Up Close with Simulated Crowds

Mark Adams
Blue Sky Studios
marka@blueskystudios.com

Justin Bisceglio
Blue Sky Studios
justinb@blueskystudios.com

Figure 1: A selection of crowd shots from: *Epic* (2013), *The Peanuts Movie* (2015), and *Ferdinand* (2017)

ABSTRACT

We discuss advancing the fine detail of deforming hero-quality simulated crowd agents in animated feature film production. To support character animation that is suitably framed arbitrarily close to camera, our approach uses a novel deformation system that combines simulation and hero-quality custom animation. Level-of-detail optimizations are handled at render time, and artists are only tasked with the design of a single high-quality resolution for each character asset. Key optimizations in our rig structures are outlined as they are fundamental to scalability, permitting our crowds to look good while numbering in the millions.

CCS CONCEPTS

• **Applied computing** → **Arts and humanities**; • **Computer methodologies** → **Computer graphic**;

KEYWORDS

ACM proceedings, computer animation, crowds, simulation

ACM Reference Format:
Mark Adams and Justin Bisceglio. 2018. Up Close with Simulated Crowds. In *Proceedings of SIGGRAPH '18 Talks.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3214745.3214756

1 INTRODUCTION

At the start of our production *Epic* in 2011, we were tasked with developing a new crowd system that could support simulated crowds with secondary animation including terrain adaptation, rag doll effects, as well as direct manipulation. Another requirement was that crowd agents need to seamlessly blend in when juxtaposed with hero animation. For these reasons, we set our goal on retaining as much of the hero deformation rig, model, and materials as possible. However, it was also necessary to avoid the bottleneck caused by baking out crowd geometry and subsequently importing this data at render time. To address this requirement, we felt it would be most practical to evaluate crowd rigs at render time. (This is a capability of our in-house renderer, *cgiStudio*.) For each show, new features are continually added, improving the quality and sophistication of the animation. Here's how it works.

SIGGRAPH '18 Talks, August 12-16, 2018, Vancouver, BC, Canada

ACM ISBN 978-1-4503-5820-0/18/08.
https://doi.org/10.1145/3214745.3214756

2 CREATING CROWD ASSETS

We begin with an unaltered hero asset. The topology stays the same, as do all director approved rendering attributes such as materials and hair. The main rig is then simplified by removing any joint or transform not directly used for deformation. The remaining joints are then structured into a fully parented forward-kinematics hierarchy. From here, the skeletal system is partitioned into two sets: 1.) a set that will be used by simulation software (usually skeletal joints in the body of a character) and 2.) a set of joints that control finer details that would otherwise tax simulation (like animation in the face and hands). This keeps the simulation tractable. The joints in the crowd asset will be snapped or retargeted to joints in the hero version to capture cycle animations. To maintain a consistent look, we replicate as much of the rig as possible using node types supported in the evaluation system at render time. We support blend shapes, skin clusters, wrap deformers, free-form-deformations (FFDs), joint clusters, subdivision smoothing, and constraints.

We use Autodesk's *Maya*® for the design and animation of our assets, and export our crowd geometry and rigs to a *cgiStudio* compatible format. During export, further optimizations are made in the rig. Hierarchical transformations are concatenated. Complex node networks are consolidated. In particular, f-curve and logic networks associated with driving a blend shape attribute are folded into the blend shape node for faster evaluation. Non-keyed attributes in the rig are cached out, and treated as static data when evaluating rigs with simulation data.

We also transform rig dependency graphs into evaluation stacks. To do that, we build a relational database that records how to associate these stacks with the shapes that they deform. Additionally, we record any possible dependencies between the nodes in their stacks so that nodes with multiple outputs are computed only once.

3 SIMULATION DATA ORGANIZATION

Our simulation data is stored as two files. The first file contains a meta-data scene description that includes information concerning the number of agents spawned from each asset, any global transformations that may be applied to an agent, as well as any asset variations that are applied to the agent, such as wardrobe selection and material variation. Essentially, this data remains constant over the entire frame range of a shot. We support a proxy display of the crowd in *Maya*® with a user interface that enables post-simulation editing of this header file.

