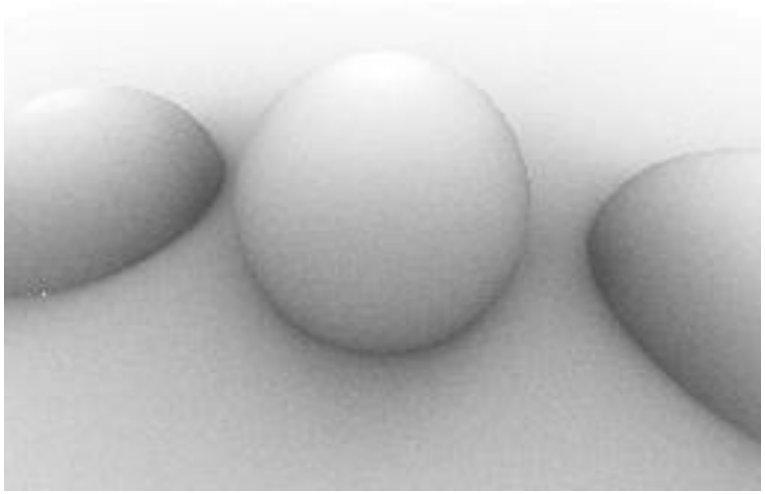
Computergraphik

Modernes OpenGL

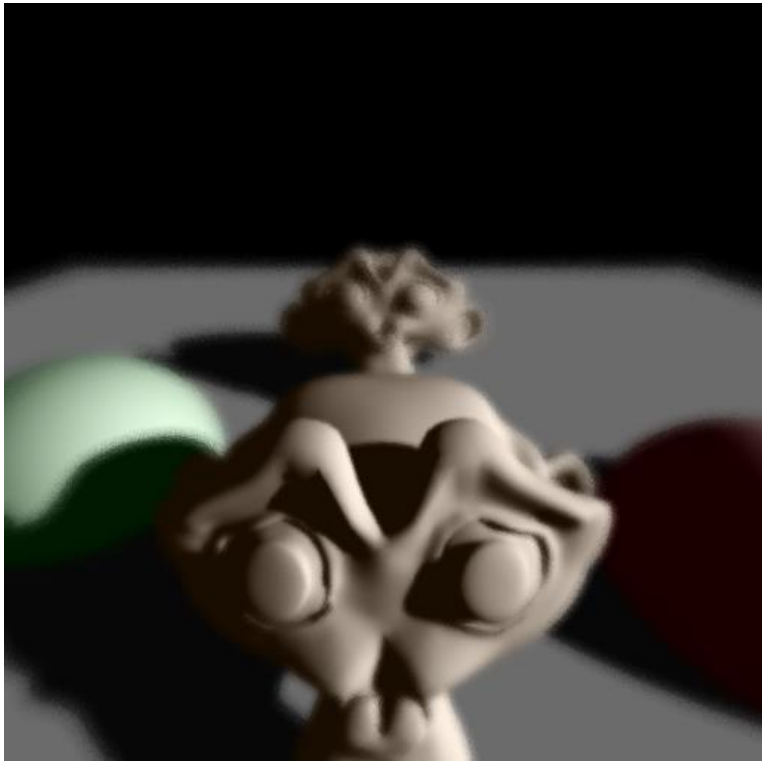
Prof. Dr.-Ing. Carsten Dachsbacher
Lehrstuhl für Computergrafik
Karlsruher Institut für Technologie

