



**Westfälische
Hochschule**

Gelsenkirchen Bocholt Recklinghausen

Computergrafik

Transformation

Rendering

Prof. Dr. Tom Vierjahn

Visual Computing (<https://vc.w-hs.de>)
Fachbereich Wirtschaft und Informationstechnik
Campus Bocholt

Sommersemester 2020



Veröffentlicht unter der Creative-Commons-Lizenz

Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0)

Dateiversion 10. Mai 2020, 23:30 — Seite 1

VBO-Inhalt (mit Normale)

$v_{0,x}$	$v_{0,y}$	$v_{0,z}$	$n_{0,x}$	$n_{0,y}$	$n_{0,z}$	$v_{1,x}$	$v_{1,y}$	$v_{1,z}$	$n_{1,x}$	$n_{1,y}$	$n_{1,z}$	$v_{2,x}$	$v_{2,y}$	$v_{2,z}$	$n_{2,x}$	$n_{2,y}$	$n_{2,z}$
-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------

Daten-Upload

```
glBufferData(GL_ARRAY_BUFFER, 72, ..., GL_STATIC_DRAW);
```

Daten-Zuordnung

```
glVertexAttribPointer(?, 3, GL_FLOAT, GL_FALSE, ?, ?);
```

Rendern

```
glDrawArrays(GL_TRIANGLES, 0, ?);
```

Definition: Uniform¹

A **uniform** is a global Shader variable declared with the **uniform** storage qualifier. These act as parameters that the user of a shader program can pass to that program. [...]

Uniforms [...] do not change from one shader invocation to the next within a particular rendering call.

Als Eingabe im (Vertex)-Shader-Code

```
uniform mat4 model_to_world_matrix;
```

Zuweisung im Programm-Code

```
GLint glGetUniformLocation(GLuint program, const GLchar* name);  
void glUniformMatrix4fv(GLint location, GLsizei count, bool transpose,  
                        const GLfloat* value);
```

¹Uniform (GLSL). (2019, April 25). OpenGL Wiki, . Retrieved May 1, 2020 from [http://www.khronos.org/opengl/wiki/opengl/index.php?title=Uniform_\(GLSL\)&oldid=14539](http://www.khronos.org/opengl/wiki/opengl/index.php?title=Uniform_(GLSL)&oldid=14539).

Shader-Code

```
#version 330 core

in vec3 vertex_normal_worldspace;
uniform vec3 user_color;

layout(location = 0) out vec3 color;

void main() {
    float nz = vertex_normal_worldspace.z;
    float factor = 0.5 + 0.5 * abs(nz);
    color = factor * user_color;
}
```

- ▶ Rasterisierung erzeugt Fragmente.
- ▶ Shader wird für jedes Fragment aufgerufen.
- ▶ Ausgabe geht ans Blending.

- ▶ Geometrirepräsentation mit Normalen
- ▶ Uniforms für Matrizen
- ▶ ein etwas komplexerer Fragment-Shader

Prof. Dr. Tom Vierjahn

► E-Mail: tom.vierjahn@w-hs.de

Visual Computing

- Web: <https://vc.w-hs.de>
- YouTube: Visual Computing WH
- Twitter: @VisComputingWH

Westfälische Hochschule
Fachbereich Wirtschaft und Informationstechnik
Campus Bocholt



Veröffentlicht unter der Creative-Commons-Lizenz
Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0)