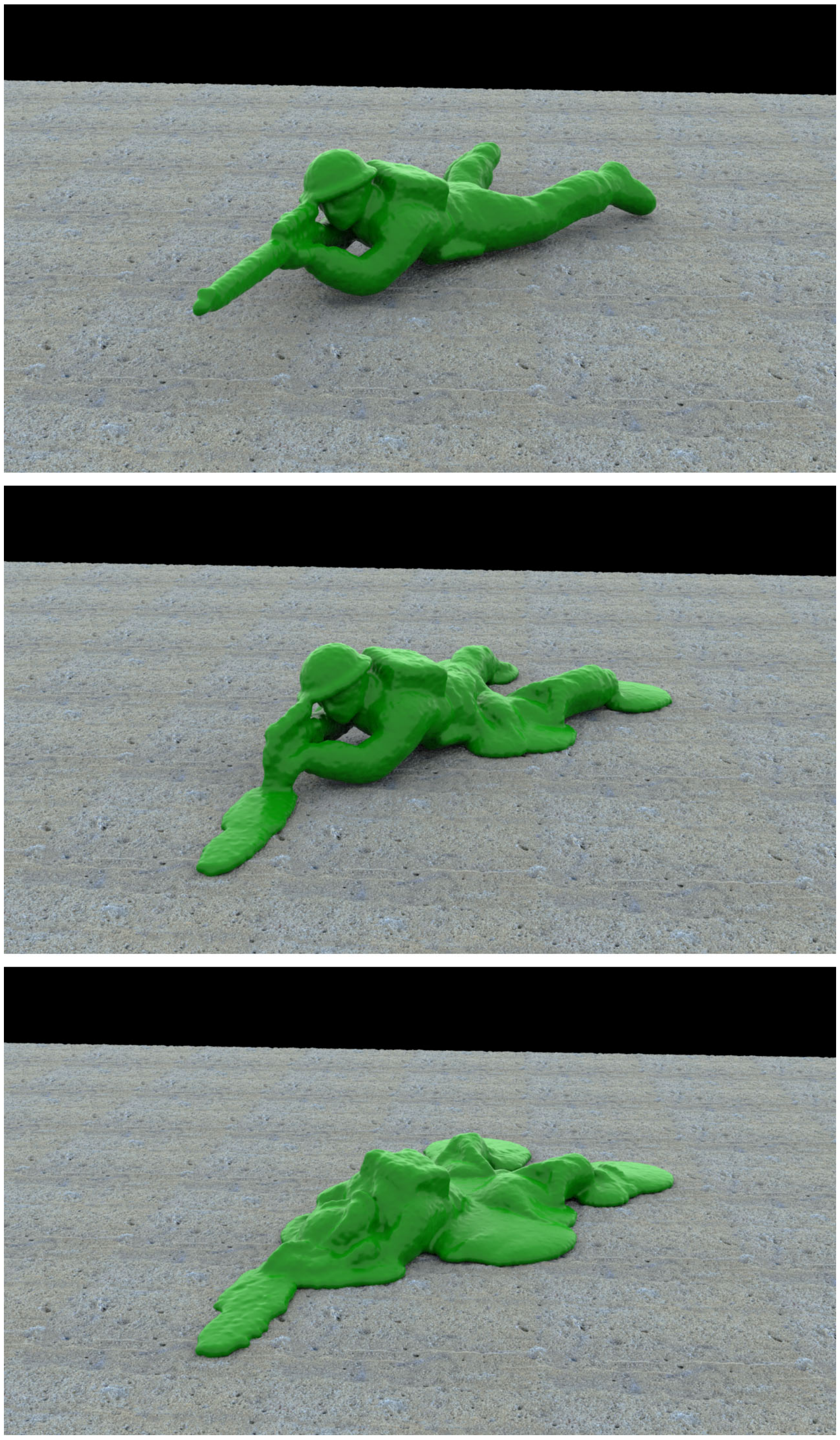


Figure 8.3: Melting all sections of a toy soldier

Section	**Viscosity**	**Temperature**	**Melting Rate**	**Cooling Rate**
head	8000	800	0.9	0.99
middle	8000	1400	0.6	0.99
leg (r)	8000	1000	0.2	0.99
leg (l)	8000	1000	0.3	0.99

Table 8.3: Simulation settings for Figure 8.3

To demonstrate how different sections can melt at different rates, a simulation was carried out on the legs of the toy soldier with a different melting rate assigned to each. Viscosity, temperature and cooling rate values were kept the same as before. The melting rates assigned to the right and left leg were 0.2 and 0.8 respectively. Figure 8.4 shows the melting progress on the legs at frame 48 of the simulation.

