We present two methods for rendering thousands of animated characters in real-time. We maximize rendering performance by using a collection of pre-computed impostors sampled from a discrete set of view directions. The first method is based on relief impostors [1] and the second one in flat impostors [2]. Our work differs from previous approaches on view-dependent impostors in that we use per-joint rather than per character impostors. Characters are animated by applying the joint rotations directly to the impostors, instead of choosing a single impostor for the whole character from a set of predefined poses. This representation supports any arbitrary pose and thus the agent behavior is not constrained to a small collection of predefined clips. To the best of our knowledge, this is the first time a crowd rendering algorithm encompassing image-based performance, small GPU footprint and animation-independence is proposed.

Our methods:

The goal of both techniques is to optimize the rendering of far away characters while rendering nearby characters with their full animated 3D geometry. Both approaches calculate separate impostors for each animated part of the articulated character. Unlike previous work where a texture was needed per viewpoint and per animation frame, our methods only require a texture for each animated part per view which can be reused for any animation. At run time impostors are transformed in the same way as the bones of the skeleton, giving the impression that our impostor character is animated.

Relief Impostors (RI)

Construction:
1. Associate mesh triangles with box.
2. Select best pose for capturing the impostors.
3. Compute the bounding boxes (OBBs)
4. Capture the textures of each OBB: Color, normal and depth projected onto the 6 box faces.

Pre-process
Real-Time

For each point
Select view directions

Animation with relief impostors

Geometry vs. flat impostors (marked in red)

LOD is used to group bones together: 3 bone hierarchies (21, 7 and 1)

Flat Impostors (FI)

Pre-Process
1. Compute a spherical Voronoi map
2. Build a cube map by projecting the Voronoi cells onto the cube faces.
3. Compute mask and apply to textures (masks consider how the geometry is affected by neighboring bones).

Real-Time
1. Animate character and cube maps.
2. Retrieve the fragment color with a single texture lookup.

Results

Performance:
1 LOD: FI are 5.6–3.3x faster than RI and 5.8–3.3x faster than polygonal meshes.
3 LODs, FI are 2.1–3.1x faster than RI and 4.5–8.2x faster than LOD meshes.

Geometry, Relief and Flat Impostors

Animation with flat impostors

Rendering with/without optimal mask

Conclusions

- Render tens of thousands of characters in real-time.
- Relief maps: ↑ highest image quality, ↓ higher per-fragment overhead (limited to distant characters).
- Flat impostors: ↓ more demanding in terms of texture memory and construction time, ↑ higher runtime performance.
- ↑ both representations outperform polygonal meshes with negligible visual artifacts.
- ↑ Both per-joint impostors support arbitrary animation cycles and animation blending, a missing feature in competing per-character impostors.