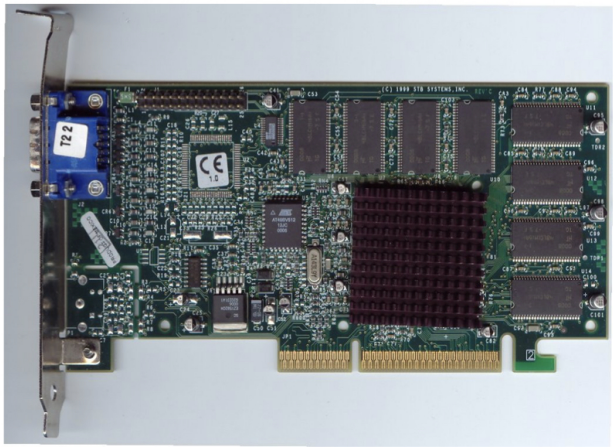
Echzeit Darstellung

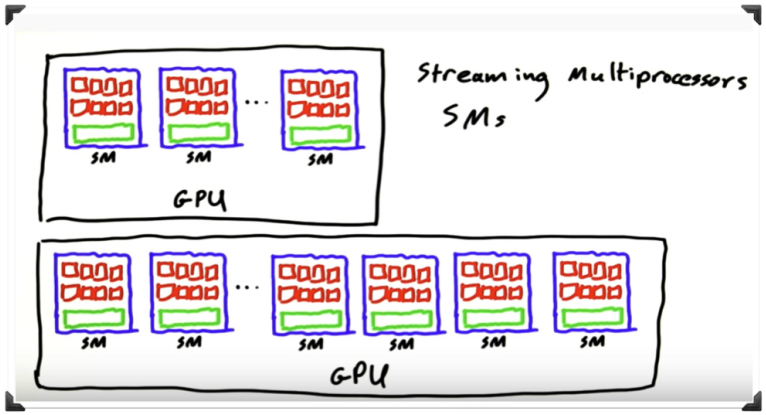
Echtzeit-Typ	**Eigenschaften**
Harte Echtzeit	Zeitlimits zwingend, Systemfehler bei Verpassen.
Weiche Echtzeit	Kleine Abweichungen erlaubt, Leistung sinkt bei Überschreitung.

Echzeit Darstellung

Standard: 60 Frames pro Sekunde bei einer Bildschirmauflösung von $4K = 3840 \times 2160$ Pixel. D.h. 497.664.000 Farbwerte müssen pro Sekunde berechnet und an das Ausgebegerät geschickt werden. Kombination von Soft- und Hardware nötig.

