Prep and Landing
Set'm and Forget'm: A Motion Graphics Pipeline for Effects

Ian J. Coony Brian Silva Bruce Wright Andrew Kinney
Walt Disney Animation Studios

figure 1: (a) 2D Loop (b) Gingerbread Man (c) Programmed Light Displays (d) Button Animation (e) Monitor Graphics

1. Introduction

For the creation of brilliant light displays, flickering control tower buttons and vibrant computer monitors, the goal was to build a motion graphics pipeline capable of running nearly unattended. Effects work would have to feed into the production front-end and allow for easy artistic intervention downstream. Once in place, the design allowed our department to work more efficiently to save time for other effects related challenges.

2. Front loading

Effects resources were devoted early on to help determine a way to deal with the volume of motion graphic shots. Graphics for monitors 1(e), the GBM 1(b) and other gadgets were animated and composited as sequences of 2D loops 1(a). The aim was to achieve front-end director approval of all loops prior to being shown rendered in a shot. Early approval and front loading our efforts allowed blocks of shots to bypass the effects department altogether and head straight into lighting.

3. Event Handling

Production efficiencies were gained by carefully considering each department's needs for an element and scripting in automated event handling. For example, hardware accelerated textures may be fantastic to visualize and adjust in one department, while causing untold headaches in others. Sometimes just viewing them in a scene file may slow animation scrubbing to a crawl. A simple solution – model elements were published with embedded scripts that could query the current department and automatically turn off hardware textured graphics for that element if needed. Simple utility functions were added as embedded script nodes should animators want to view them in scene for reference. Attributes were also procedurally added to each model and acted as pointers to the graphic loops on disk. These could easily be updated in the pipeline external to the 3D package using simple text meta-data. Other attributes added allowed control of loop start and end frame, time offset and scale, reverse loop, bounce and loop rock modes. Hardware and software shader expressions would then determine assignments based on these preset attributes. Since two identical sets of these loops were composited (Tiff and Ptex**) as part of our front loading process, hardware shaders in Maya would use the tiffs as cached texture maps for interactive feedback. 1(b,d). Once the shot reached lighting, the same attributes would allow Renderman to software render using the corresponding Ptex file in the shader 1(b).

A "Button Script" interface was created to allow procedural animation of the control room button glows and could also be used to choreograph strands of lights. Simply pick each light in their firing order and run the script. Expressions drove stored attributes for each light geometry, defining intensity, phase, offset, and fcurve type over time, (sawtooth, static, binary, sine, weighted tangents, etc.). This setup allowed for a large variety of light animation on the various button panels. Lights could simply switch on and off in moving marquee fashion or be displayed as sophisticated glowing, flickering pulses 1(c,d,e). This could all be visualized by front-end departments as animated material shaders played back in the scene – no rendering required. Director approvals could now happen on hardware playback alone. This allowed these elements to be published along with the layout master sets for downstream travel to the character, look and lighting departments. Attributes were easily adjusted as needed all the way up to the final lighting render. The look department developed shader expressions that read the attributes stored on each piece of geometry and allowed them to software render accordingly - very similar to the way the GBM and monitor graphics setups were handled.

4. Fix On Fail

Element progression through the pipeline could be verified by watching daily "auto renders". These ambient lit renders allowed quick feedback should one of our elements go awry in a shot. Fixes were implemented in the pipeline by publishing a snippet of scripted meta-data. Embedded attributes were adjusted by this script to offset an animation loop or change a distracting button sequence without ever having to crack open a scene file. "Fix On Fail" required a certain leap of faith, but turned out to be a time saver, as shots didn't need to come through our department in the majority of cases.

Characters would occasionally need to interact directly with on screen buttons 1(b) (GBM). By visualizing hardware textures in a scene, character animators could see exactly where to place a finger on a button. In some cases, the loops needed to be modified if a preset loop didn't fit the bill. By building the scene in the effects department with the character animation as reference, it was easy to update loops of screen controls in the composite. These updated loops were then reloaded as textures in hardware and rechecked for registration to the character animation.

4. Conclusion

Effects pipeline workflows supporting the combined power and flexibility of departments working in parallel, proved key to achieving high quality and consistent efficiency for the production of *Prep and Landing*. Future extension of this may be possible by even tighter integration and control of textures for display in up and downstream departments. Ideally, one multi-resolution texture format could serve as both hardware texture preview and final software render. This may be accomplished via plug-in with the ability to quickly convert the .ptx format to a commonly viewable native hardware format (e.g., tiff) then downsample resolution on demand for quick interactivity.

**Ptex: Per-Face Texture Mapping for Production Rendering
Burley, Lacewell-Eurographics Symposium on Rendering 2008