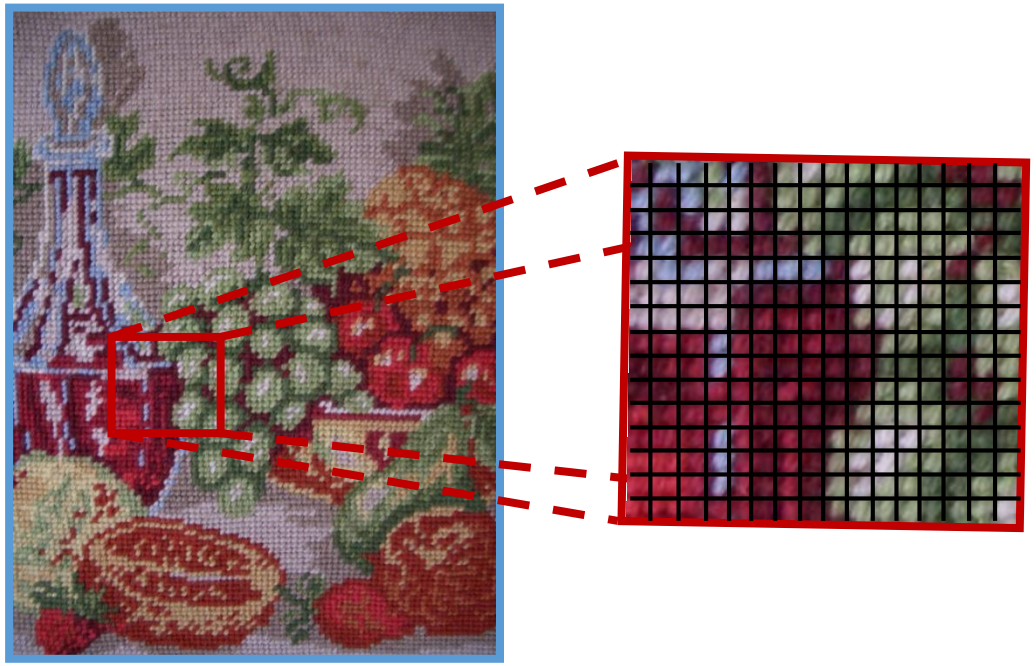
Rastergrafik und die OpenGL-Pipeline

Folien nach Prof. Reinhard Klein

Bildraster

- Speicherung von Bildern als Bildpunktmatrix
  - Feste Informationsmenge pro Bildpunkt
  - Kompatibel zu Fernsehbildern



Rastergrafik

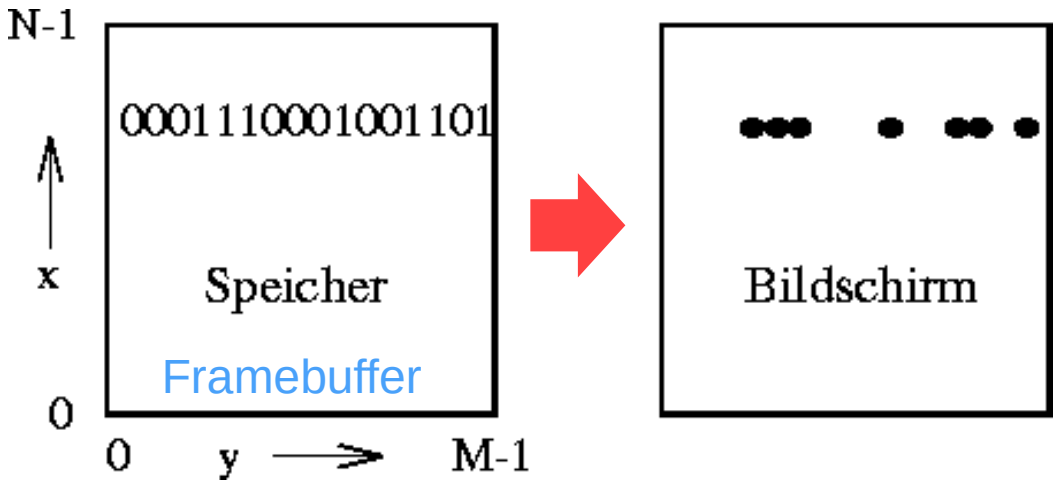
Daten pro Bildpunkt:

- **z.B. Farbe**
  - Binär/Bitmap (0 oder 1) – 1bit/px
  - Index auf Farbpalette (vgl. Stickmuster) – 2-8bit/px
  - Vollfarbe (RGB) – 16-48bit/px
  - Spektraldaten

Quiz: Tintenstrahldrucker (CMYK)?