The other file contains a database of skeletal joint animations for each agent per frame of the shot. The database is queried by indexing according to asset type, agent instance number, and frame number.

4 THE EVALUATION PROCESS

With simulation data loaded, we use the following algorithm to evaluate each shape of an agent before rendering.

Algorithm 1 Evaluate deformed meshes for all crowd agents

```
for i < size (crowd assets) do
  use database to connect shapes with their deformation stack
  for j < size (instances of asset) do
    for k ∈ {surfaces in each asset} do
      for l < size (motion blur time samples) do
        V ← shape vertices for surface(k)
        N ← deformation stack for surface(k)
        for m < size (N) do
          /*process stack node*/
          V = evaluate (N.pop(), V)
        end for
        shape vertices for surface(k) ← V
      end for
    end for
  end for
end for
```

5 AUGMENTING AND OVERRIDING SIMULATION RESULTS

We use blend shapes and skin clusters to modify simulation results and add detail that would otherwise be too costly to simulate. We refer to three different types of overrides as baked, correctives, and post-animation.

Baked blend shapes are inserted at the top of the asset's deformation stack by a scripted modification of the stack at render time. Typically, these blend shapes contain offset deltas and are used to append simulated garment detail into the asset's animation.

Correctives are more traditional style blend shapes. They are defined in the asset before exporting the rig. However, the keyframed weights used in these blend shapes are not simulated. They are read as cycle data from an independent file also at render time.

Post-animation refers to fine detail animation through localized skin clusters for highly articulated regions of the character, often in the face and hands. Here, joint transformations are stored relative to some parent transformation contained in the simulation data.

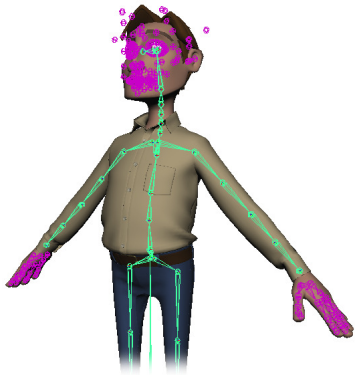

Figure 2: An example partitioning of skeletal joints. Simulation joints are highlighted in green, and non-simulated joints are highlighted in magenta.

Post-animation skin clusters and corrective blend shapes in turn drive complex rig components in the head and hands adding nuanced deformations, bringing the agents to life and looking natural when rendered up close. The head rig typically consists of several FFDs. These FFDs are skinned and overlap one another. Each has it's own weight map and the results are mixed together in a blend shape. Post-animation skin clusters also support animated bind poses. These permit localized pose-specific deformations which help sculpt deformations around a character's mouth and eyes.

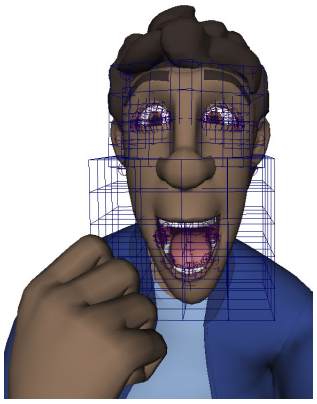

Figure 3: FFDs in action on a crowd asset

6 RENDERING

Once we have computed the deformed shapes for a crowd agent, we may render the geometry directly as a subdivision surface, or we may use the geometry to influence a coarse voxelization of the undeformed agent. Here, the geometry acts as a wrap deformation while creating a voxelization cache. A cached voxelization reduces the memory overhead for rendering crowds, but it is limited to crowds in the distant background.

cgiStudio supports procedural textures and tracking to subdivision patches directly, which allows us to avoid the storage overhead of texture maps and micro-facet geometry for accelerated ray-tracing. As a result, our renderer scales well as scene data increases, permitting us to render a large number of uniquely deforming crowd agents as subdivision surfaces in the foreground.