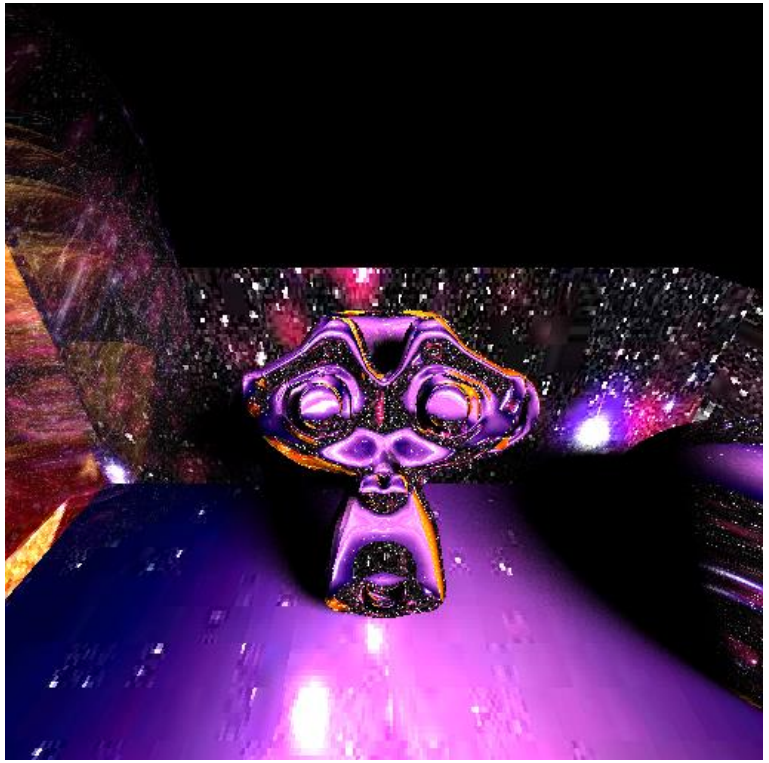
Evaluation

Distributed Raytracing – Best of

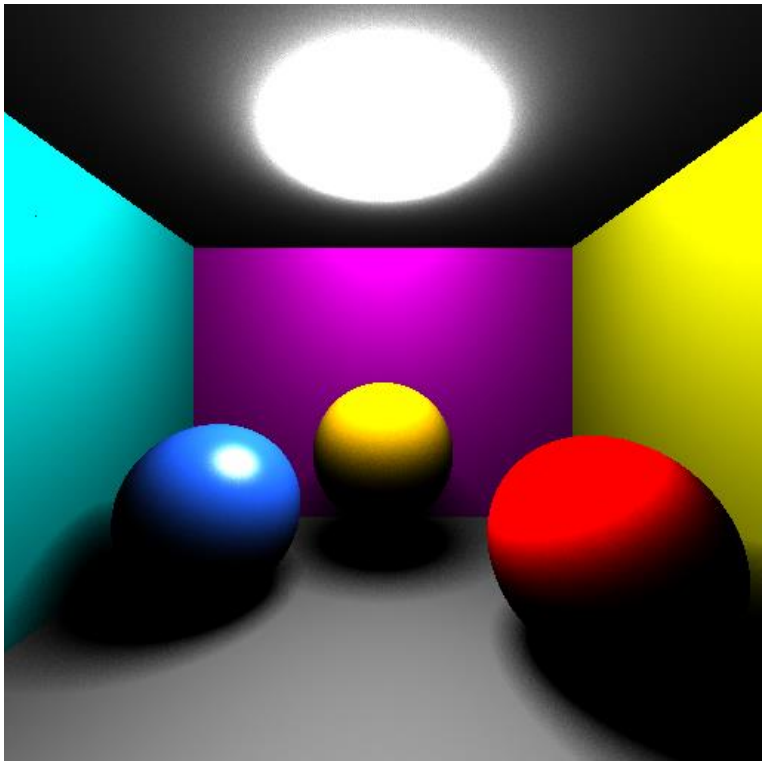


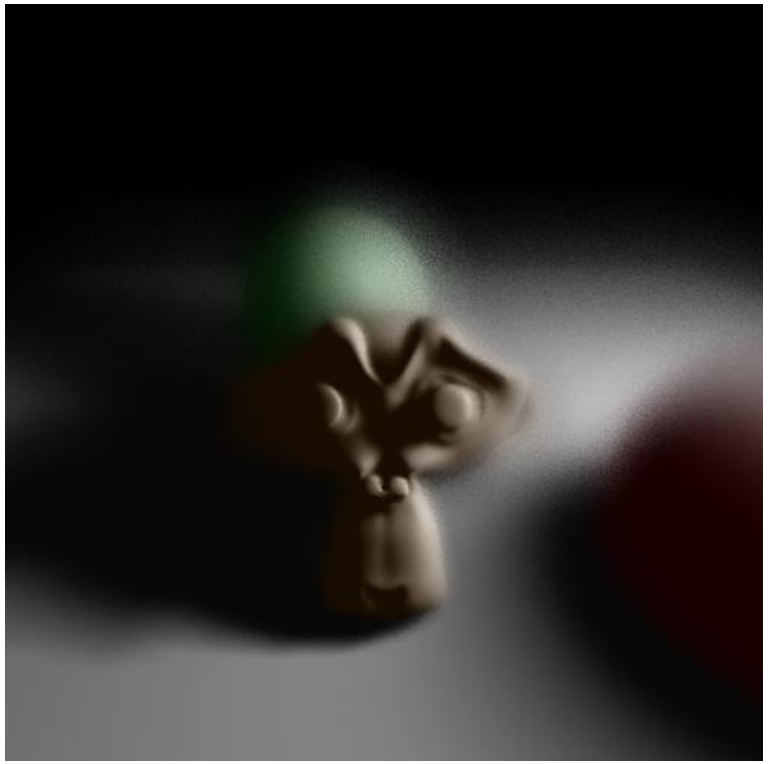
