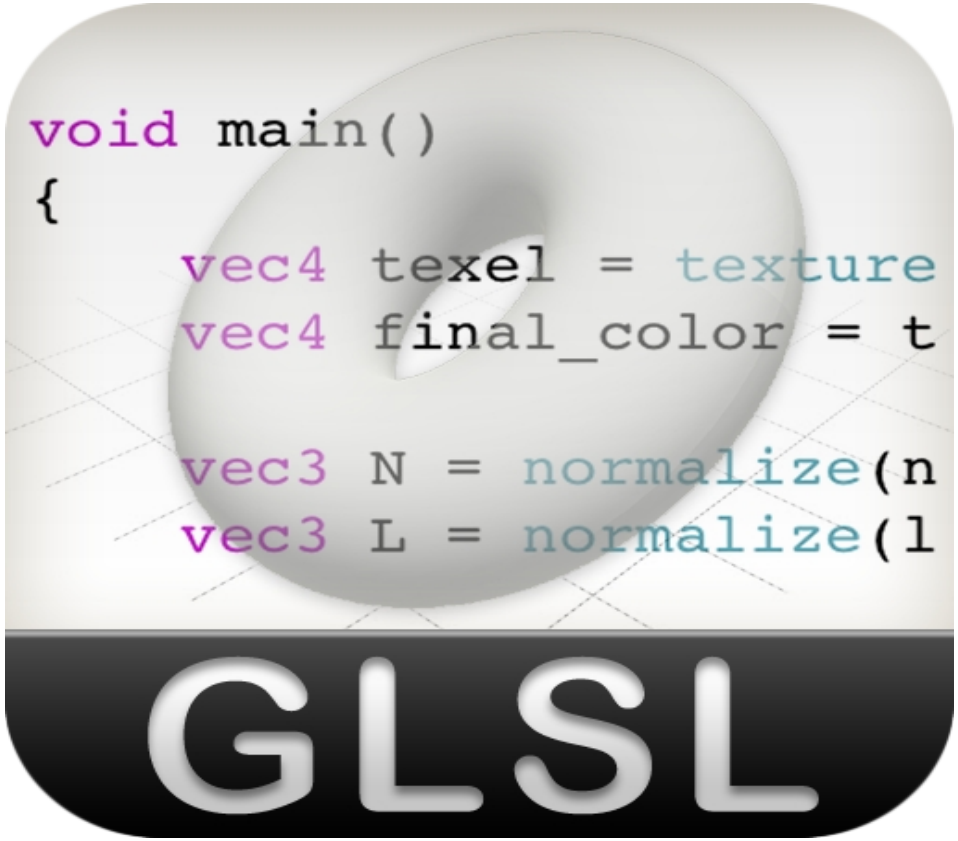
Praktikum Computergrafik

Steven Schlegel

Abteilung für Bild- und Signalverarbeitung

Betreuer: Steven Schlegel (schlegel@informatik.uni-leipzig.de)


OpenGL


void main()
{
vec4 texel = texture
vec4 final_color = t
vec3 N = normalize(n
vec3 L = normalize(l
GLSL

