

Aufbau einer modernen Rendering-Pipeline

Von der Anwendung zum Bildschirm

Fakultät für Informatik - Lehrstuhl für Computergrafik - Institut für Visualisierung und Datenanalyse



Motivation



Motivation	Anwendung	Geometry Stage	Rasterisierung	Per Frag Ops	Koordinatensysteme	Compute Shader	Ausblick
○○	○	○○○○○○○○	○	○	○	○	○○○○○○○○
Jonas Heinle – Aufbau einer modernen Rendering-Pipeline							
						23. Juni 2019	2/25

Motivation



Motivation



Anwendung



Geometry Stage



Rasterisierung



Per Frag Ops



Koordinatensysteme



Compute Shader



Ausblick

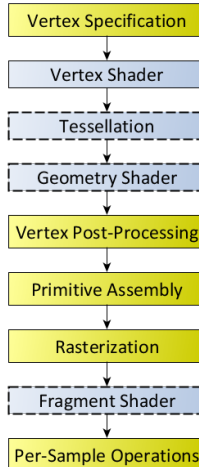


Jonas Heinle – Aufbau einer modernen Rendering-Pipeline

23. Juni 2019

3/25





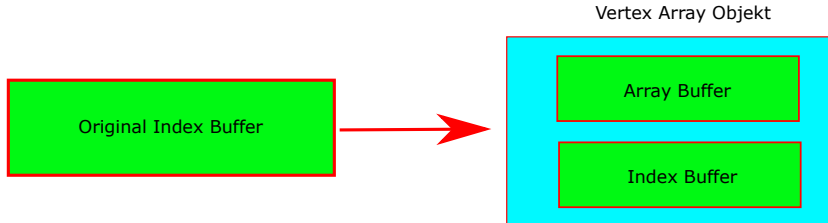
7

```
1 # Blender v2.79 (sub 0) OBJ File: ''
2 # www.blender.org
3 mtl lib Suzanne.mtl
4 o Suzanne
5 v 0.437500 0.164062 0.765625
6 v -0.437500 0.164062 0.765625
7 v 0.500000 0.093750 0.687500
8 v -0.500000 0.093750 0.687500
9 v 0.546875 0.054688 0.578125
```

Abbildung: Beispielhafte
Objektdefinition

```
4 newmtl Material
5 Ns 96.078431
6 Ka 1.000000 1.000000 1.000000
7 Kd 0.640000 0.640000 0.640000
8 Ks 0.500000 0.500000 0.500000
9 Ke 0.000000 0.000000 0.000000
10 Ni 1.000000
11 d 1.000000
12 illum 2
13
```

