Steep Parallax Mapping

Morgan McGuire* and Max McGuire
Brown University and Iron Lore Entertainment

The viewer perceives an intersection at (P) from shading, although the true intersection occurred at (I).

Overview

§ Real-time, appropriate for games
§ Parallax
§ Self-occlusion
§ Self-shadowing
§ Realistic grass and fur in one pass
§ Uses existing art tools and assets

Performance

Renders a full 1024×768 screen of fur at 30 fps on GeForce6800 with 4x super-sampling and 30 ray-marching iterations. Typical scenes with are faster, where not all pixels use steep parallax mapping.

We unroll the loop (up to 7 iterations) for older PS 2.0 cards, which do not have branch instructions.

Related Work


Our heightfield ray tracer achieves similar results using 2D textures, which are compatible with existing art assets and allow high enough resolution to describe fine details like hair.

[2] Donnelly, Per-Pixel Displacement Mapping with Distance Functions, GPU Gems 2, 2005

Pixel Shader

// tsE = tangent space eye vector
void main(void) {
    const int numSteps = 30;
    float height = 1.0, step = 1.0 / numSteps;
    vec2 offset = texCoord.xy;
    vec4 NB = texture2D(bumpMap, offset);
    vec2 delta = vec2(tsE.x, tsE.y) * bumpScale / (tsE.z * numSteps);
    while (NB.a < height) {
        height -= step; offset += delta;
        NB = texture2D(bumpMap, offset);
    }
    // Choose the color at the location we hit
    const vec4 color = texture2D(texMap, offset);
    // Use the normals out of the bump map
    vec3 tSN = NB.xyz * 2.0 - 1.0;
    ...
    Apply illumination algorithm
}

Details

Avoid repeated MIP-computation with texture2DLod. LOD bias of –1.5 and 4x or 8x FSAA act as low-pass filters to avoid undersampling. Self-shadowing by tracing shadow rays to the light. Approximate self-shadowing as increasing with depth [3] for fur.