OpenGL

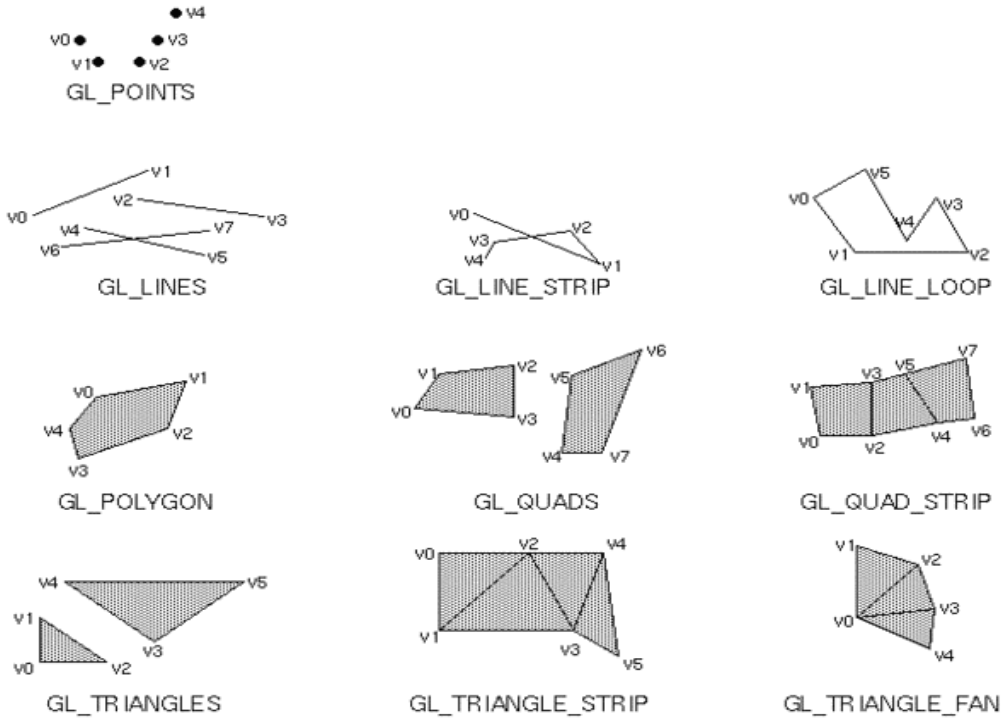
- **OpenGL** (**Open G**raphics **L**ibrary)
  - plattform- und programmiersprachenunabhängige Programmierschnittstelle zur Entwicklung von 2D- und 3D-Computergrafik
  - ermöglicht die Darstellung komplexer 3D-Szenen in Echtzeit
  - Implementierung ist normalerweise durch Grafikkartentreiber gewährleistet (hardwarebeschleunigt), ansonsten auf der CPU
  - Windows-Pendant: Direct3D

OpenGL

- Bekannte Spiele:
  - Escape From Monkey Island (GrimE-Engine)
  - Doom 3, Brink, Rage (Id Tech 4/5 Engine)
  - Neverwinter Nights (Aurora Engine)
  - Half Life 2 (Source Engine)

OpenGL – Programmierung (C/C++ Interface)

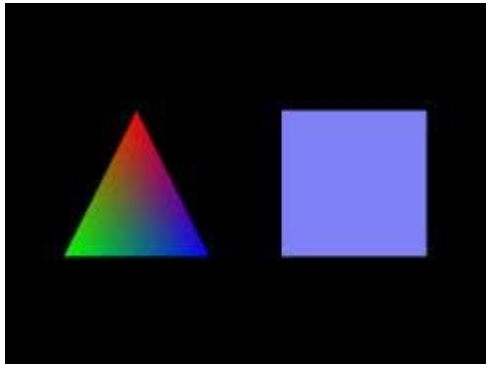
- OpenGL ist ein Zustandsautomat
  - ein Zustand bleibt immer so lange erhalten, bis er explizit geändert wird (z.B. Hintergrundfarbe - *glClearColor*)
- Primitive werden innerhalb eines glBegin()/glEnd() Blockes gezeichent

