1 Abstract

Covering large savannas of Africa with grass created unique challenges for set-dressing, character interaction, and rendering for DreamWorks Animation's "Madagascar: Escape 2 Africa". In many sequences, grass is our most important set dressing element. We enabled layout artists to set-dress grass by placing large geometric shapes. We wrote tools that enable our surfacing department to fill these shapes with stylized grass. We expanded our fur collision-avoidance software, *smoosh*, to enable thousands of characters to interact with grass. To keep computation cost manageable, each colliding strand is assigned a simple collision response animation rather than using a dynamic strand simulation. We sped up turnaround time and deployment using arbitrary fur geometric regions, geometry simplification, significant parallelization, and "click button" simplicity. To improve grass anti-aliasing in our renderer, we apply transparency. We made improvements to our deferred shading renderer in order to accommodate large volumes of transparent geometry.

2 Layout

Traditionally, our surfacing department develops the placement and style of grass on our ground environments. This approach provides little feedback for our set-dressing and animation departments. However, with grass taking on a greater role in this film by involving large, stylized expanses and sometimes careful character interaction, we developed a system, *sod*, to address these issues.

![Figure 1: A sod setup in Madagascar 2](image)

*Sod* encompasses a suite of tools which allows the set-dressing department to quickly place simple geometric shapes on the ground. A layout artist draws curves, lofts them into slabs, and conforms them to the ground. In addition to the ease of layout of these shapes, this geometry provides benefits to other departments needing quick feedback as to the placement and location of the grass. The surfacing department then converts these shapes into data for our fur shader.

3 Simulation

Once placed, the grass is then simulated for any character and object interaction. We improved our fur interaction system, *smoosh*, to handle both large expanses of grass and potentially thousands of interaction objects. Upon collision, each fur strand in the simulation set is assigned a direction and moved along this vector using a pre-defined heuristic until it is no longer colliding with the object. *Smoosh* applies different collision response animations based on the degree it collides. A gentle brush might result in an oscillating response whereas a crushed blade of grass will never fully recover. The fur shader applies the *smoosh* data to nearby strands using a simple closest point algorithm. In addition, we updated the system to work on any arbitrary fur region to allow for decreased geometry generation time, massive parallelization and a significant improvements to artist iteration time by using either a procedurally generated "best fit" or artist set region curve. We also developed a simplified GUI tool to assist in the setup and execution of the simulation and integration of the data back into the production shot.

![Figure 2: An example smoosh simulation](image)

4 Rendering

The grassy savannas are soft, dense and endless. As the grass recede into horizon, its blades become thinner than the finite sampling precision of the renderer, causing temporal aliasing and buzzing artifacts. To eliminate buzzing, we automatically scale the grasses to be at least a screen pixel wide and adjust the transparency to maintain the overall opacity and luminance of the image.

Transparency is a classic technique used for fur antialiasing, but the grassy savannas contain millions and millions of curves; making this technique prohibitively expensive for most film renderers. To improve the efficiency of transparent curve rendering, we made several extensions to our renderer. We added a high resolution zbuffer to maximize the occlusion effects of opaque objects which are always rendered before transparent objects; we extended the depth buffer storage to support a z sorted list of transparent micropolygons; we track the accumulated opacity and maximum z value of each pixel in the depth buffer to reduce or even eliminate the rendering cost of hidden grass; and finally we shade the micropolygons in each pixel front to back and store the alpha blended color in the final pixel.