- **Transparenz** (alpha, optional)
- **Tiefeninformation** (Z/depth, optional)
- und vieles mehr ...

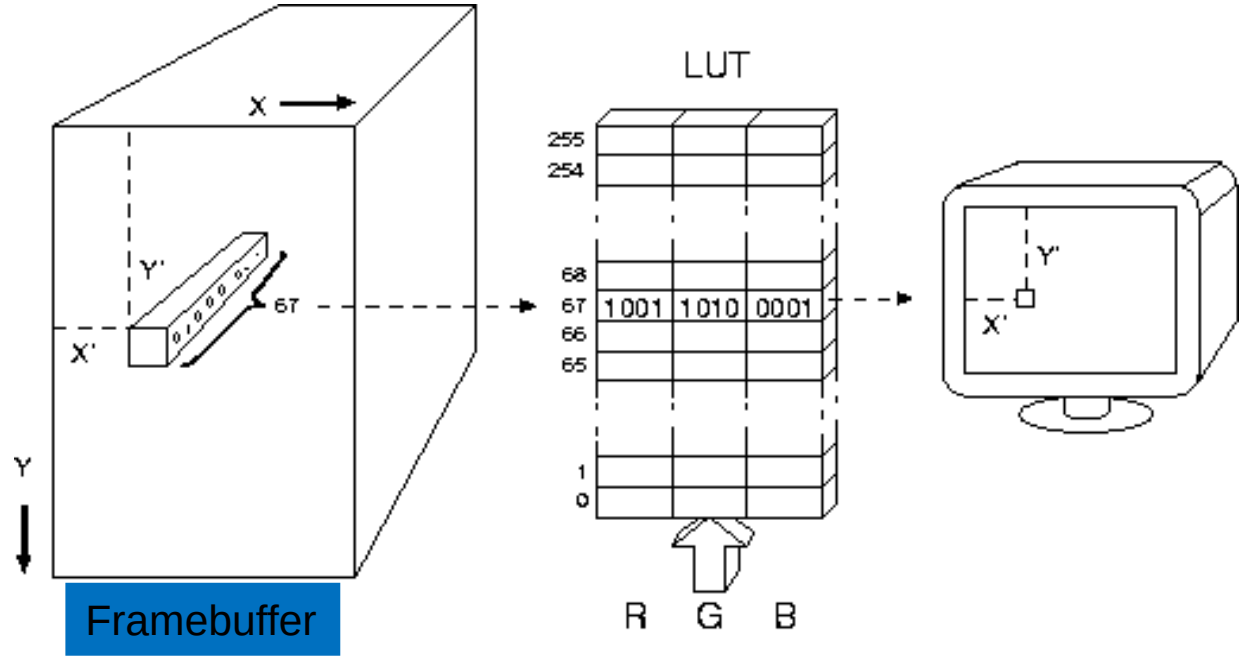
Map-Display



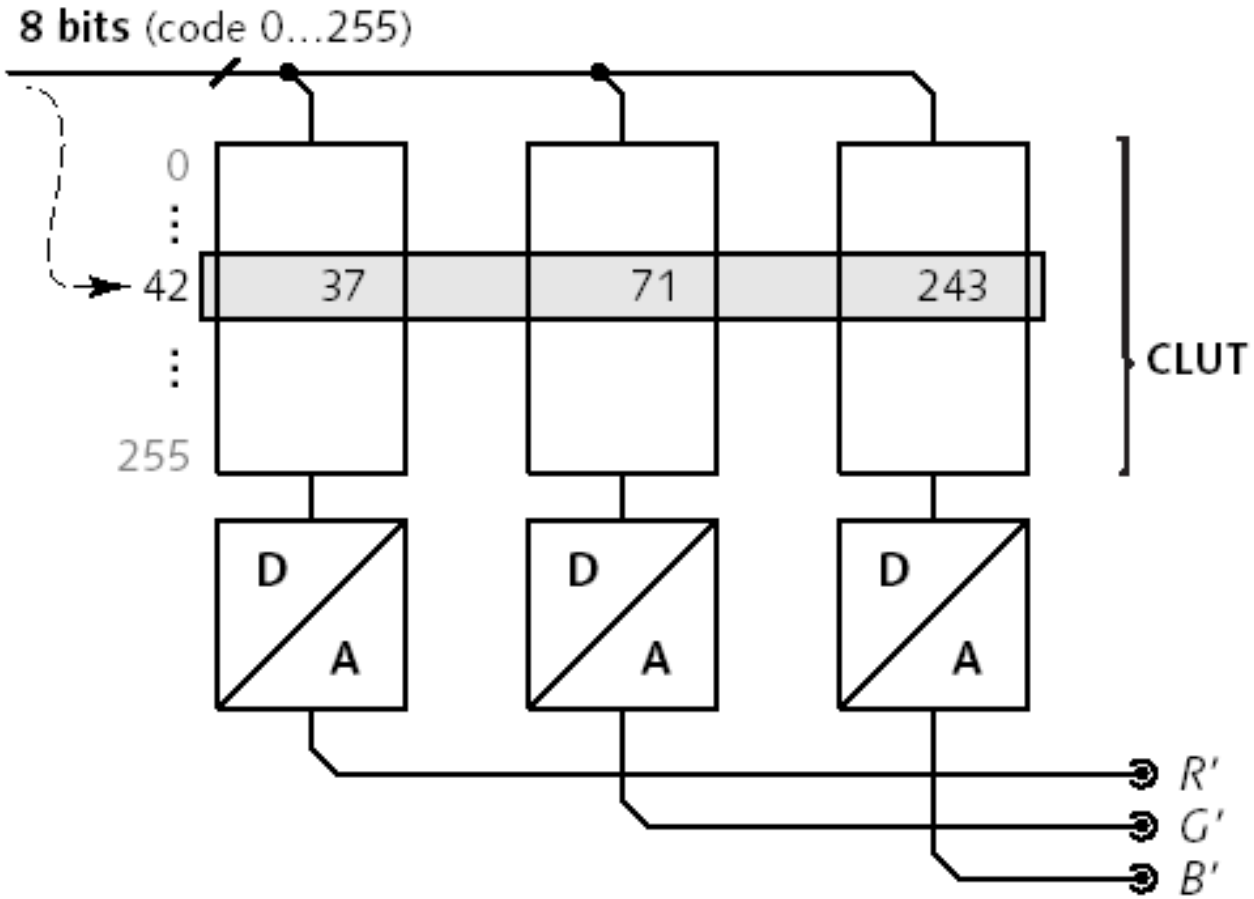
- Der Framebuffer wird mit einer bestimmten Frequenz neu befüllt und ausgelesen.
- Für Echtzeitgrafik werden mehr als 25 Bilder/Sekunde benötigt

Farbtafel (Lookup Table, LUT)

Pixelwert steuert meist nicht direkt die Intensität oder Farbe, sondern wird als Adresse für eine Farbtafel (LUT) interpretiert. Die Tafeleinträge definieren dann die auszugebende Farbe.