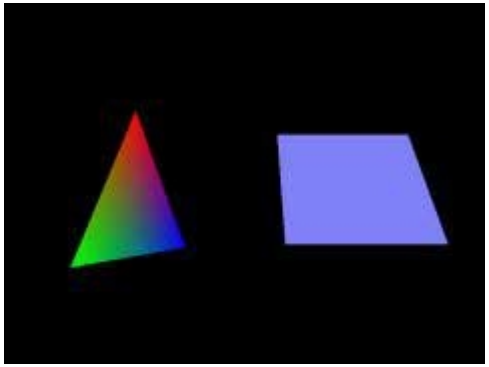


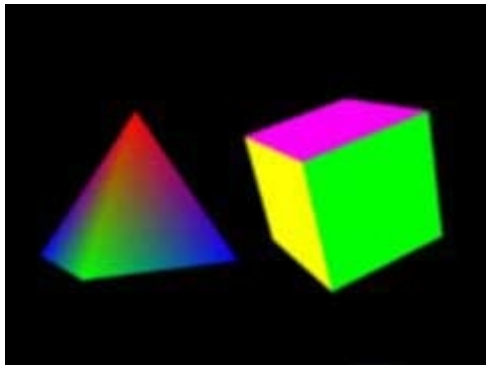
OpenGL - Tutorial

- From http://nehe.gamedev.net/tutorial/your_first_polygon/13002/

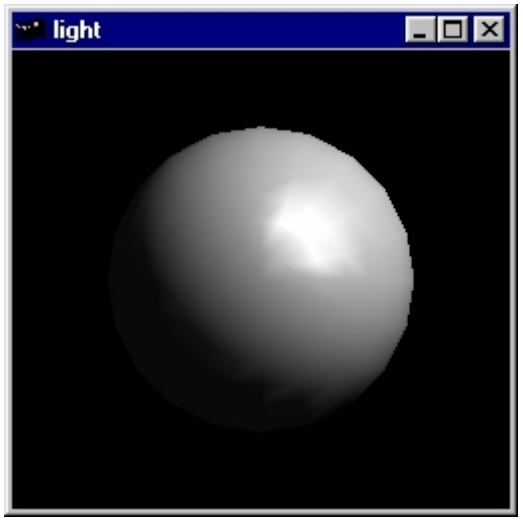
```
void paintGL()                                          // Here's Where We Do All The Drawing
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT); // Clear Screen And Depth Buffer
    glLoadIdentity();                                   // Reset The Current Modelview Matrix
    glTranslatef(-1.5f,0.0f,-6.0f);                     // Move Left 1.5 Units And Into The Screen 6.0
    glBegin(GL_TRIANGLES);                              // Drawing Using Triangles
        glVertex3f( 0.0f, 1.0f, 0.0f);                  // Top
        glVertex3f(-1.0f,-1.0f, 0.0f);                  // Bottom Left
        glVertex3f( 1.0f,-1.0f, 0.0f);                  // Bottom Right
    glEnd();                                            // Finished Drawing The Triangle
    glTranslatef(3.0f,0.0f,0.0f);                       // Move Right 3 Units
    glBegin(GL_QUADS);                                  // Draw A Quad
        glVertex3f(-1.0f, 1.0f, 0.0f);                  // Top Left
        glVertex3f( 1.0f, 1.0f, 0.0f);                  // Top Right
        glVertex3f( 1.0f,-1.0f, 0.0f);                  // Bottom Right
        glVertex3f(-1.0f,-1.0f, 0.0f);                  // Bottom Left
    glEnd();                                            // Done Drawing The Quad
}
```








