Trolls World Tour: Desert Bling

Youxi Woo
DreamWorks Animation
Glendale, California, USA
Youxi.Woo@dreamworks.com

Doug Rizeakos
DreamWorks Animation
Glendale, California, USA
Doug.Rizeakos@dreamworks.com

Figure 1: Glitter Desert shots. Left: Cooper Pile; Center: Glitter Throwup; Right: Footprints and Wind Flurry

ABSTRACT

The fantastically realistic environment of *Trolls World Tour* took a detour into a blistering desert made purely with flecks of glitter. In order to capture the Trolls Glitter Desert experience, we blended the visual expectations of a sand-filled desert with the physical nature of flattened glitter pieces. We found the need to develop mathematical procedurals to integrate with various simulation techniques, and create a hand-drawn keyframe system to choreograph the glitter with 2d artistic control. We built custom USD software with performance increases of almost 10 times native cook times in order to work with the millions of sparkly plastic glitter instances, and integrated it with shaders for our proprietary MoonRay renderer.

CCS CONCEPTS

• **Computer Graphics** → Animation; Rendering.

KEYWORDS

sand, glitter, keyframe animation, rendering, shader, USD, hydra

ACM Reference Format:
Youxi Woo and Doug Rizeakos. 2021. Trolls World Tour: Desert Bling. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Talks (SIGGRAPH '21 Talks), August 09-13, 2021.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3450623.3464657

1 ELEMENT VARIATIONS

The Trolls desert scene included a variety of glitter sand interactions, such as artistically-driven wind flurries, rolling glitter ground cover, footsteps, glitter piles moved by characters or alien suction beams, and of course glitter throwup. Each of these were quite distinct in behavior, requiring distinct simulation types or motion treatment, but all components were required to be visually recognized as the same material interacting in the same world.

1.1 Wind Flurries

Across the high-wind desert, scores of squalls carried glitter in distinctly Trolls-looking patterns. The flurries required strict choreography and art direction, both individually and by the dozens. So we developed a system allowing hand-drawn flurry paths, much like a 2d keyframing tool: stepping through shots, drawing the shapes of flurries from camera space onto the environment. Alternatively, in shots with hundreds of flurries, procedural mid- and background-flurry curves were created. The system pulled curls and lips from a library to append to the ends of our curves, based on the direction and topology of the ground, and then conformed the curves' speed and maneuvering to the physical characteristics of the environment. We could then playback or scrub an animated onion-skinned representation of the shot to visualize the timing and composition. Curves were injected into a particle simulation with mathematical "glitter flutter" equations to supplement the particle motion (Fig. 2).

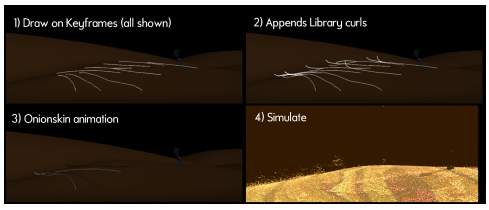

Figure 2: Keyframing flurry paths

1.2 Procedural Footsteps

As the characters walked through this non-spherical, flaky material, we needed a way to hit a specific look of a footprint, while having each glitter fleck facing more aesthetically pleasing angles after settling to its final rest position. We reversed the workflow by modeling the final footprint and glitter angles and calculated how they

SIGGRAPH '21 Talks, August 09-13, 2021, Virtual Event, USA

ACM ISBN 978-1-4503-8373-8/21/08.
https://doi.org/10.1145/3450623.3464657

would get there. The final positions were generated by molding a poly to the desired footprint shape, packing it with glitter geometry, then applying some normals-math to face them generally outwards. To find a starting position, we projected a grid from above the final model and, like a scanline render, we traversed the grid, grouping glitter instances into each grid space. Then we stacked the glitter in each grid group beneath one another under the ground. From this start position, we generated spaced out paths for each glitter piece to traverse to their modeled end position. The interpolation along the path was triggered by the character's relative position as they walked.