Abbildung: Beispielhafte
Materialdefinition



Eckpunkt in Modelkoordinaten
und Attribute



Vertex
Shader

Eckpunkt in Clip Space



Motivation



Anwendung



Geometry Stage



Rasterisierung



Per Frag Ops



Koordinatensysteme



Compute Shader



Ausblick

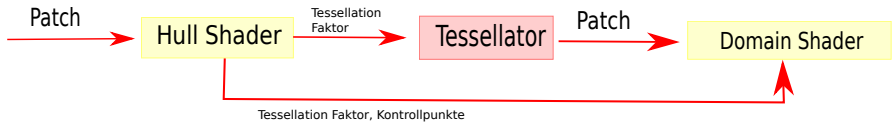


Jonas Heinle – Aufbau einer modernen Rendering-Pipeline

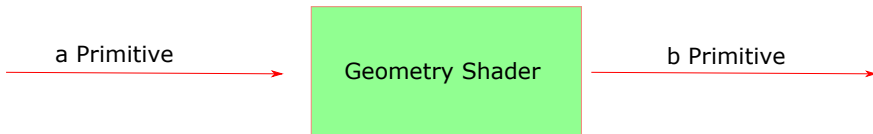
23. Juni 2019

7/25

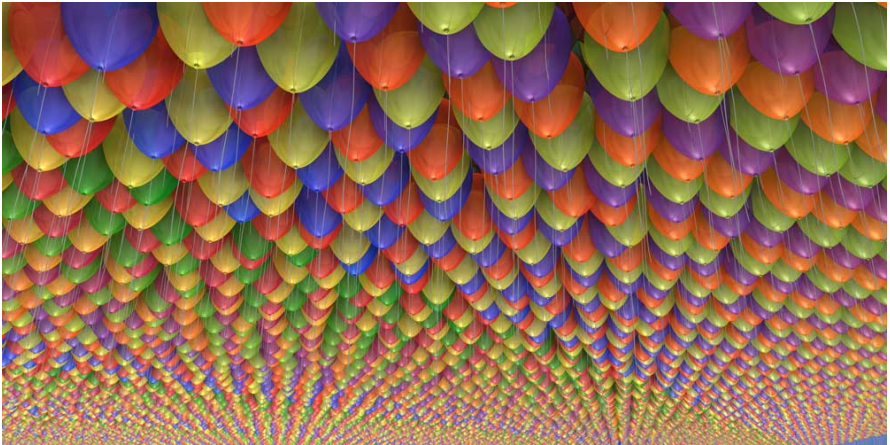




Primitive (Punkt, Linie, Dreieck) werden vervielfacht, entfernt oder umgewandelt



Geometry Shader - Instanziierung



25

Motivation



Anwendung



Geometry Stage



Rasterisierung



Per Frag Ops



Koordinatensysteme



Compute Shader



Ausblick



Jonas Heinle – Aufbau einer modernen Rendering-Pipeline

23. Juni 2019

10/25



```
layout (triangles) in;  
layout (triangle_strip , max_vertices = 3 )out;  
void main() {  
    for(int j = 0; j < gl_in.length();j++) {  
        gl_Position = MVP * gl_in[j].gl_Position;  
        vec3 P = vec3 ( MV * in_position );  
        //Gourand Shading  
        vec3 N = normalize ( vec3 ( matrixNormale  
                                * vec4 ( in_normal , 0.0 ) )  
        vec3 L = normalize ( lightSourcePos-P );  
        color = vec4 ( max ( 0.0, dot( L, N ) )  
        EmitVertex();  
    }  
    EndPrimitive();  
}
```

Motivation

Anwendung

Geometry Stage

Rasterisierung

Per Frag Ops

Koordinatensysteme

Compute Shader

Ausblick

○○

○

○○○○○●○○

○

○

○

○

○○○○○○○○

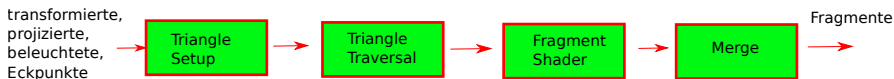
Jonas Heinle – Aufbau einer modernen Rendering-Pipeline

23. Juni 2019

11/25

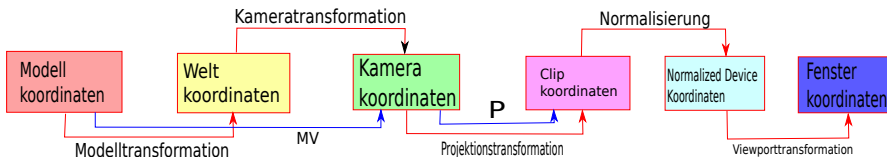


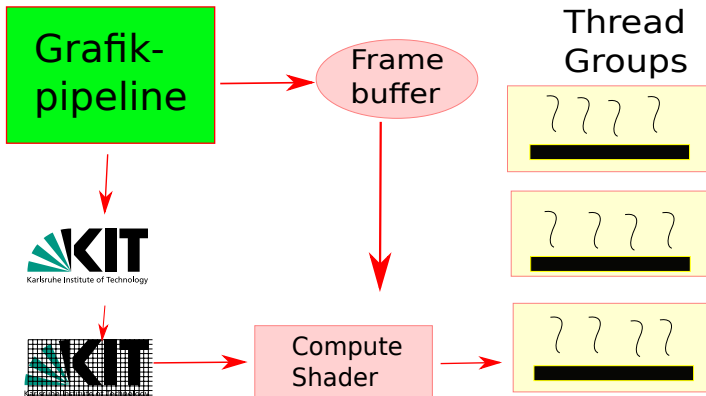
Motivation	Anwendung	Geometry Stage	Rasterisierung	Per Frag Ops	Koordinatensysteme	Compute Shader	Ausblick
○ ○	○	○ ○ ○ ○ ○ ○ ● ○	○	○	○	○	○ ○ ○ ○ ○ ○ ○ ○
Jonas Heinle – Aufbau einer modernen Rendering-Pipeline						23. Juni 2019	12/25

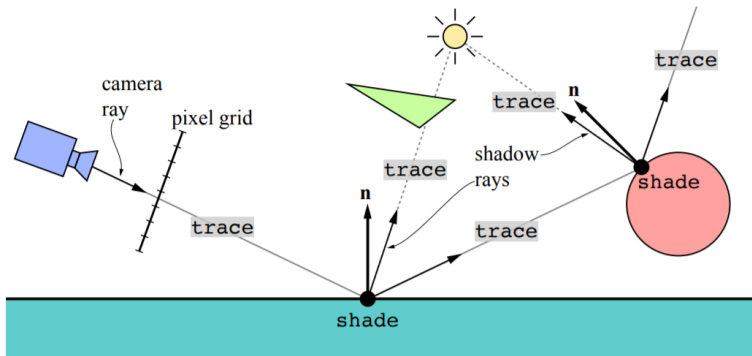


Per Fragment Operations

Motivation ○○	Anwendung ○	Geometry Stage ○○○○○○○○	Rasterisierung ○	Per Frag Ops ●	Koordinatensysteme ○	Compute Shader ○	Ausblick ○○○○○○○○
Jonas Heinle – Aufbau einer modernen Rendering-Pipeline						23. Juni 2019	15/25







