Implementing Renderman displacement shader in Cg

Yi Yong-il

Renderman displacement shader

- In Renderman, geometry data divides into several micro polygon
  - Usually, this micro polygon is smaller than pixel
- Displacement shader is performed for Renderman micro polygon
- Befor shading, we can change position and normal of micro polygon

Renderman displacement shader example

```
displacement sinewaves(float freq=1.0,
ampl=1.0, sphase=0, tphase=0, paramdir=0)
{
  //displace along normal, using sin(s) or sin(t)
or both
  if(0==paramdir) {
     P +=
ampl*sin(sphase+s*freq*2*PI)*normalize(N);
  }
  else if (1==paramdir) {
      P +=
ampl*sin(tphase+t*freq*2*PI)*normalize(N);
  }
  else {
      P +=
ampl*sin(sphase+s*freq*2*PI)*sin(tphase+t*fre
q*2*PI)*normalize(N);
  }

  N = calculatenormal(P);
}
```

Implementation in Cg

- In case of normal mapping, we can't change shape of geometry
  - When we see edges of the geometry, we can find the problem
- Use geometry shader
  - In geometry shader, subdivide triangles
  - Subdivided triangles becomes input data for next stage
    - To do this, use NV_transform_feedback and buffer_object
  - Rasterization is disabled until subdividing ends

Code(OpenGL)

```
// vbo is for vertex, tex_vbo is for normal
glBindBufferBaseNV(GL_TRANSFORM_FEEDBACK_BUFFER_NV, 0, vbo[1-current_buffer]);
glBindBufferBaseNV(GL_TRANSFORM_FEEDBACK_BUFFER_NV, 1, tex_vbo[1-current_buffer]);

// begin transform feedback
glBeginTransformFeedbackNV(GL_TRIANGLES);
// disable rasterization
glEnable(GL_RASTERIZER_DISCARD_NV);

glBeginQuery(GL_TRANSFORM_FEEDBACK_PRIMITIVES_WRITTEN_NV, query);

            …// draw geometry data

glEndQuery(GL_TRANSFORM_FEEDBACK_PRIMITIVES_WRITTEN_NV);

glEndTransformFeedbackNV();

// read back query results, used for next stage input number
glGetQueryObjectuiv(query, GL_QUERY_RESULT, &primitives_written);
```

Code(OpenGL)

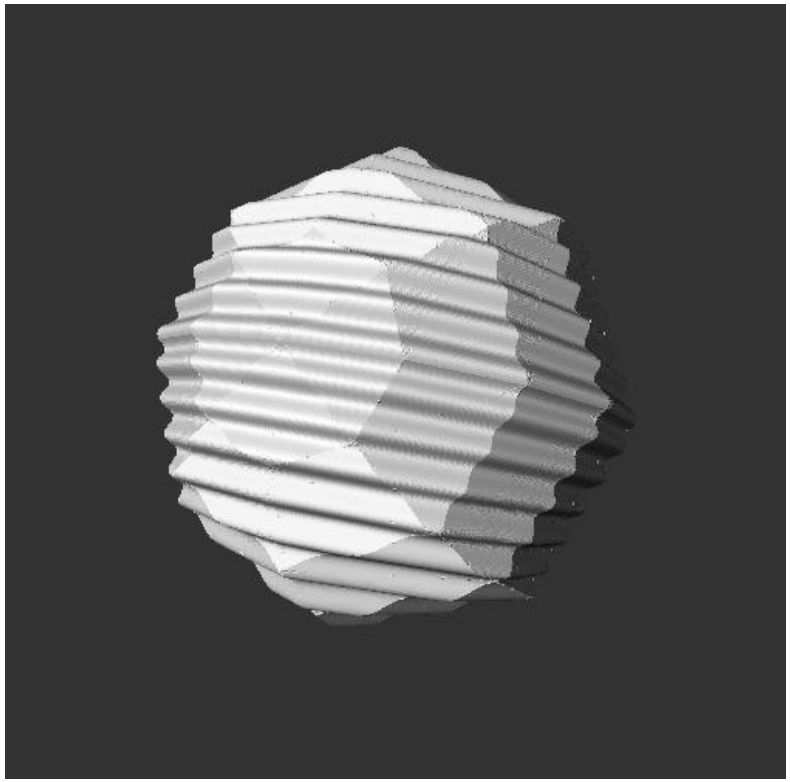
```
// draw geometry
// bind buffer object
glBindBuffer(GL_ARRAY_BUFFER, vertex_buffer);
// set the buffer object to vertex array(0)
glVertexAttribPointerARB(0, 4, GL_FLOAT, GL_FALSE, 0, 0);

// bind buffer object
glBindBuffer(GL_ARRAY_BUFFER, tex_buffer);
// set the buffer object to texture coordinate array(8)
glVertexAttribPointerARB(8, 4, GL_FLOAT, GL_FALSE, 0, 0);

// enable for vertex, texcoord array
glEnableVertexAttribArrayARB(0);
glEnableVertexAttribArrayARB(8);

// draw
glDrawArrays(prim, 0, verts);

// disable
glDisableVertexAttribArrayARB(0);
glDisableVertexAttribArrayARB(8);
```

After subdividing

- In vertex shader
  - Change the position of vertex
    - Example : using sin function
    - new_pos += 0.02*sin(pos.y*freq*2*PI)*normal;
- In geometry shader
  - Calculate normal of triangle
    - float3 edge0 = v1 - v0;
    - float3 edge1 = v2 - v0;
    - normalize(cross(edge0, edge1));
  - Calculate simple lighting and emit the color

Result

Low-polygon sphere



Apply displacement shader