Distributed Raytracing – Best of



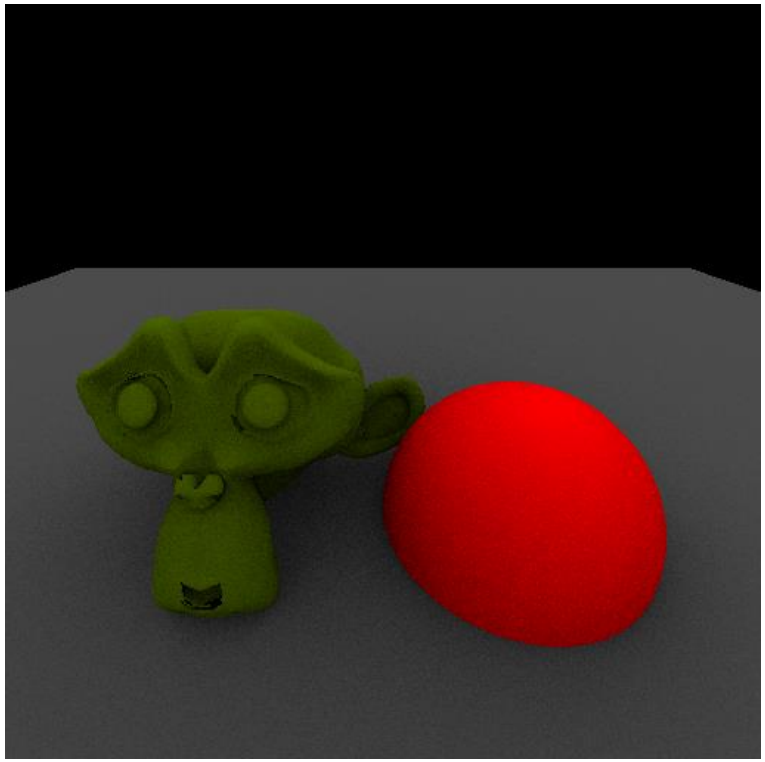


Distributed Raytracing – Best of

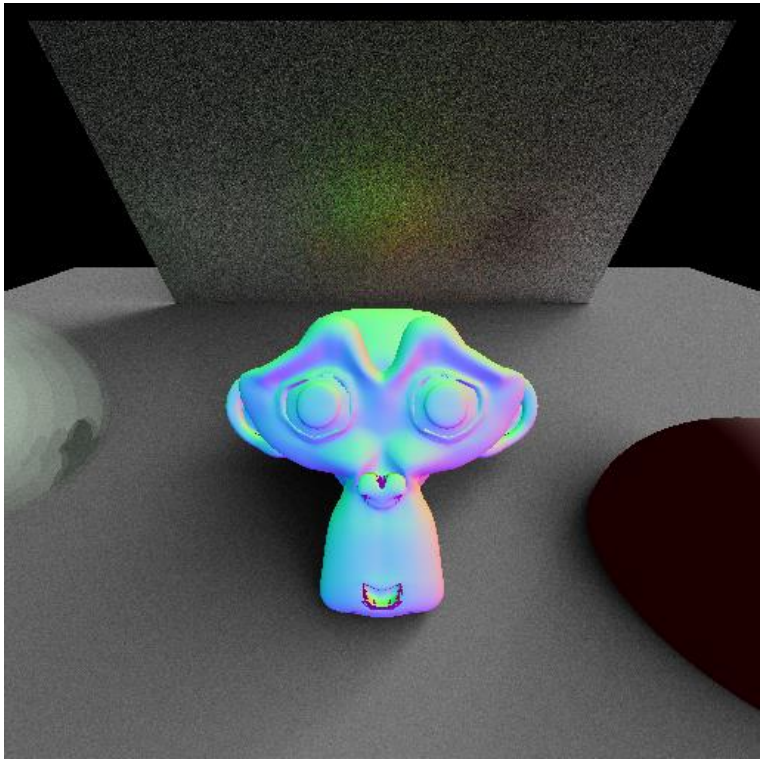


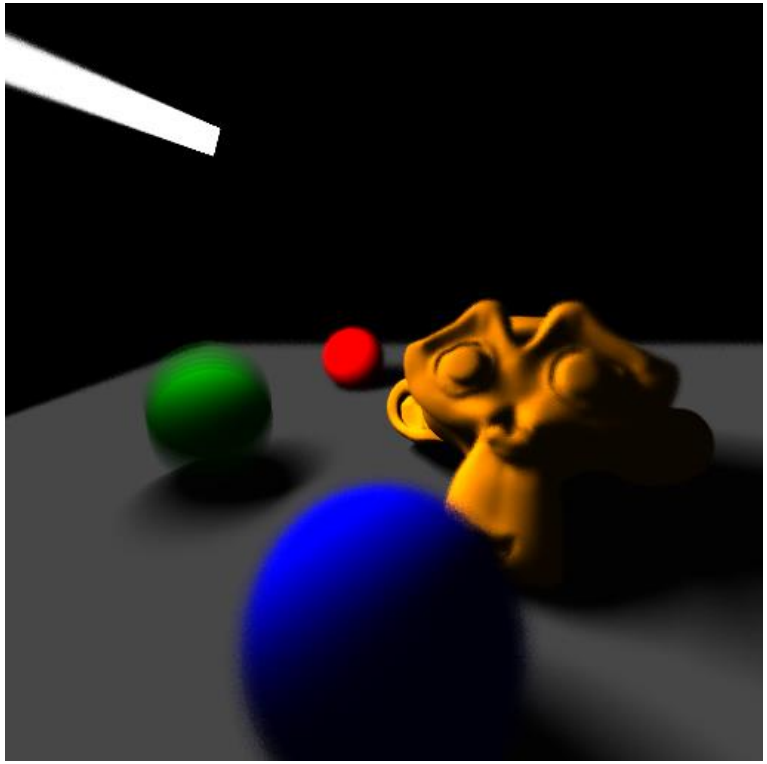