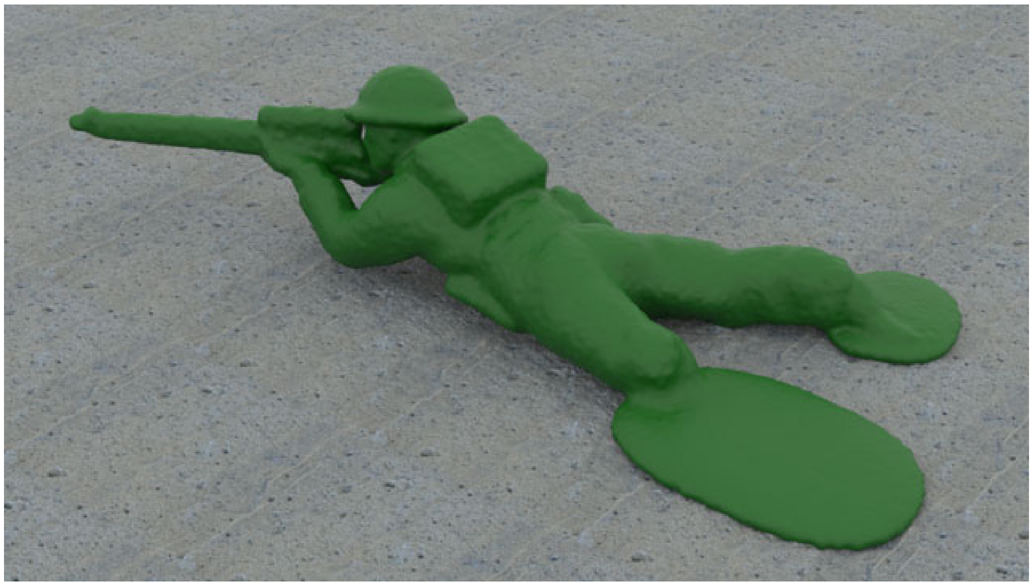


Figure 8.4: Melting the legs at different rates.

By changing the minimum viscosity in Temperature Control, it is possible to demonstrate that the simulation results in a thicker melted fluid as shown in Figure 8.5. Previous simulations had a minimum viscosity value of 10, which resulted in the melted fluid having a very low viscosity. For this test, the minimum viscosity value was increased to 1000.

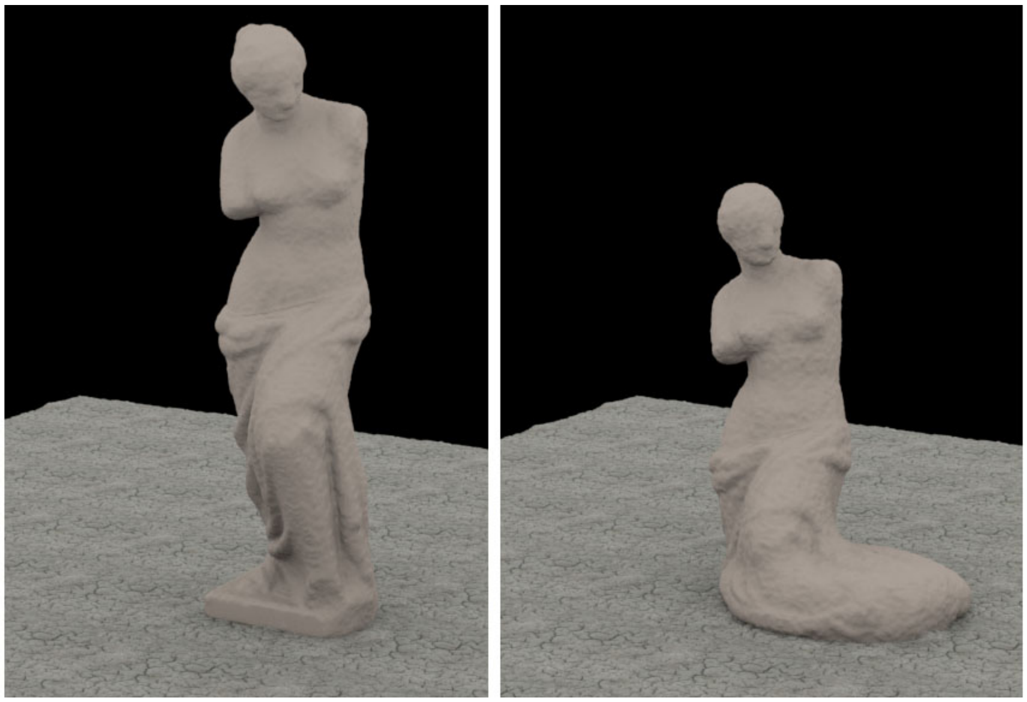


Figure 8.5: Original model (left) and with the base melted (right)

9. CONCLUSIONS

The developed Houdini Digital Assets work as outlined in the Goals chapter. They provide the user with a straightforward workflow to control object melting. Users experienced with creating FLIP simulations in Houdini will find these assets easy to use and integrate into their node networks.