25



25

Motivation

○○

Anwendung

○

Geometry Stage

○○○○○○○○

Rasterisierung

○

Per Frag Ops

○

Koordinatensysteme

○

Compute Shader

○

Ausblick

○●○○○○○○

Jonas Heinle – Aufbau einer modernen Rendering-Pipeline

23. Juni 2019

19/25



- Transparenz und Durchsichtigkeit Ray-Tracing
- Global Illumination Ray-Tracing Compute
- G-Buffer Layout Rasterisierung
- Direkte Schatten Ray-Tracing Rasterisierung
- Reflexionen Ray-Tracing Compute
- Direkte Beleuchtung Compute
- Post-Processing Compute

25

- Transparenz und Durchsichtigkeit Ray-Tracing
- Global Illumination Ray-Tracing Compute
- G-Buffer Layout Rasterisierung
- Direkte Schatten Ray-Tracing Rasterisierung
- Reflexionen Ray-Tracing Compute
- Direkte Beleuchtung Compute
- Post-Processing Compute

25

Hybrides Rendering in *PICA PICA*

- Transparenz und Durchsichtigkeit Ray-Tracing
- Global Illumination Ray-Tracing Compute
- G-Buffer Layout Rasterisierung
- Direkte Schatten Ray-Tracing Rasterisierung
- Reflexionen Ray-Tracing Compute
- Direkte Beleuchtung Compute
- Post-Processing Compute

25

- Transparenz und Durchsichtigkeit Ray-Tracing
- Global Illumination Ray-Tracing Compute
- G-Buffer Layout Rasterisierung
- Direkte Schatten Ray-Tracing Rasterisierung
- Reflexionen Ray-Tracing Compute
- Direkte Beleuchtung Compute
- Post-Processing Compute

25

- Transparenz und Durchsichtigkeit Ray-Tracing
- Global Illumination Ray-Tracing Compute
- G-Buffer Layout Rasterisierung
- Direkte Schatten Ray-Tracing Rasterisierung
- Reflexionen Ray-Tracing Compute
- Direkte Beleuchtung Compute
- Post-Processing Compute

25

- Transparenz und Durchsichtigkeit Ray-Tracing
- Global Illumination Ray-Tracing Compute
- G-Buffer Layout Rasterisierung
- Direkte Schatten Ray-Tracing Rasterisierung
- Reflexionen Ray-Tracing Compute
- Direkte Beleuchtung Compute
- Post-Processing Compute

25

- Transparenz und Durchsichtigkeit Ray-Tracing
- Global Illumination Ray-Tracing Compute
- G-Buffer Layout Rasterisierung
- Direkte Schatten Ray-Tracing Rasterisierung
- Reflexionen Ray-Tracing Compute
- Direkte Beleuchtung Compute
- Post-Processing Compute

25

Task/-Mesh Shaders



25

Motivation



Anwendung



Geometry Stage



Rasterisierung



Per Frag Ops



Koordinatensysteme



Compute Shader



Ausblick



Jonas Heinle — Aufbau einer modernen Rendering-Pipeline

23. Juni 2019

21/25

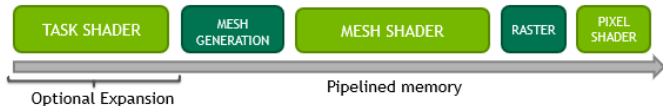


MESHLETS

TRADITIONAL PIPELINE



TASK/MESH PIPELINE



Fragen?



25

Motivation	Anwendung	Geometry Stage	Rasterisierung	Per Frag Ops	Koordinatensysteme	Compute Shader	Ausblick
○○	○	○○○○○○○	○	○	○	○	○○○○●○○
Jonas Heinle – Aufbau einer modernen Rendering-Pipeline						23. Juni 2019	23/25



Interesse geweckt ...?

Weiterführende Literatur

- 1 Quake2 Real-Time Raytracing Project Q2VKPT
- 2 NVIDIA RTX Ray Tracing
- 3 RTX
- 4 Physically Based Rendering
- 5 Heise about NVIDIA Asteroid Demo
- 6 Link to very much information!
- 7 Vulkan Rendering Pipeline

Bilder

- Titelbild
- Cyberpunk 2077
- Kette
- Pipeline
- SEED's Project *PICA PICA*
- NVIDIA Asteroid Scene
- Witcher Endslide
- Geometry Shader Instanziierung
- Ray Tracing Prinzip

Videos

- 1 SEED's Project *PICA PICA*
- 2 **Nvidia's Asteroids Mesh Shaders Demo**

Motivation	Anwendung	Geometrie Shader	Instanziierung	Per-Vertex-Color	Koordinatensysteme	Compute Shader	Ausblick
○○○	○○○	○○○○○○○○	○○	○○	○	○	○○○○○○○●
Jonas Heinle – Aufbau einer modernen Rendering-Pipeline						23. Juni 2019	25/25
◀ ▶ ⌂ 🔍 ↺							