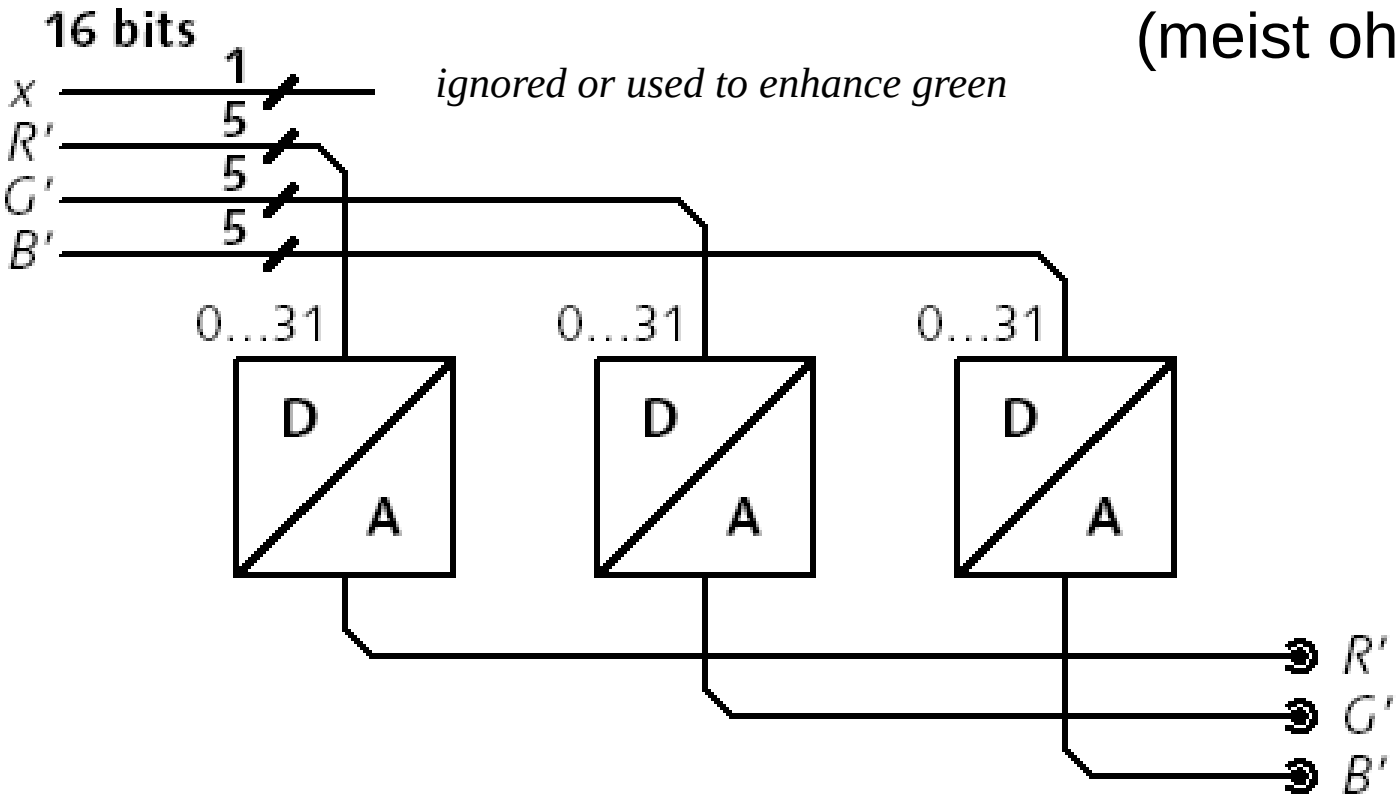


Farbrepresentationen Pseudo-Color (8bit)