NeHe
GameDev

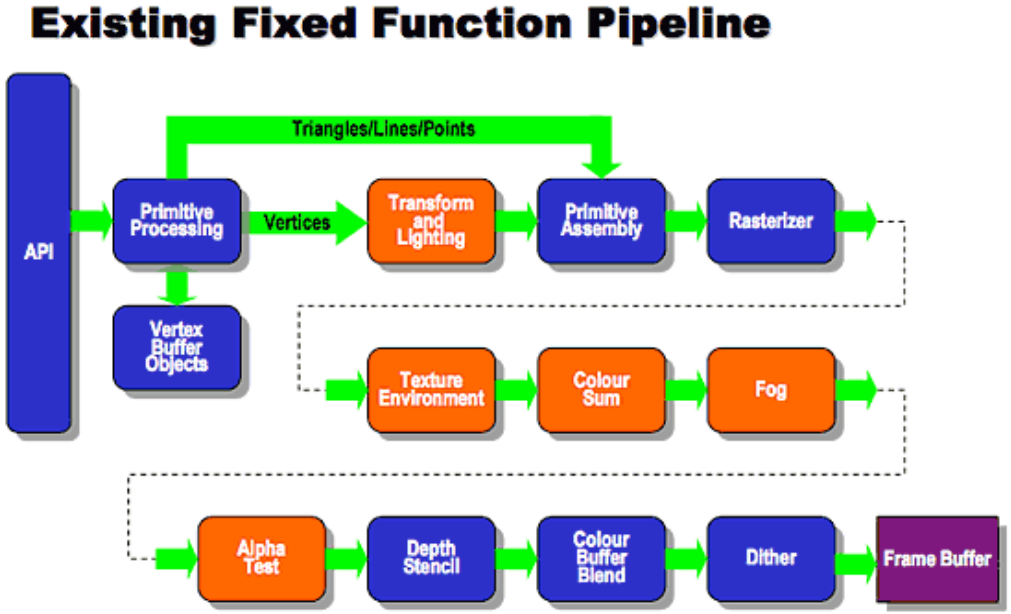

light

OpenGL - Limitierung

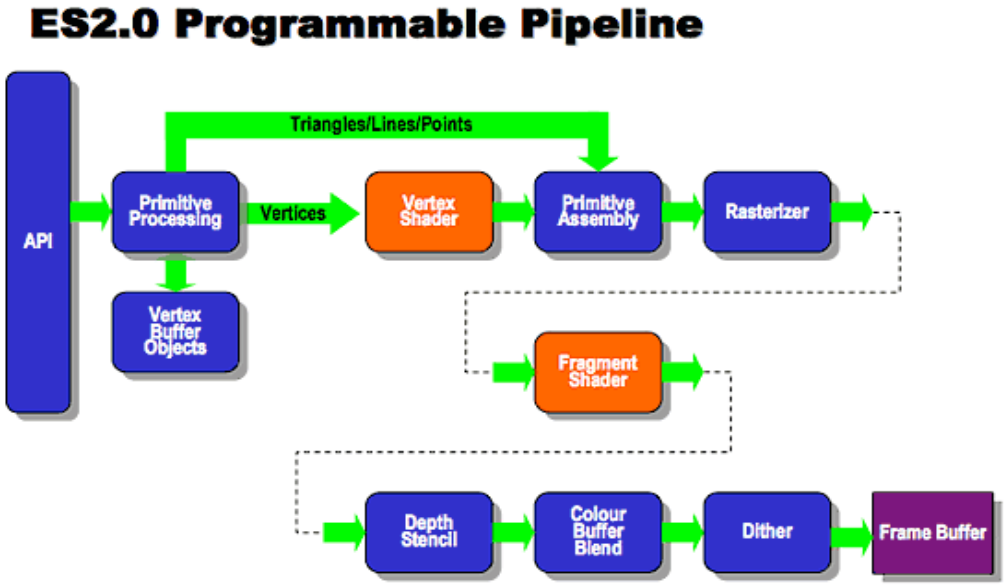
- Alle Operationen in OpenGL lassen sich höchstens auf per-Vertex Operationen herunterbrechen (z.B. Färben)
  - per – Pixel Operationen sind nicht möglich!
  - z.B. Beleuchtung:
    - Für jedes Primitiv können Materialeigenschaften übergeben werden und für jeden Vertex kann eine Normale übergeben werden

      → Gouraud Shading (auch Flat Shading) mit Phong Beleuchtung (siehe z.B. CG-Viewer) sind fest integriert

- Man kann verschiedene Einstellungen festlegen, sobald man aber die Primitive (Vertices) an die Grafikkarte übergibt kann man das Renderergebnis nicht weiter beeinflussen!
    - FIXED PIPELINE



OpenGL



OpenGL + GLSL

GLSL

- ...eine Programmiersprache, um mittles OpenGL eigene Programme auf der Grafikkarte (*Shader*) auszuführen
- Unter DirectX: HLSL
- im Praktikum: Beschränkung auf Vertex- und Pixelshader (GLSL version 1.5)
    - Ausserdem gibt es z.B. noch Geometry-, Hull- und Tesselationsshader oder sog. Unified Shader
- **Pipeline**:
    - OpenGL übergibt Vertex mit verschiedenen Eigenschaften (Material oder Farbe, Texturkoordinaten, Farbe usw.)
    - Vertexshader “bearbeitet” den Vertex und evtl. die übergebenen Eigenschaften
    - Pixelshader bekommt die *interpolierten* Eigenschaften (z.B. Normalen) und färbt das Pixel im Framebuffer

GLSL

- Syntax entspricht im Wesentlichen ANSI-C
- wurde um spezielle Datentypen erweitert, wie z.B. Vektoren, Matrizen und Sampler (für Texturzugriffe)
- Tutorial z.B. unter http://nehe.gamedev.net/article/glsl_an_introduction/25007/

GLSL - Beispiel

- VertexShader:

```
varying vec3 normal, vertex;

void main()
{
    //gl_Position    = ftransform()
    gl_Position     = gl_ModelViewProjectionMatrix * gl_Vertex;
    normal          = normalize(gl_NormalMatrix * gl_Normal);
    vertex          = vec3(gl_ModelViewMatrix * gl_Vertex);
}
```

GLSL - Beispiel

- PixelShader:

