Populating the Crowds in *Ferdinand*

Greg Mourino, Mason Evans, Kevin Edzenga, Svetla Cavaleri, Mark Adams, Justin Bisceglio
Blue Sky Studios *

ABSTRACT

With the help of new tools, we streamlined our review and render processes for crowds to triple our shot count on our latest show, *Ferdinand*. At the same time, we integrated some novel approaches to complex deformation features for cloth and facial animation, which elevated the quality of our crowd animations.

CCS CONCEPTS

•Computing methodologies →Animation; Procedural animation; Physical simulation; *Motion processing; Rendering;*

KEYWORDS

Crowds, Blue Sky Studios, Ferdinand

ACM Reference format:
Greg Mourino, Mason Evans, Kevin Edzenga, Svetla Cavaleri, Mark Adams, Justin Bisceglio. 2017. Populating the Crowds in *Ferdinand*. In *Proceedings of SIGGRAPH '17 Talks, Los Angeles, CA, USA, July 30 - August 03, 2017,* 2 pages.
DOI: http://dx.doi.org/10.1145/3084363.3085055

Figure 1: ***©2017 Twentieth Century Fox Film Corporation. Not for sale or duplication***

1 RE:WORKING THE CROWD

Optimizing our workflow practices was crucial in tackling the shear volume of shots within the production schedule. We shifted our entry point to earlier in the pipeline, introducing a rough pass that could be refined in final shotwork. We developed render tools that overhauled our review process. For new compound agent types, we made a framework that managed the dependencies between combinations of cars, drivers and passengers.

* e-mail: {gregm,masone,kedzenga,sivanova,marka,justinb}@blueskystudios.com

SIGGRAPH '17 Talks, Los Angeles, CA, USA
 978-1-4503-5008-2/17/07.
DOI: http://dx.doi.org/10.1145/3084363.3085055

1.1 Crowd Control

Through custom Houdini nodes, we created a flexible framework that allowed us to export the crowd simulation data and re-import that data in various contexts. The crowds artist could use that data for general placement of scattered points, load the exact assets from that placement, or even load the entire simulation. Once the data was loaded, the artists could selectively re-simulate, change assets or add additional animation on different agents as needed. We implemented this into our new "Blocking" phase, where the crowd artist worked closely with camera and staging artists to block crowds for entire sequences. This workflow allowed data from the planning phases to be reused in final shotwork, which reduced initial preparation and helped increase continuity between shots.

1.2 Bringing People Together

Master Blaster is our new toolset for automating the creation of high quality review images with *CGI Studio*®. While coarse body movement could be observed in Houdini or Maya through our simulation rigs, highly detailed motion such as the hands, face and garment animation could only be viewed when rendered. Upstream departments created environment and hero animation playblast images for review and approval, and the Master Blaster would collect and combine those sequences with our detailed crowd renders.

The challenge in this process was that Maya iff files, which upstream departments rely upon, use a depth channel that contained parallel depth information. Standard rendered images use divergence depth, and the two formats are not directly compatible. For each rendering iteration, Master Blaster created a separate lens distortion pass, where a plane was rendered a fixed distance from the camera. Using simple trigonometric equations combined with the lens distortion data, we formulated a mapping between these image formats. This allowed us to un-distort both images even with moving cameras.

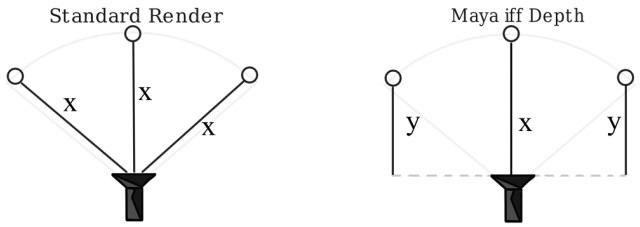

Figure 2: ***Distance from camera position versus camera plane***

This tool also served as a user friendly GUI for launching renders and making it more accessible for new artists to use *CGI Studio*®. As shotwork progressed, we used it to send our image sequences to other teams, facilitating inter-department collaboration and freeing resources to address artistic notes.

