计算机图形学

作业3

陶钧

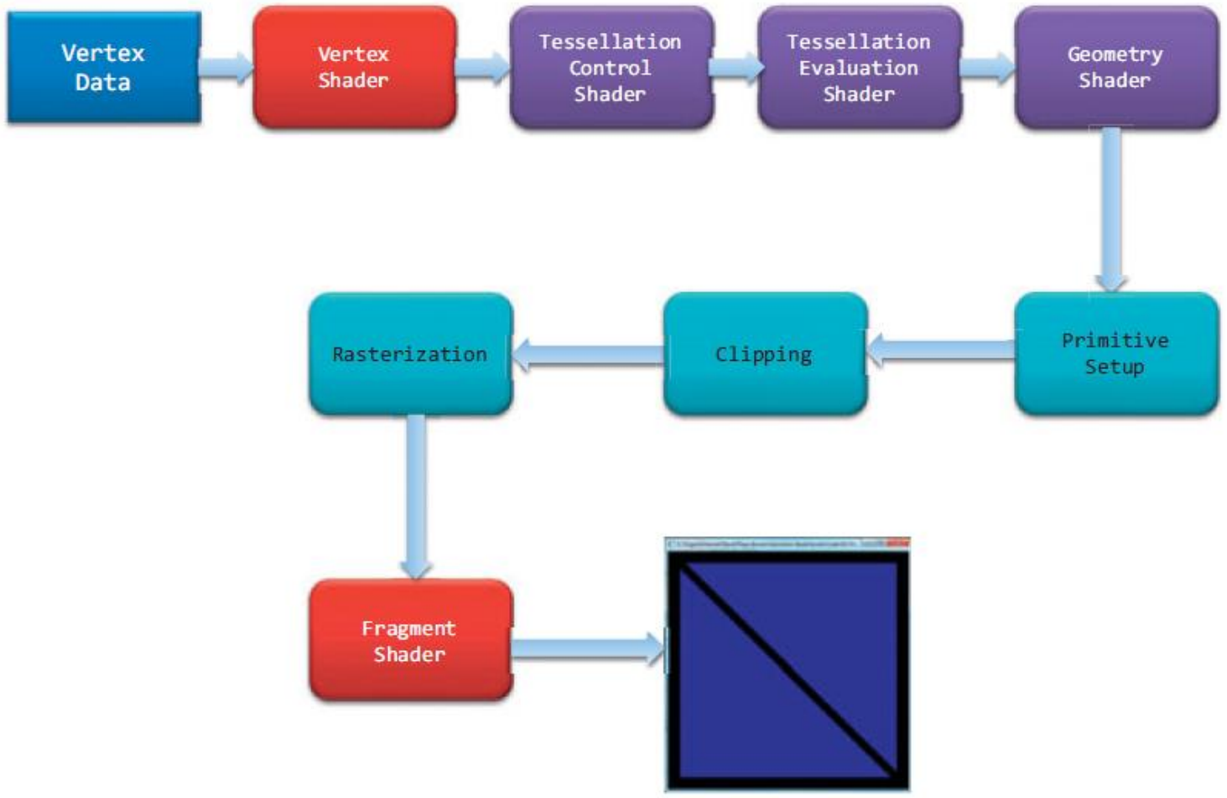
taoj23@mail.sysu.edu.cn

中山大学 数据科学与计算机学院

国家超级计算广州中心

作业3：GLSL入门

- 回顾此前介绍的渲染管线（讲义5）
  - 显卡是高度并行化的硬件，对所有数据采用同一工作流程进行处理
  - GLSL (GLslang)
    - OpenGL Shading Language
    - 对着色器（shader）进行编程
  - Vertex shader
    - 顶点变换，法向量变换
    - 光照
    - 纹理坐标的产生与变换
  - Fragment shader
    - 纹理访问及颜色计算，等



作业3：GLSL入门

- Vertex shader：齐次坐标顶点→屏幕坐标顶点
  - 同时产生顶点相关的属性（经过插值后将输入至fragment shader）

```
varying vec3 normal;
varying vec3 vertex_to_light_vector;

void main()
{
   // Transforming The Vertex
   gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;

   // Transforming The Normal To ModelView-Space
   normal = gl_NormalMatrix * gl_Normal;

   // Transforming The Vertex Position To ModelView-Space
   vec4 vertex_in_modelview_space = gl_ModelViewMatrx * gl_Vertex;

   // Calculating The Vector From The Vertex Position To The Light Position
   vertex_to_light_vector = vec3(gl_LightSource[0].position – vertex_in_modelview_space);
}
```

