Building Crowds Of Unique Characters

David Prescott, Darren Hendler, Dave Hodgins, David Blumenfeld
Spencer Alexander, Leonardo Szew, Adam Sidwell, Manny Wong
Erick Miller
Digital Domain

Introduction

Constructing realistic shots with hundreds, or even thousands of completely unique, fully animated, deforming, and realistically shaded characters poses an entirely different challenge than the traditional character pipeline. Workflow, Level of Detail (LOD), and current hardware/software limitations require unique solutions to achieve the goal of a usable crowd animation and rendering suite. Although the system described in this sketch was initially developed and used specifically for the blockbuster effects film, The Day After Tomorrow, at the time of writing this, it has also been successfully used in 3 feature length films, and one commercial.

Publishing Procedurally Created Characters

Generating a crowd of characters begins with the "Character Creation Kit". Leveraging concepts learned from The Grinch, we created a modular system for slider and enumerated data type based character generation. The users change modifiable attributes such as: type, sex, height, weight, race, clothing, neck and leg length, textures, etc. Once users are happy with the character they have generated, they choose to publish the character into a defined character database. All of the data necessary to procedurally re-create that character is saved as separate swappable components of a published character.

Unique Characters, Generic Animation

The characters that are published are seemingly unique characters. For example, characters could span from being short fat women in skirts, to broad shouldered bipedal robots. Although extreme variation is possible, the crowd system is built to apply all the same motion from the library onto all possible character varieties. This is achieved through an intrinsic retargeting system built into the character definition, which uses a simple mix of additive, multiplicative and driven weighted interpolation to modify the motion as it is applied to varying degrees of different characters. The animation is stored in a non-linear format, and therefore can be very simply applied and filtered through any computational graph prior to its final output.

Multiple Levels Of Characters

The base character definition is encapsulated into a very small set of fast DAG nodes. Specifically there are two matrix transforms, a curve path and a node that contains all of the inputs for a character's animation. A small node network exists to re-map the character's forward z-axis translation into the parameter space of a NURBS curve, which is then normalized by the curve length, and multiplied by a constant based on the leg length. The character then has 3 LODs that can be applied for viewing or rendering in varying details. An additional system was added for using mapped camera aligned planes for more distant low LOD shots.

Populating Scenes

The act of populating a crowd scene is a very simple process of painting them into the scene. Painting crowds of characters is both a fun and novel approach to populating, as it is fast and easy to change density, and area, while still maintaining control over placement. Characters are chosen randomly and slightly jittered to give a random feeling to their layout. Additional population options are available, such as placing directly onto points in the scene, or along predefined velocity paths.

Animating Crowd Motion

Crowd motion, as previously explained is encapsulated as nonlinear clips. The most vital aspect of animation application and editing in a crowd scenario is the ability to reference and instance data. A system is therefore built for referencing the clips from disk, and instancing the curves onto the characters so that no "real" animation data ever exists in the scenes. The animation can then be updated, and the crowd scene automatically updates. Each additional level of detail also has the unique ability to animate on top of, or partially over the motion clips, and the system saves and restores this as data if the LOD is loaded/unloaded.

Dynamic Aspects

An interesting thing about dynamically derived motion in a real production environment is that it must remain editable and capable of art direction at all times. So, the system was built to allow placement of characters, and dynamic application of motion on either the front-end of the system, or post – after animation has already been applied. The possible dynamic inputs to the characters consist of three separate components.

1. Initial curve path generation, with per frame velocity float data (i.e., seek and follow, avoid, boids, etc.)
2. Post "pass" based character collision avoidance for fixing minor collisions in already animated crowds.
3. Attribute based clip editing of associated clips using triggered attributes that represent events for predetermined associated motions.

Rendering Scenes

Rendering crowds of characters without splitting them up into separate scenes poses an interesting challenge. Luckily, Prman's procedural call gives us exactly the behavior we need. The crowd characters are sent to the farm to be RIB archived, distributed by number of characters per CPU, with user defined granularities, and then re-ordered into custom RIB files sorted by frames in time. A C++ RIB parsing class was constructed and written into a procedural run program, which used gzip, diff, patch, and various RIB keyword based logic around I/O streams for parsing in non-deforming transforms and surface calls from a master rib into the archived ribs. The final 3d scene used for lighting is a fast interactive file containing all the characters as moving cubes, which when rendered, output the procedural calls into the rib stream to pass the characters to Renderman (as opposed to passing the cubes). The source RIB for any particular character can be updated and all the surface calls and non-deforming geometry get swapped in at run time by the procedural RIB generator program. The result is an editable rendering pipeline that uses very little disk space or system resources.