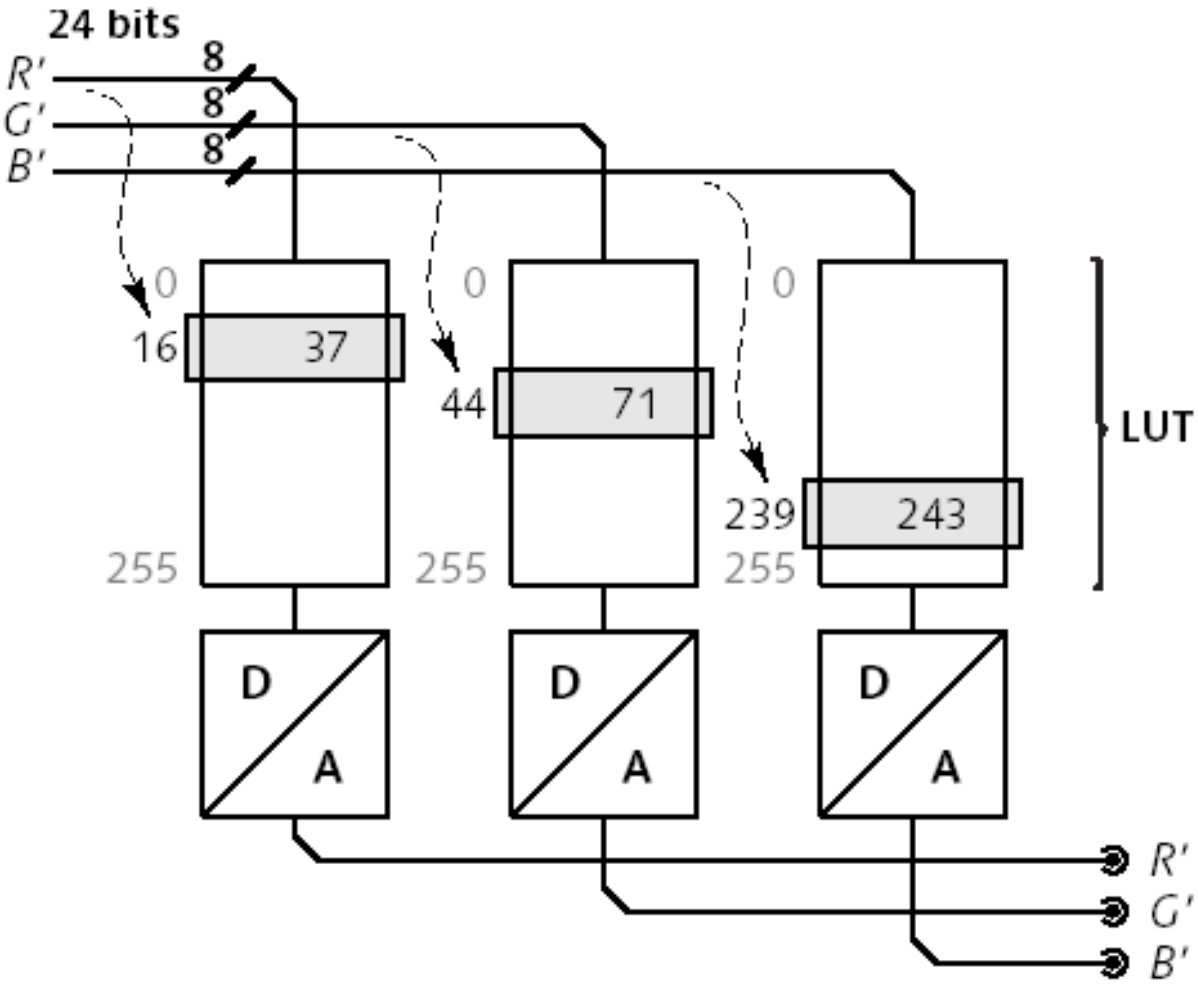


Farbrepresentationen Hi-Color (16bit)

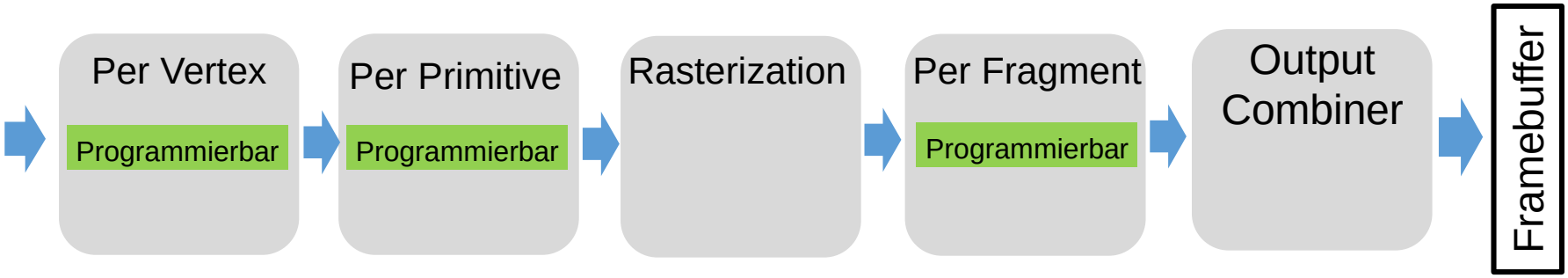
(meist ohne LUT)