来自http://nehe.gamedev.net/article/glsl_an_introduction/25007/index.html

作业3：GLSL入门

- Fragment shader：计算片元颜色
  - 计算过程中可能使用顶点属性插值后得到的片元属性

```
varying vec3 normal;
varying vec3 vertex_to_light_vector;

void main()
{
   // Defining The Material Colors
   const vec4 AmbientColor = vec4(0.1, 0.0, 0.0, 1.0);
   const vec4 DiffuseColor = vec4(1.0, 0.0, 0.0, 1.0);

   // Scaling The Input Vector To Length 1
   vec3 normalized_normal = normalize(normal);
   vec3 normalized_vertex_to_light_vector = normalize(vertex_to_light_vector);

   // Calculating The Diffuse Term And Clamping It To [0;1]
   float DiffuseTerm = clamp(dot(normal, vertex_to_light_vector), 0.0, 1.0);

   // Calculating The Final Color
   gl_FragColor = AmbientColor + DiffuseColor * DiffuseTerm;
}
```

作业3：GLSL入门

- GLSL中的数据类型
  - 向量： vec2, vec3, vec4, ivec2, ivec3, ivec4, bvec2, bvec3, bvec4
  - 矩阵： mat2, mat3, mat4
  - 纹理： sampler1D, sampler2D, sampler3D, samplerCube, sampler1Dshadow, sampler2Dshadow
- GLSL中的数据修饰词
  - uniform：对所有顶点而言为常量，不因顶点而异（如光源位置）
  - attribute：因顶点而异，只读，只能在vertex shader中使用
  - varying：vertex shader的输出，fragment shader的输入，传输过程中进行插值
  - in, out：表明变量为输入或输出

作业3：GLSL入门

- GLSL中的内置变量
  - Vertex shader中的内置attribute
    - gl_Vertex, gl_Normal, gl_Color, gl_MultiTexCoordX
  - 内置uniform
    - gl_ModelViewMatrix, gl_ModelViewProjectionMatrix, gl_NormalMatrix
  - Shader输出
    - vertex shader： gl_Position
    - fragment shader：gl_FragColor, gl_FragDepth

作业3：GLSL入门

- 编译GLSL着色器程序

```
GLuint vertexShader = glCreateShader(GL_VERTEX_SHADER);
GLuint fragmentShader = glCreateShader(GL_FRAGMENT_SHADER);

glShaderSource(vertexShader, 1, &vsource, 0);
glShaderSource(fragmentShader, 1, &fsource, 0);

glCompileShader(vertexShader);
glCompileShader(fragmentShader);

GLuint program = glCreateProgram();

glAttachShader(program, vertexShader);
glAttachShader(program, fragmentShader);

glLinkProgram(program);
```

作业3：GLSL入门

- 使用GLSL着色器程序
  - **glUseProgram**(program)：指明在接下来的绘制中使用program所代表的着色器程序
  - **glUseProgram**(0)：使用默认着色器
- 设置程序中uniform变量的值
  - 获取uniform变量在程序中的位置
    - **glGetUniformLocation**(program, ”variable_name")
  - 设置uniform变量在程序中的值
    - **glUniform{a}{b}{c}**(location, value);
    - {a}: 1, 2, 3, 4
    - {b}: f, i, ui
    - {c}: /, v

作业3：GLSL入门

- 作业内容：使用GL_POINTS绘制沿固定轨道运动的小球
  - 每个glVertex调用指明一个小球的球心位置
  - 小球大小根据离观察点距离变化（近大远小）
  - 使用Phong shading
- 参考vertex shader及fragment shader

```
uniform float radius;

void main()
{
   gl_PointSize = radius;

   gl_TexCoord[0] = gl_MultiTexCoord0;
   gl_Position = gl_ModelViewProjectionMatrix
               * gl_Vertex.xyz

   gl_FrontColor = gl_Color;
}
```

```
void main()
{
   // calculate normal from texture coordinates
   vec3 N;
   N.xy = gl_TexCoord[0].xy * vec2(2.0, -2.0)
        + vec2(-1.0, 1.0);
   float mag = dot(N.xy, N.xy);

// kill pixels outside circle
   if (mag > 1.0) discard;

   gl_FragColor = gl_Color;
}
```

作业3：GLSL入门

- 参考绘制函数：建议使用vbo进行绘制（可选做）

```
glEnable(GL_POINT_SPRITE_ARB);
glTexEnvi(GL_POINT_SPRITE_ARB, GL_COORD_REPLACE_ARB, GL_TRUE);
glEnable(GL_VERTEX_PROGRAM_POINT_SIZE_NV);

glUseProgram(program);
glUniform1f(glGetUniformLocation(program, "radius"), 10.0f);

glBindBufferARB(GL_ARRAY_BUFFER_ARB, point_vbo);
glVertexPointer(3, GL_FLOAT, 0, 0);
glEnableClientState(GL_VERTEX_ARRAY);

glBindBufferARB(GL_ARRAY_BUFFER_ARB, color_vbo);
glColorPointer(4, GL_FLOAT, 0, 0);
glEnableClientState(GL_COLOR_ARRAY);

glDrawArrays(GL_POINTS, 0, num_points);

glBindBufferARB(GL_ARRAY_BUFFER_ARB, 0);
glDisableClientState(GL_VERTEX_ARRAY);
glDisableClientState(GL_COLOR_ARRAY);

glUseProgram(0);
```

作业3：GLSL入门

- 交作业时间：2020年1月4日晚23时59分
- 评分标准与此前作业相同
- slip days的使用与此前作业相同

Questions?