Distributed Raytracing – Best of

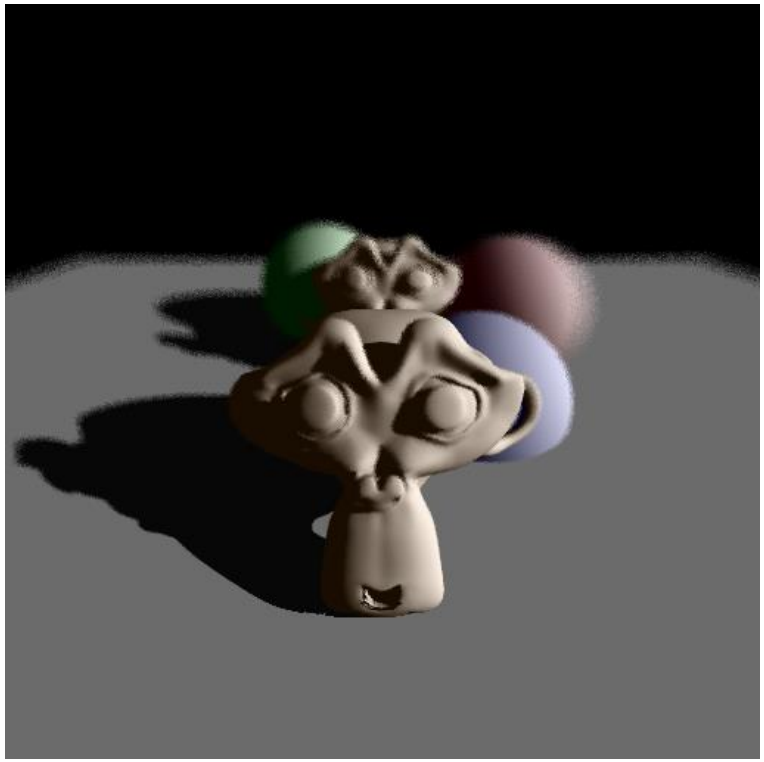


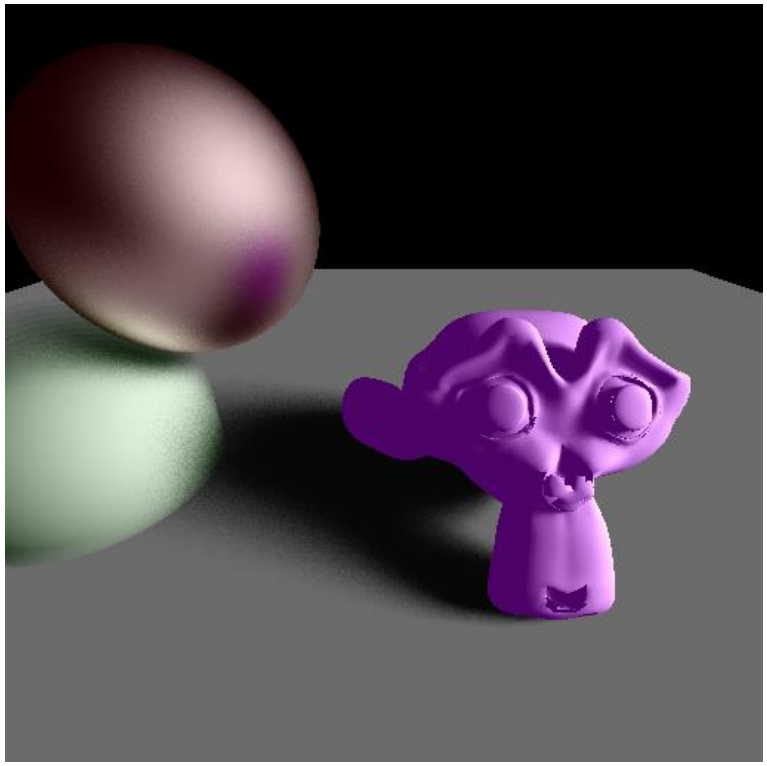
Distributed Raytracing – Best of