1.3 Crowd Cars

For the first time, our team simulated vehicles. Rather than combining the car, driver, and passengers into fixed assets, we opted to build a system where the car agent was dynamically populated with human variations. We used joints placed at the seats' positions to drive the human agents' global location. Custom solvers and an extension of Houdini's IK system allowed us to use the car's velocity to steer the front wheels, which in turn controlled the driver's hand placement on the steering wheel. Effectively, the car's direction controlled the driver's animation. This flexible, artist-directable solution was indispensable when agents were close to camera.

Figure 3: ***Car and passengers: ©2017 Twentieth Century Fox Film Corporation. Not for sale or duplication***

2 STANDING OUT IN A CROWD

On every film we strive to bring our crowd rigs closer to the fidelity of a hero rig while maintaining the requirements of the simulation software. Depicting human characters with detailed clothing and facial articulation made this especially important on *Ferdinand*. These extra levels of detail allowed us to freely cast agents where needed, without concern for camera proximity or contrast with hero characters.

2.1 Tailored to a Crowd

A goal for *Ferdinand* was adding simulated cloth into the crowds pipeline while avoiding the need to simulate garments on individual agents per shot. To achieve this goal, we turned to an in-house sculpt solver to create a two-stage workflow capturing the deformation of the garment simulation in a per-frame blendshape. Through this we attained the ability to deform the garments through the skeletal rig to acheive secondary animation including terrain adaptation, "look at" overrides, and transitioning between cycles, while keeping the initial garment simulation intact.

During cycle processing, we imported both the crowd asset with rigged garments and animated asset with simulated garments. The solver was then set as the first node in the deformation stack of a selected crowd garment. At each frame of the cycle, we pointed the solver to the corresponding simulated garment as a target. The solver then created a difference map, taking into account the entire deformation stack to reach the target shape. The result was a new blendshape node at the beginning of the stack. When the character returned to that pose and the blendshape was enabled, the rigged garment matched the target simulated garment. Next, we created a second blendshape between the solvers output and the rigged garments' bind pose. The final result was a per-frame blendshape for each garment, which gave us the ease of "baking out" the cloth simulation, but more importantly the freedom to deform the garments through the skeletal rig within a limited, but sufficient range for secondary animation without the need to re-simulate the garments on that unique motion.

2.2 A Face in the Crowd

To improve the quality of facial animation in our crowd rigs, we added support for free-form deformation (FFD) or lattices. A typical setup consisted of five lattices, two around each eye and one around the mouth. Each lattice had the same input geometries, and sometimes their output drove a blend shape channel. This permited weighted control for each lattice on a per vertex basis. Our final crowd deformations are computed at render time, so we needed to replicate this FFD setup in our rendering system *CGI Studio*®.

Besides supporting lattices, we also needed to introduce several enhancements to our rig evaluation method. Previously, rig evaluation was limited to deformation functions that operate on a single input and produce a single output. Deformers were then chained together to form a rig. During export of rigs from Maya to *CGI Studio*®, these simple chains were preprocessed and optimized in the form of a stack.

With the FFD setup, we no longer have simple chains of deformers. To maintain the use of a stack while supporting a more generalized rigging setup, we coupled our stack with additional data tables. Using tables to store historical information allowed us to keep track of what nodes were already processed, and helped avoid redundant calculations for nodes with multiple outputs. It provided a means to cache vertex positions and optimize cases when nodes operate on multiple shapes. Because we were already using data tables to store key-frame data for the simulation of each agent, enhancing rig evaluations with additional tables turned out to be a natural extension that was easy to implement.

3 STRENGTH IN NUMBERS

More than 50% of the crowd shotwork in *Ferdinand* took place at *Plaza de Toros de Las Ventas*, Madrid's famous arena holding more than 15,000 spectators. We implemented dynamic camera culling, layering, and voxelization so that downstream departments could render the stadium crowd in a timely manner.

First, we performed a dynamic, per-frame, camera culling where any agent outside a buffered camera frustum was removed before render time. Next, we divided the crowd into render layers to be processed in parallel. These layers were procedurally determined by the agents' distance from camera. For moving cameras, the procedure processed each frame of the shot and locked each agent into the layer it was sorted into when it first appeared on camera. This technique of accumulation insured that the agents would not change render layers during the shot, avoiding problems in compositing. When voxelizing the crowd, these layers were further subdivided along a camera space plane. This gave us a smaller RAM footprint while maintaining the accumulated layering and high quality of individual character appearance.