GPU

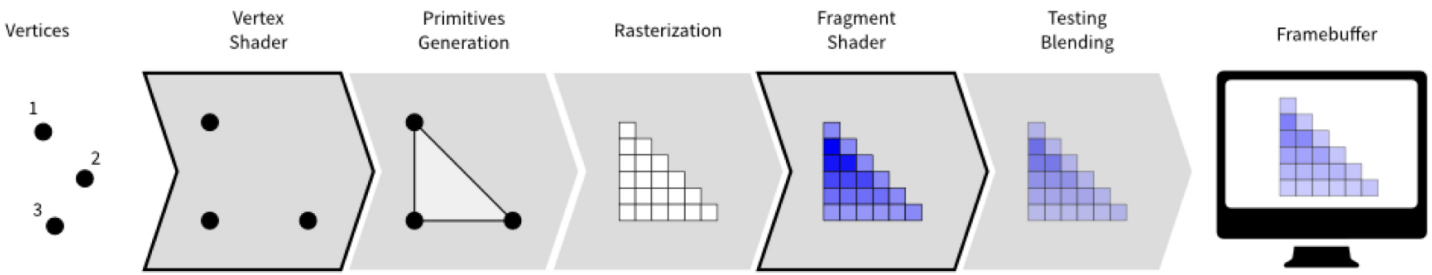




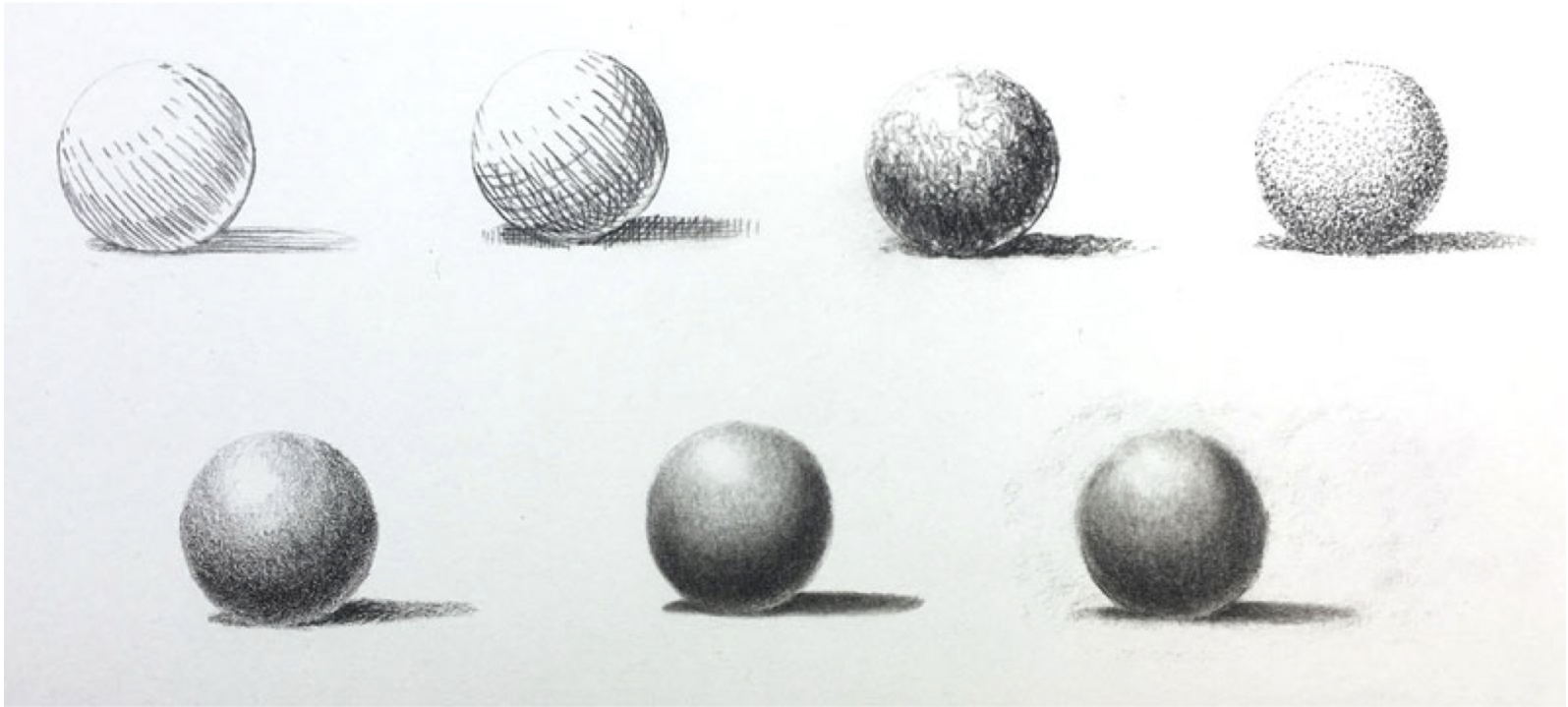
Aufbau und Funktionsweise einer GPU

- **Architektur:** Viele einfache Rechenkerne in *Streaming Multiprocessors (SM)*, optimiert für parallele Berechnung.
- **Speicherhierarchie:**
  - *Global Memory (VRAM)*: Großer, langsamer Speicher für die gesamte GPU.
  - *Shared Memory*: Schneller Zwischenspeicher pro SM.
  - *Register*: Klein, extrem schnell, lokal pro Kern.
- **SIMD-Prinzip:** Gleiche Operation auf mehrere Daten gleichzeitig (Single Instruction, Multiple Data).
- **Thread-Modell:** Threads in *Thread-Blocks* organisiert, diese in einem *Grid*.
- **Spezialisierte Hardware:** Rasterisierung, Texturierung und Shading für Grafikoperationen.