Distributed Raytracing – Best of





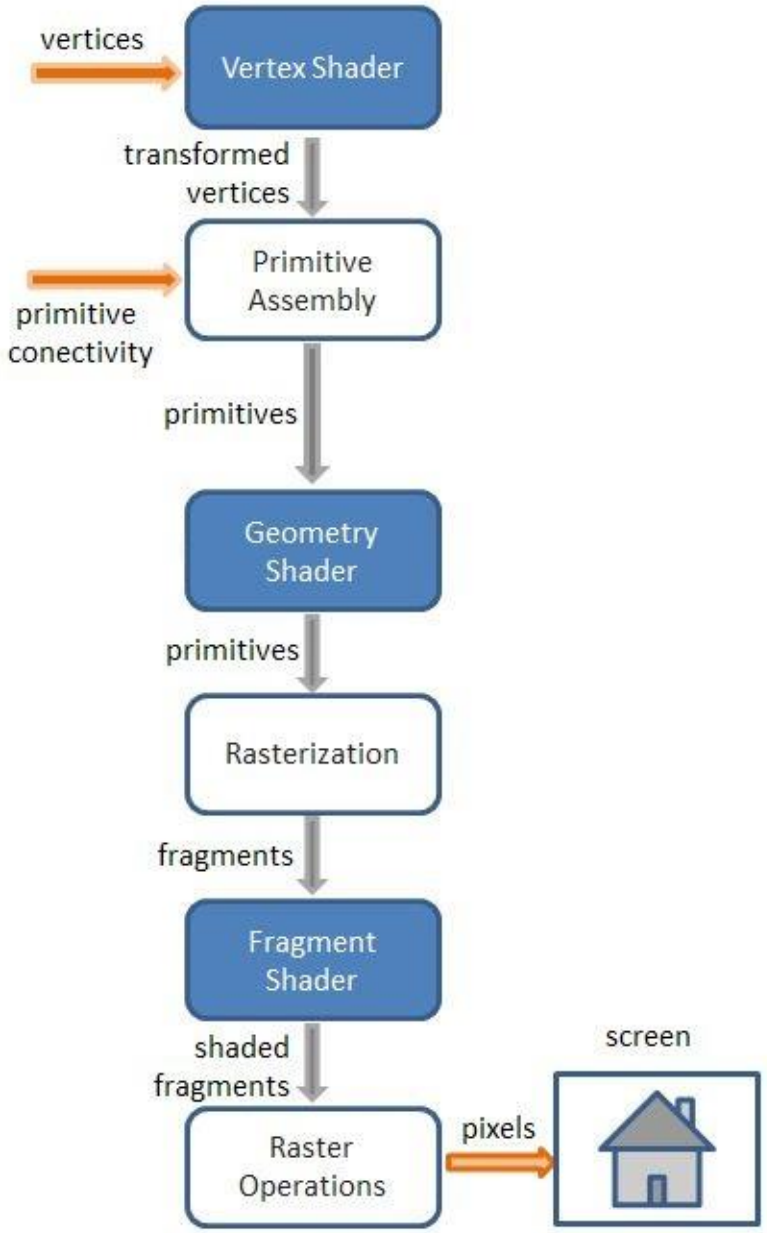
Distributed Raytracing

- Motion Blur: Deterministisches Mitteln über ein paar Bilder zu unterschiedlichen Zeitpunkten