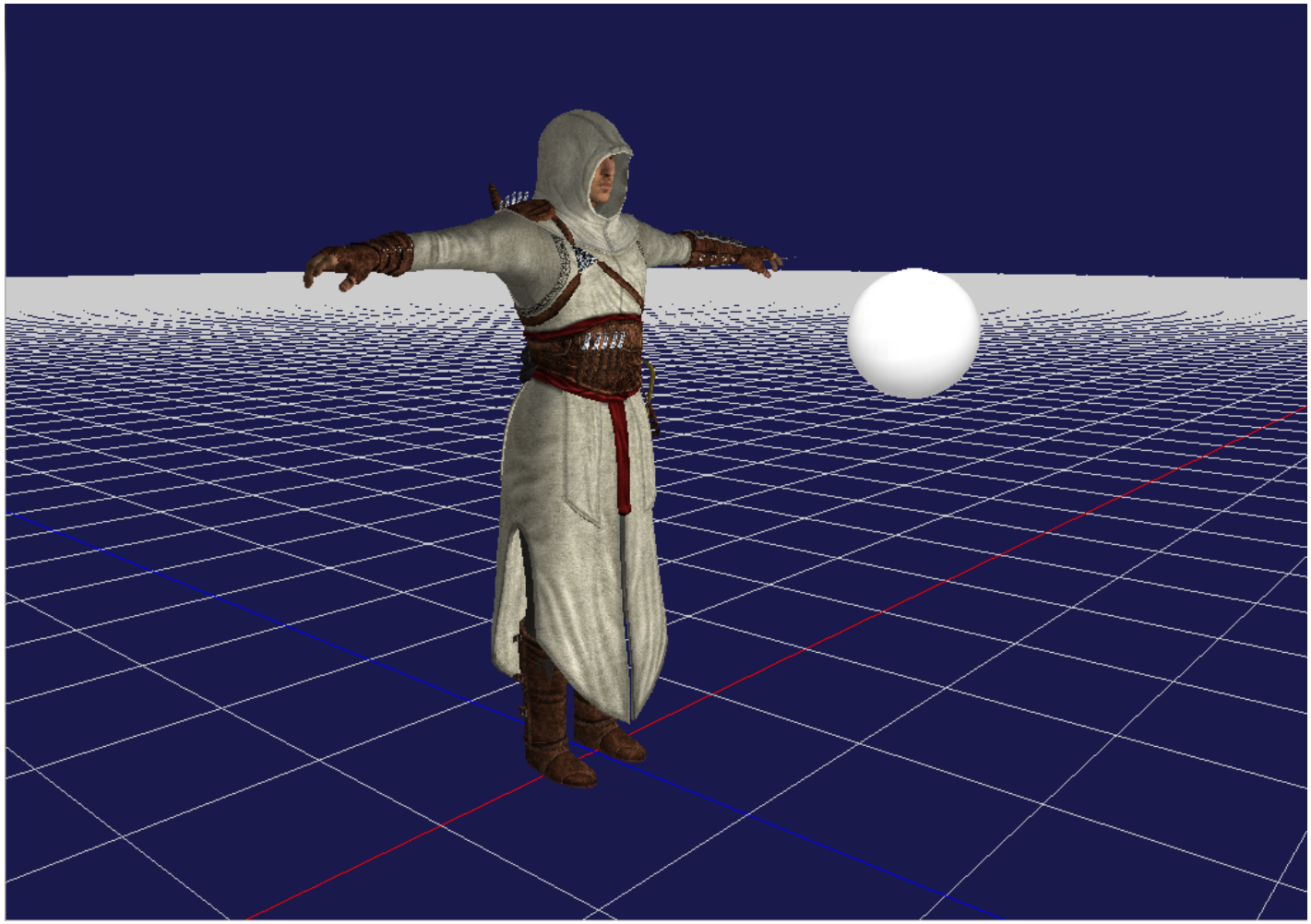
```
uniform int numLights;
varying vec3 normal, vertex;

void main()
{
    vec4 color = vec4(0.0, 0.0, 0.0, 1.0);
    for (int l=0; l<numLights; ++l)
    {
        vec3 lightPosition = gl_LightSource[l].position.xyz;
        vec3 vertexToLight = normalize(lightPosition - vertex);
        color.r += dot ( normal, vertexToLight );
    }

    gl_FragColor = color;
}
```

GLSL - Beispiel

- ohne Shader



GLSL - Beispiel

- mit Shader