Virtual Bones: animating skinned meshes on GPU

Vladimir V. Lopatin*
Senior VFX Artist, Guerrilla-Games

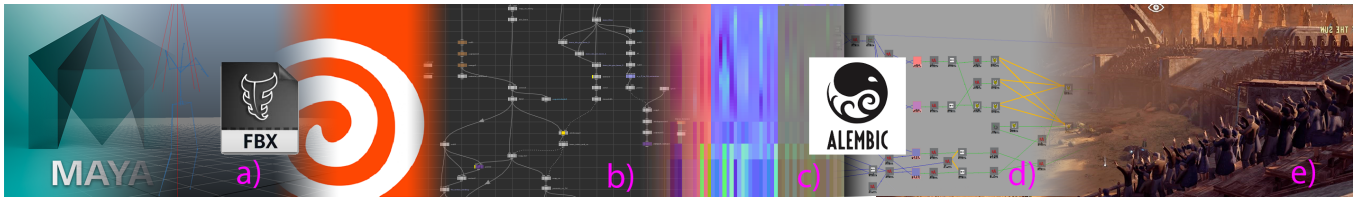

Figure 1: *Character rig in Maya (a), Houdini network (b), Virtual Bones texture (c), Maya shader network (d), multiple 3D impostors (e).*

1 Introduction

During the production of open-world game 'Horizon: Zero Dawn', we had to develop a method to render large amounts of animated assets in real-time, such as crowds and locations heavily decorated with animated cloth-like assets. That required creating a new pipeline for generating and authoring new content. Limited CPU and RAM budget made us look for a solution to keep the footprint minimal. These challenges had to be met in the middle of production when programmers time is very limited. This forced us to use available technology, tools and to reuse existing content where possible while keeping it compatible with the current pipeline.

2 Previous Work

We already had an implementation of [Norman], which allowed us creating assets with animations baked to a texture that drives a vertex displacement program on GPU. However, this approach is vertex-count bound, i.e. vertex count = horizontal texture resolution; it requires additional normal transformation texture. We had to look for a method to keep the texture size small.

3 Implementing Virtual Bones

Original assets were complex, CPU-heavy, scaling was a challenge. Instead of skinned meshes, driven by skeletons with CPU control structures, assets had to be simplified, computations offset to GPU.

The proposed method includes a vertex displacement shader, driven by a texture. The texture is a rows-columns lookup table, where every row of pixels corresponds to a keyframe and every pixel represents a component vector of a 4×4 transform matrix. A component vector is a 3 component vector, corresponding to RGB, hence 2 vectors can be encoded as RGBRGB (6 color channels or 2 RGB pixels). The remaining vector is restored as a cross-product of the other two. Vertex rest position and rest normal are supplied by the rest-mesh. Using only 3 pixels per bone allowed us to keep the texture size small. Horizontal texture resolution is $n \cdot 3$, rounded to the closest power of 2, where n is the number of bones, vertical resolution equals the number of keyframes. (Figure 1c)

It provides correct vertex position and normal transformation and does not require additional textures to store normal transformation.

4 Reusing content and texture generation

We used SideFX Houdini Animation Tools to set up a pipeline that supported both normal and specialized assets. We could reuse existing materials that were set up in Autodesk Maya and only had to update the displacement shader part. However we ended up creating specialised simplified shaders, sharing displacement program across all assets. This allowed us to avoid any extra work from programmers. FBX was used to export animation of existing rigs from Maya to Houdini (Figure 1a). We built a network in Houdini that extracted transformation of every significant joint as a 4×4 matrix and mapped it to a texture (Figure 1b). Mantra renderer was used to write a 16bit integer unfiltered png texture (Figure 1c). Alembic format was used to send meshes back to Maya (Figure 1d).

5 Results and Performance.

This allowed keeping the texture size small, offset nearly all computations to GPU, keep CPU footprint minimal or nill; reuse existing content, avoid extra work for programmers or changing the pipeline, while significanly improving the visual results. During one encounter the player sees 1200 instances of stadium crowd characters (Fig. 1e) (rig: 18 bones, 202 verts, 352 faces) with a 10 second animation loop mapped to a 64x512 texture, drawing average 114,928 tris, mean GPU time of 0.325μs, RAM use of 256Kb.

6 Limitations and Future Work

Because the horizontal texture resolution was rounded to the closest power of 2, that led to some texture space waste. We used transformation matrices and linear blending - using quaternions can reduce horizontal texture resolution by 30% while improving rotation interpolation between keyframes. To minimize rounding errors the texture had to be stored as RGBA half float (uncompressed). Storing rotation and translation components separately may improve space use and allow compression. More research is required in regards to compressing a virtual bones animation texture.

References

"Vertex-count-agnostic Morph Targets" by Norman Schaar, 2015

*e-mail:vladimir.lopatin82@gmail.com