erzeugt kein physikalisch korrektes Motion-Blur.

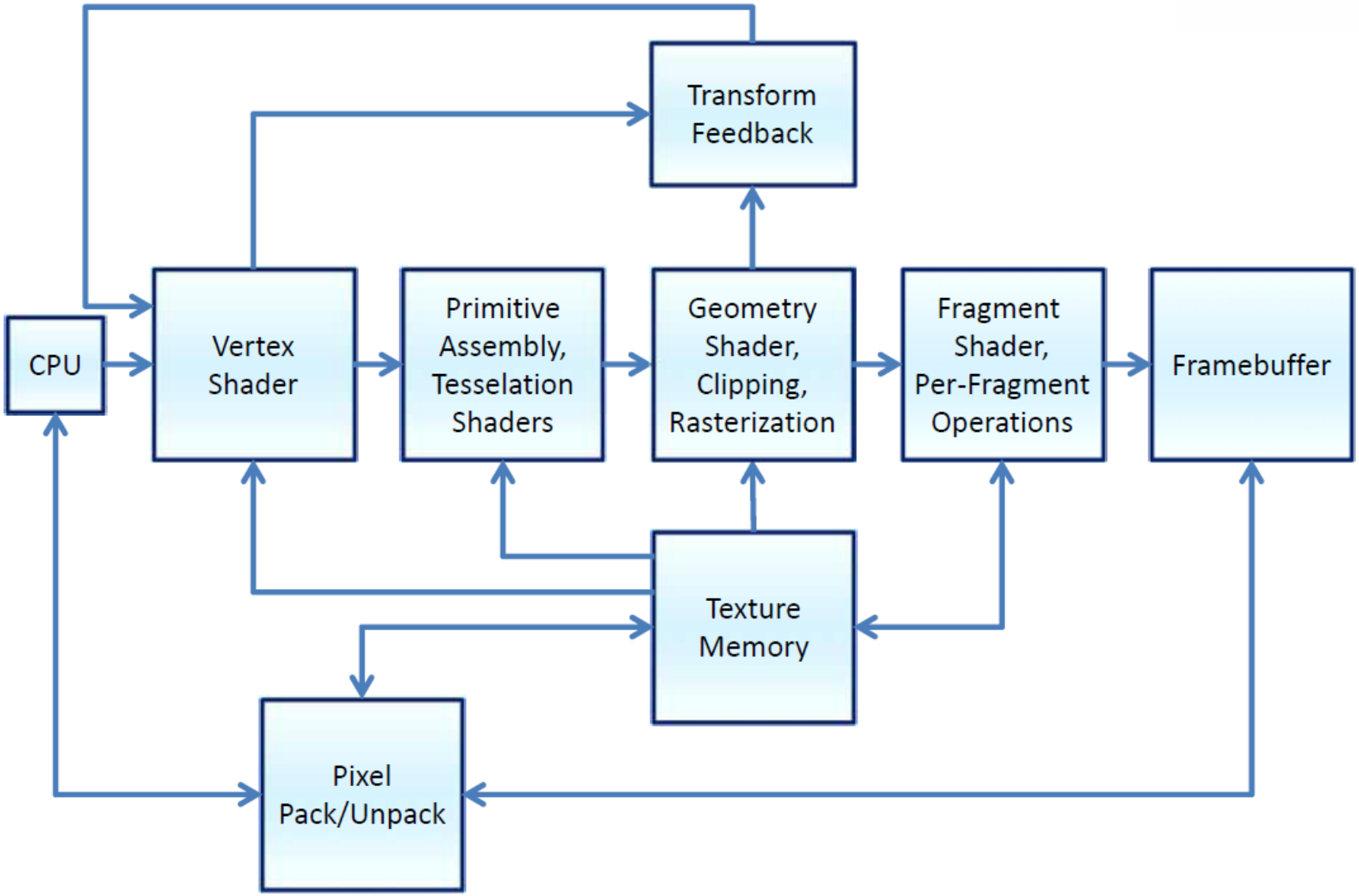
- Generell wird ein kontinuierlicher Raum (Zeit 1d, Linsenfläche (2d)) zufällig abgetastet.