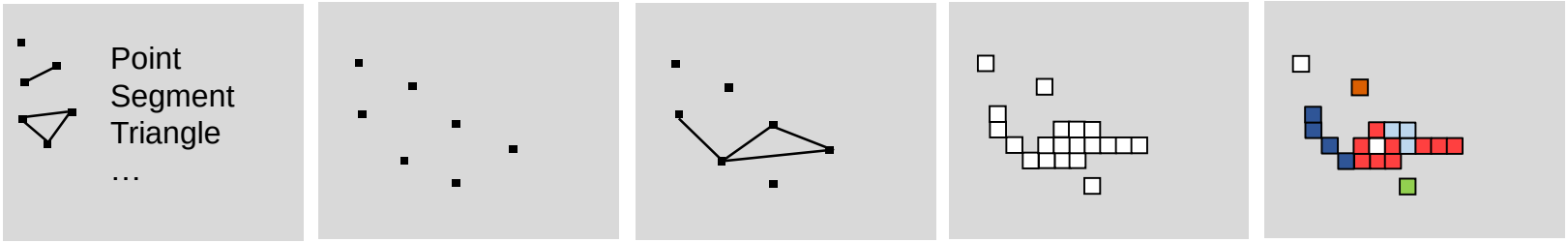


Farbrepresentationen True-Color (24bit)