At the geometry level, Section Create provides the user with a simple way to create object sections and their attributes, while Heat Source provides a straightforward method to control those attributes. Using these assets reduces the time needed to set up the model for simulation. This approach should be effective in real production environments, especially if melting effects have to be generated over many shots or on different models.

At the simulation level, Temperature Control manipulates the viscosity over time depending on changes in temperature. The user can set different values for parameters such as threshold temperature, cooling rate, and minimum and maximum viscosity to achieve different melting results. Temperature Blend can be used to control the propagation of heat through the object model based on the Distance and Blend parameters.

9.1 Limitations

Using the developed assets relies on some prior Houdini knowledge on the part of the user. A basic competency in creating FLIP simulations is assumed.

Only polygonal mesh models, more specifically triangle mesh models, can be simulated using these assets in their current state. A single mesh is also the only acceptable input for Section Create. Models with different geometry groups pose a problem when using the Boolean operator to create the split geometry, as this could lead to more sections being created than desired.

There is also no easy way to find out which sections the section ID's represent. The user must rely on VEX expressions or other operators to determine each sections ID, or to view them individually.

9.2 Future Work

The developed tool set deals with one phase transition, namely melting. It might be further developed to handle another state transition such as hardening. Phase change phenomena involving gases such as evaporation and condensation could be investigated. Another functionality that might be added to the simulation is the use of water content along with temperature and viscosity attributes.

Since each of the assets developed in this thesis work has a specific function, they could be used to create other HDAs if desired.

The assets might also be modified to simulate other materials or phenomena, such as the behavior of wet sand or mud. These assets could also be used with Houdini's Grain Solver to control viscosity, or any other attribute, in mud simulations.

REFERENCES

[1] M. Carlson, P. J. Mucha, R. B. Van Horn III, and G. Turk, "Melting and flowing," in *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 167–174, ACM, 2002.

[2] M. B. Abbott and D. R. Basco, "Computational fluid dynamics - An introduction for engineers," *NASA STI/Recon Technical Report A*, vol. 90, 1989.

[3] G. Miller and A. Pearce, "Globular dynamics: A connected particle system for animating viscous fluids," *Computers & Graphics*, vol. 13, no. 3, pp. 305–309, 1989.

[4] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," *ACM SIGGRAPH computer graphics*, vol. 21, no. 4, pp. 25–34, 1987.

[5] D. Terzopoulos, J. Platt, and K. Fleischer, "Heating and melting deformable models," *Computer Animation and Virtual Worlds*, vol. 2, no. 2, pp. 68–73, 1991.

[6] D. Tonnesen, "Modeling liquids and solids using thermal particles," in *Graphics Interface*, vol. 91, pp. 255–262, 1991.

[7] M. Desbrun, M.-P. Cani, *et al.*, "Smoothed particles: A new paradigm for animating highly deformable bodies," in *Proceedings of the Eurographics workshop on Computer animation and simulation*, vol. 96, pp. 61–76, Springer, 1996.

[8] D. Stora, P.-O. Agliati, M.-P. Cani, F. Neyret, and J.-D. Gascuel, "Animating lava flows," in *Graphics Interface (GI'99) Proceedings*, pp. 203–210, 1999.

[9] N. Foster and D. Metaxas, "Realistic animation of liquids," *Graphical models and image processing*, vol. 58, no. 5, pp. 471–483, 1996.

[10] N. Foster and D. Metaxas, "Controlling fluid animation," in *Computer Graphics International, 1997. Proceedings*, pp. 178–188, IEEE, 1997.

[11] J. Stam, "Stable fluids," in *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pp. 121–128, ACM Press/Addison-Wesley Publishing Co., 1999.

[12] N. Foster and R. Fedkiw, "Practical animation of liquids," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pp. 23–30, ACM, 2001.

[13] D. Enright, S. Marschner, and R. Fedkiw, "Animation and rendering of complex water surfaces," in *ACM Transactions on Graphics (TOG)*, vol. 21, pp. 736–744, ACM, 2002.

[14] T. G. Goktekin, A. W. Bargteil, and J. F. O'Brien, "A method for animating viscoelastic fluids," in *ACM Transactions on Graphics (TOG)*, vol. 23, pp. 463–468, ACM, 2004.

[15] N. Rasmussen, D. Enright, D. Nguyen, S. Marino, N. Sumner, W. Geiger, S. Hoon, and R. Fedkiw, "Directable photorealistic liquids," in *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 193–202, Eurographics Association, 2004.

[16] Wikipedia, "Houdini (software) — wikipedia, the free encyclopedia," 2017. [Online; accessed 17-September-2017].