- Jeder „gültige“ Wert muss mit Wkt > 0 gewählt werden können

Moderne OpenGL Pipeline – 3.x



Moderne OpenGL Pipeline – 4.x



Modernes OpenGL

- Shader-Stufen sind programmierbar
- Vertex, Fragment, Geometry- und Tesselation-Shader
- C-ähnliche Programmiersprachen (GLSL, HLSL, ...)
  - Mit eigenen Datentypen (vec3, mat4, ... siehe glm)
- In Shadern kann flexibel auf Texturen und Buffer zugegrifen werden
- Geometrieinformationen können in den Speicher auf der Graphik-HW geladen und wiederverwendet werden
- ...

Vertex-Shader

- Eingaben sind üblicherweise per Vertex
  - Position
  - Normale
  - Farbe oder Texturkoordinate
- Ausgabe
  - Weitergeleitete Vertex-Attribute
  - Position nach MVP-Transformation

- Vorstellung
  - Für alle Vertices werden parallel die Vertex-Shader von der Graphik-HW ausgeführt
  - „Ein Thread pro Vertex“

Vertex-Shader

- Beispiel:

```
varying vec4 position;
varying vec4 normal;

void main() {
      gl_Position = MVP * gl_Vertex;

      position = MV * gl_Vertex;
      normal = N * gl_Normal;
}
```

- `gl_Vertex` und `gl_Normal` werden beim „draw-Aufruf“ an die Graphik-HW geschickt.
- Wie kann man die Matrizen **`MVP`**, **`MV`** und **`N`** übergeben?

Uniforms

- Können durch OpenGL Aufrufe mit Daten gefüllt werden
- Bleiben während des gesamten „draw-Aufrufs“ konstant

```
uniform mat4 MVP;
uniform mat4 MV;
uniform mat4 N;

varying vec4 position;
varying vec4 normal;

void main() {
      gl_Position = MVP * gl_Vertex;

      position = MV * gl_Vertex;
      normal = N * gl_Normal;
}
```

Uniforms

- Beispiel

```
glm::mat4 mv = glm::scale(...);

GLuint mvLocation =
      glGetUniformLocation(program, "MV");

glUniformMatrix4fv(mvLocation, 1, GL_FALSE, &mv);
```

Fragment-Shader

- Eingaben sind üblicherweise
  - interpolierte Vertex-Attribute
  - Beleuchtungsinformationen
- Ausgabe
  - Farbe des Fragmentes

- Vorstellung
  - Für alle Fragmente werden parallel die Fragment-Shader von der Graphik-HW ausgeführt
  - „Ein Thread pro Pixel/Fragment“

Fragment-Shader

- Beispiel:

```
varying vec4 position;
varying vec4 normal;

void main() {
      vec3 color = normalize(normal.xyz);
      color += vec3(1.f, 1.f, 1.f));
      color *= 0.5f;

      gl_FragColor = vec4(color, 1.f);
}
```

- Gibt die Normaleninformation als Farbe aus
  - hilfreich zum debuggen

Phong-Shading != Phong-Beleuchtungsmodell

- Phong-Beleuchtungsmodell
  - Materialmodell (wie bei Ray-Tracing)
  - Diffuse + Spekulare Komponente

- Phong-Shading
  - Shading bzw. Beleuchtungsberechnung **pro Fragment** mit interpolierter Normale
  - Auch mit anderen Beleuchtungsmodellen möglich

- Gouraud-Shading
  - Beleuchtungsberechnung **pro Vertex** mit gemittelter Normale der anliegenden Facetten
- Flat-Shading
  - Beleuchtungsberechnung **pro Vertex** (oder pro Fragment) mit Facetten-Normale

Shading Arten