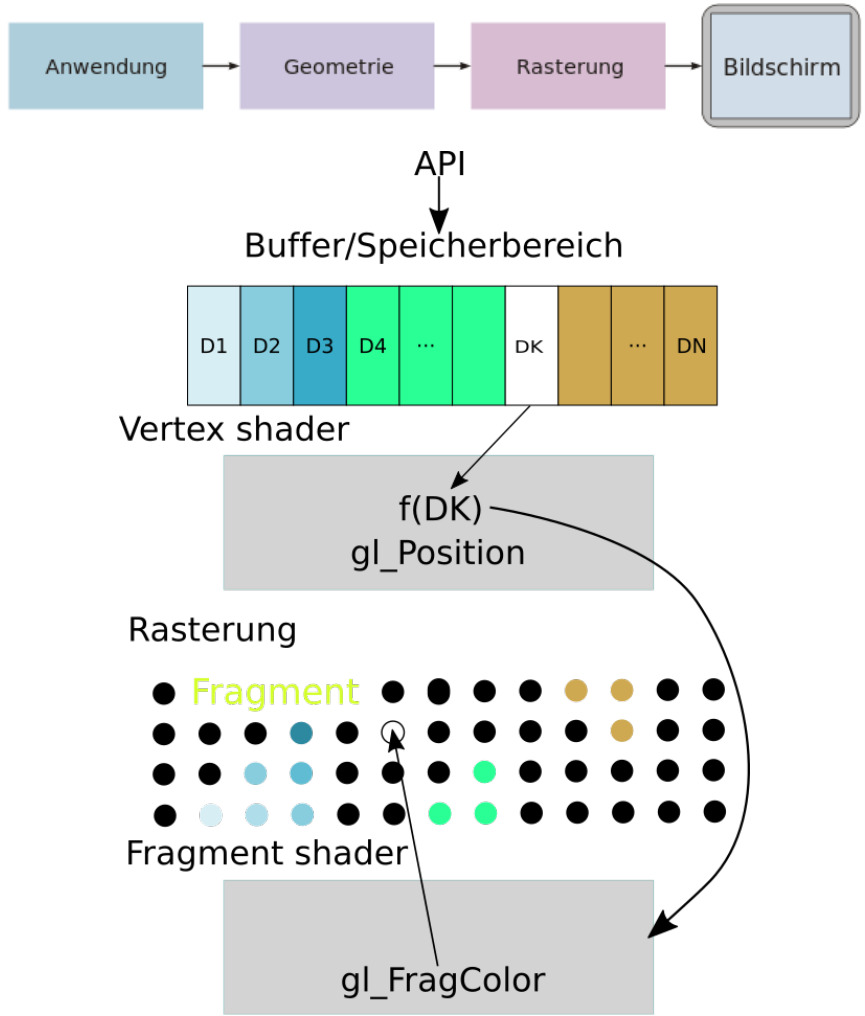
Vertex und Fragmentshader



Shader=Schattierer



Shaderprogramm



OpenGL Pipeline

```
<script id="2d-vertex-shader" type="x-shader/x-vertex">
    attribute vec2 a_position;
    uniform float t;
    varying float T;
    void main() {
    //  gl_Position = vec4(a_position, 0.0, t);
    T = t;
    gl_Position = vec4(a_position[0], a_position[1], 0.0, 1.0);
    }
</script>

<script id="2d-fragment-shader" type="x-shader/x-fragment">
    precision mediump float;
    varying float T;
    void main() {
    gl_FragColor = vec4(0.0 ,1.0,0.0,1.0);
    }
</script>
```