Sparse GPU Voxelization of Yarn-Level Cloth

Supplementary Material

1. OpenGL and CUDA Interaction

```c
// Register VBO with CUDA
glBindBuffer(GL_ARRAY_BUFFER, m_profileVertexBufferObject); 
registerGLBufferObject(modelMatrixBuffer, &m_cuda_vbo_resource);
```

2. Bindless Texture Management

```c
// Create a 3D texture per block
unsigned int UV_text_handle = glGenTextures(1, &textureBuffer);
unsigned int ModelMatrixBuffer = glGenBuffers(1, &modelMatrixBuffer);
unsigned int ProfileBuffer = glGenBuffers(1, &profileBuffer);
```

3. Vertex Shader

```c
#version 440 core
#extension GL_ARB_bindless_texture : require

// Output position of the vertex, in clip space : MVP
out vec4 MVP;
```

submitted to COMPUTER GRAPHICS Forum (10/2015).
4. Fragment Shader

```c
#define _GLES31_MATH

// Fragment Shader

void main() {
    float PI = 3.14159265358979323846264;

    vec4 aux = texelFetch(tex_handles, index);
    vec4 vol_handle = aux.rgb;
    layout(r8) image3D volDensity = layout(r8) image3D(vol_handle);
    aux = texelFetch(theta_handles, index);
    vol_handle = aux.rgb;
    layout(r8) image3D volTheta = layout(r8) image3D(theta_handles);
    aux = texelFetch(phi_handles, index);
    vol_handle = aux.rgb;
    layout(r8) image3D volPhi = layout(r8) image3D(phi_handles);

    float tmpTheta = imageLoad(volTheta, vec3(x3Dfine, y3Dfine, z3Dfine));
    float tmpPhi = imageLoad(volPhi, vec3(x3Dfine, y3Dfine, z3Dfine));

    // A special value (e.g., 0) is used as non-initialized tag
    // to avoid averaging the first hit (conditional not listed)
    imageStore(volTheta, vec3(x3Dfine, y3Dfine, z3Dfine), vec4((theta+tmpTheta)/2.0));
    imageStore(volPhi, vec3(x3Dfine, y3Dfine, z3Dfine), vec4((phi+tmpPhi)/2.0));
}
```

This shader performs sparse GPU voxelization of yarn-level cloth. It uses uniforms and samplers to load data from the image and update the voxel values accordingly. The shader computes the orientation and updates voxel data based on the current orientation and position.