Building the Snow Footprint Pipeline on *Brave*

Keith Klohn Michael O'Brien Tim Speltz Tom Wichitsripornkul
Pixar Animation Studios

Early pre-production plans for *Brave* called for Scotland to be dominated almost entirely by snow. Our task was to come up with believable snow that would look and act naturally when characters interacted with it.

To make computer generated snow feel and look real, we knew we would need to develop a sophisticated pipeline to be visually convincing and also sufficiently automated to the point of enabling a small team of artists the ability to complete a large number of shots within the our production budget.

We discuss our implementation of our snow interaction pipeline as well as address some of the technical challenges we encountered along the way.

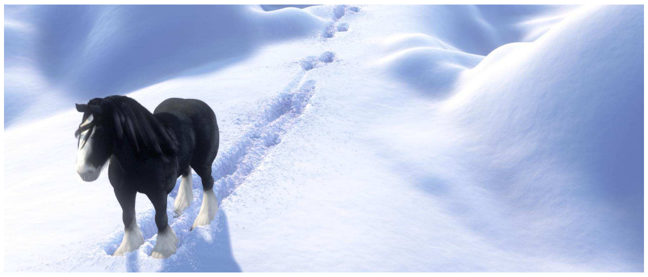

Figure 1: Character-snow interaction in *Brave*. ©Disney / Pixar. All rights reserved.

1 Snow Trenching: Adaptive Subdivision

Our previous method of character to ground interaction was generally solved using a ground shader with animated displacement maps that were carefully timed to match the character's movement. This technique worked reasonably well for shallow ground penetration but was insufficient in situations where deeper depths were needed, especially in cases where a collision mesh was needed for both particle dynamics and character cloth. As a result, we developed a series of *Surface Operators* (*SOPs*) in *SideFX's Houdini* using *RenderMan's* (*prman*) hierarchical subdivision (*hsubdiv*) edit mesh library to adaptively subdivide the polygonal mesh (snow surface) around a character's feet. Depending on the motion and size of the geometry "penetrating" the snow mesh, the points of this refined mesh were distorted to create trenches and berms based on attributes such as depth and directionality of the imprint. We then would use this distorted *hsubdiv* edit mesh as a collision surface and we also tagged specific shading attributes onto the mesh points for later use in shading. Both the original surface and the newly created *hsubdiv* edit mesh would be exported together as a single model and rendered with *RenderMan*.

2 Snow Debris: Particles and Instances

Along with trenching the polygonal mesh, we added more realism by implementing a pipeline to generate snow dust (particles) and clumps (instances). One way we achieved this realism was to base the volume, velocity, and clustering of the ejected snow material on the motion characteristics of the character that created them. For example, the faster and deeper the character penetrated the snow determined not only the region where emission would occur, but also the amount and speed of material that would be emitted. We kept track of how much volume in snow was lost from frame to frame and would use that difference in volume as the active region where particle emission would occur. We called this technique *temporal volume differencing* and it provided a useful means of achieving a more realistic look to the amount of snow and debris that would be kicked up by character walking, running, or sliding through snow.

3 Snow Shading: Primitive Variables, Point Clouds, and Ambient Occlusion

Pushing the trenching of the snow into the polygonal mesh itself gave us the basis of our look and also provided a collision mesh for other purposes. However, only shading could provide us that final level of realism we needed to visually blend the snow surface to the particles and instances that settled around it. Therefore, we widely utilized *primitive variables* (*primvars*) to store per-vertex information on the mesh that was later used by the snow surface's shader. Trenching and berming data that was previously used to displace the points of the *hsubdiv* edit surface were also used to define regions where higher frequency detail could be added with shading. This allowed us to fine tune the displacement both in and around the trench and provide a way to control the color and light attenuation of the subsurface scattering. Additionally, we used *point clouds* to define the location of snow debris and clumps that had landed on the snow surface. These *point clouds* stored information about location, size, and velocity and were read into the snow shader. From this we were able to create directional berms, divots, and holes that helped integrate the snow surface and particle data.

The last important step that pushed the sense of realism for this pipeline was the use of ambient occlusion in the lighting passes. Ambient occlusion provided a way of making particles and instances seem like they were part of the same surface and visually blended any intersections.

4 Renderfarm: Simulations and Caches

Considering the large number of snow shots that were originally in *Brave*, our artists needed a way to quickly turn around shots. Therefore a large engineering effort was put forth between our effects department and our render pipeline group (RPG) to provide a means to be able to off-load our *Houdini* tasks to the renderfarm including particle/fluid simulations and computationally expensive geometry caches. As a result, several *Render Operators* (*ROPs*) were developed for *Houdini* that enabled us to submit any arbitrary type of data to the renderfarm, either in serial or parallel. This enabled us to setup complex simulation and geometry tasks for an entire shot and became the backbone for automating the majority of the snow pipeline from start to finish.

SIGGRAPH 2012, Los Angeles, California, August 5 – 9, 2012.
ISBN 978-1-4503-1435-0/12/0008