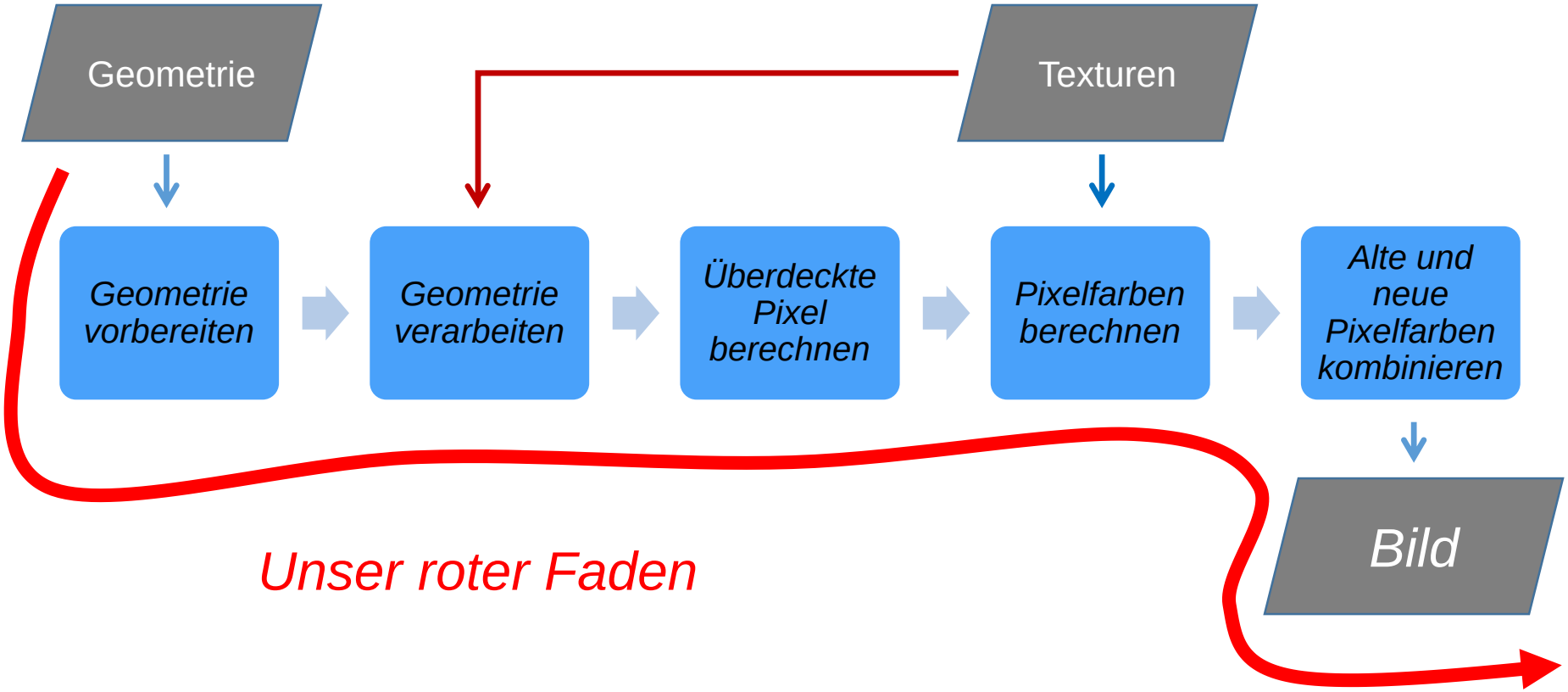


Abstrakte