1.3 Piles with Motion Levels of Detail

We wanted the feel of dry, coarse desert sand, but with the physics of flat, round plastic glitter pieces. Though these two principles contradict in that glitter, on the whole, doesn't move, stack or shape itself like sand. To compensate, we combined a workflow using sand/grain simulations, RBD simulations, procedural motion (such as with the footsteps), volumetric and particle sims, static set dressing, and calculated movements integrating between steps. Each step created a layer representing a Level of Detail for the motion of the glitter (Fig. 3), with as much glitter pushed to the lower, more controlled layers as possible. At the bottom was the static glitter that was modeled. Above that we applied a grain sim that would shape the overall motion to give a macro-feel of sand in the desert. Grains are naturally spherical, where glitter is flat, so we applied a post-sim movement integration process, where grains beginning to move or settling into place, were subject to a smoothing operation to move the grains more like disks. This included more sliding, rather than rotating, more packing to keep the distance of a flat disk, and rotating into a position that was flatter towards the normal when the grain settled. Above that, we created a procedural motion level, to specifically place glitter into positions that looked nice, on top of the grains. Then, we added a thin RBD simulation layer on top, like a skin, so that we could show flat glitter interactions and give the impression it was happening all beneath that skin. Lastly, we added volume advected particle sims to depict the light, top layer of glitter that trailed the main motion and fluttered softly through the air.

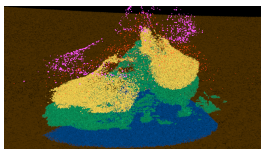

Figure 3: Layers of motion detail

2 USD WORKFLOW

All the various techniques flowed through a consistent set of IO tools based on Pixar's USD format [ElKoura et al. 2019]. Leveraging its Point Instancer schema, we augmented the plugins and wrote custom versions to tune for performance and artist usability. Used in-app, USD's point instancer draw times can be slowed by behind-the-scenes prep and conversion. We developed a viewport plugin to circumvent native behavior and use hydra to draw instead [Spitzak 2018]. Playback speed for 100 frames of 2.3 million instances in loading a single object with our custom plugin took 7s, but without it, took 1m20s. The end-to-end points workflow developed for working with individual instances was an even higher performance boost. With 1 million points/instances playing back for 24 frames, our points workflow had a processing time of .06s and a draw time of .5s, while native instances had a cook time of .5s and a draw time of 40s. Choosing instance sources was simplified by our instancer offering relative probability-based sourcing - artists were able to use sliders to quickly dial in ratios of source objects. Our exporter wrote out an improved 1 million instances at about 1.6s per frame, compared to the native packed prim 3.5s per frame.

3 REAL WORLD OF TROLLS

Conveying the scale of the petite Trolls characters was a priority and the glitter desert needed to maintain that illusion. We added a sense of toy-like scale by animating a blended 1 to 2-frame step motion for the glitter pile interactions. Much like water simulations that separate behavior when the particles are within the body of the water versus spray elements, our glitter instances were marked as being in the pile, fluttering above the main pile, or crawling across the surface of the ground. While they were above or crawling, we moved them every one frame so their motion and flutter could be tracked well. When the instances touched down on the main pile, they moved every two frames to maintain that playful stepped motion. The determination of being in the pile was made based on the distance from our ground layer, relative speed and the force each glitter fleck exerted on the pile of flecks. Our world was extended by the *MoonRay GlitterFlake* material shader [Cegielski 2017], where every part of the world that was not instanced with glitter geometry, would be textured in glitter. The shader placed glitter-like flakes on top of the geometry as a displacement and color map. It used a computationally inexpensive microfacet model to help reduce render times.

4 CONCLUSION

Technology can create exceedingly precise models and design the most physically accurate simulations, but the art of animation continues to demand off-physical motion. Aiming to walk the line between realism and imagination, our desert set used familiar yet custom and hand-crafted techniques, to create a make-believe world of accurately flying and tumbling flakes. They gathered by the millions, loading 10 times faster and rendering sooner - under, around and out from our beloved Trolls characters, enhanced by the power of our USD-based instancing procedures and custom MoonRay rendering techniques. Tens of millions of glitter flecks lined the Trolls world, answering the question: what exactly does it feel like to be in a desert made of glitter?

REFERENCES

Scott Cegielski. 2017. GlitterFlake Material. Internal, DreamWorks Moonshine.
George ElKoura, Sebastian Grassia, Sunya Boonyatera, Alex Mohr, Pol Jeremias-Vila, and Matt Kuruc. 2019. A deep dive into universal scene description and hydra. *ACM SIGGRAPH 2019, Courses* 1 (2019), 1–48. https://doi.org/10.1145/3305366.3328033
Bill Spitzak. 2018. Stats and Analysis. Retrieved May 26, 2021 from https://github.com/PixarAnimationStudios/USD/pull/723