# Real-Time Crowd Visualization in Point-Cached Pipelines

Jeremy Cowles
Pixar Animation Studios

Takahito Tejima
Pixar Animation Studios

David Yu
Pixar Animation Studios

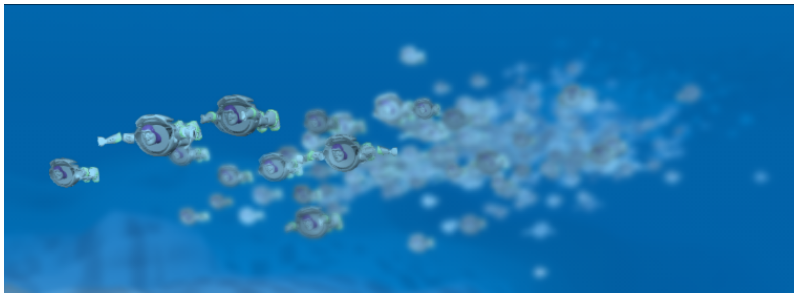

**Figure 1:** *A real-time swarm of Buzz Lightyears.*

## 1 Point Cached Crowds

In computer animation, a crowd of characters can create a dramatic visual impact and make an expansive set feel like it's teeming with life. To orchestrate this large scale animation, technical directors use procedural tools to author animation en masse. The character animation is created by animators for an array of background models, such as walk cycles, action transitions, and reactions. Despite being modeled and rigged specifically as a background character, deformation of crowd characters is still driven by high fidelity rigs that are too expensive to execute in real-time, even for small scale crowds. Background characters are designed to have parametric controls to create visual diversity, which require heavier mesh geometry to support such variation. The character structure must support standard shading and lighting workflows, which result in characters that include separate object primitives for skin, hair, and clothing, and frequently even more fine grained objects for eyes, teeth, fingernails, etc.

As modern production pipelines shift to point-cache-based solutions, such as Alembic and Universal Scene Description (USD), real-time preview of crowd animation becomes constrained by CPU driver overhead, disk IO, and CPU-GPU memory bandwidth: this shifts the typical crowd visualization problem from one of GPU rig execution, to a disk IO and memory optimization problem.

## 2 Simplification and Point Instancing

Once character animation is created, an automated process exports a point cache preserving the original deformed mesh, object hierarchy, transforms and shader structure. The point cache is then used to generate a simplified version of the character with fused, decimated meshes and a greatly simplified object hierarchy. All versions of the model co-exist and are expressed as a live, hot-swappable structure in the scene graph. This structure enables both high-performance and high-fidelity preview. The new switchable representation is then instanced into a point cloud, which is also expressed in the scene graph.

Our contributions over prior work include: 1) a novel GPU dispatch structure that leverages modern hardware features to enable parallel culling at individual instance granularity; 2) a GPU indirection table that allows for unordered instances, enabling maximal hardware instancing via dynamic remapping; 3) an animation cache that remaps the last frame's hardware prototypes to the next frame's required pose, when possible.

## 3 Dispatch Structure

When dealing with instanced geometry, frustum-bounding box intersection tests are excessively expensive to compute on the CPU due to the large number of instances and the resulting computation requred to construct and test bounds for each instance. Given that this problem is highly parallel in nature, we have developed a dispatch strategy to leverage modern GPU hardware to process instances in parallel. The strategy includes indirect drawing, transform feedback, atomics and coherent CPU-GPU memory to allow for both efficient CPU dispatch (extremely low driver overhead) and GPU frustum culling of individual instances.

## 4 Instancing Indirection

The GPU data structures are organized such that each primitive can express instance offsets in the point cloud. The number of instance offsets becomes the number of hardware instances and the aggregate instance offset table is accessed from the shader to discover final instance data in the GPU representation of the point cloud. This indirection table enables efficient per-instance GPU culling and dynamic aggregation of prototypes.

## 5 Prototype Animation Caching

As time offsets of individual instances vary over time, so too does the number of master prototypes, where each prototype can draw one or more hardware instances. Previously uploaded point cache data is reclaimed on each frame as a GPU-local animation cache by dynamically remapping instances to prototype primitives, dramatically reducing disk IO, while still only caching a single frame of animation per prototype.

With the strategies outlined, we obtain real-time frame rates, while only caching a single frame of animation data per prototype. In one case, an animated crowd of 5000 characters with 408 unique time offsets were visualized at 16ms per frame (60 FPS) on a Quadro 4000 GPU.


SIGGRAPH 2015 Talks, August 09 – 13, 2015, Los Angeles, CA.
ACM 978-1-4503-3636-9/15/08.
http://dx.doi.org/10.1145/2775280.2792586