Rasterisierungs-Pipeline



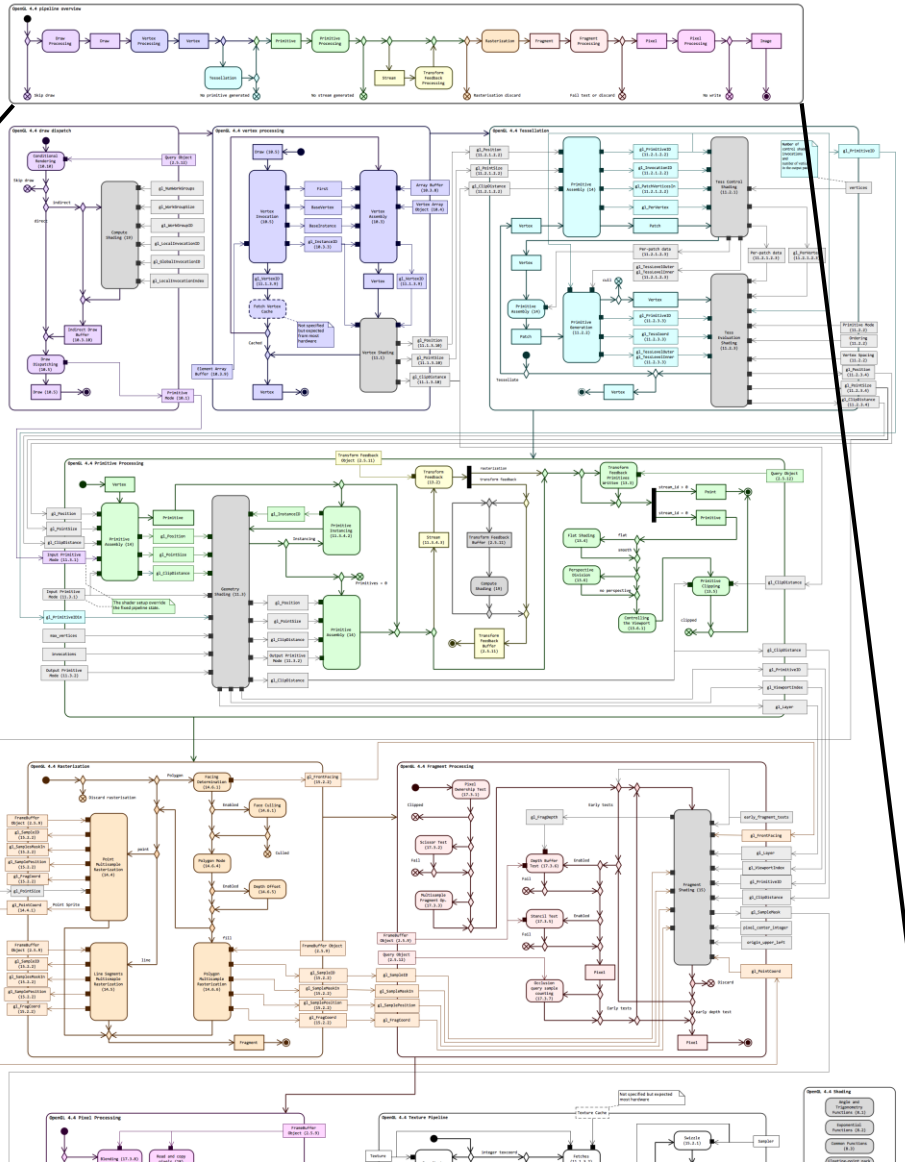


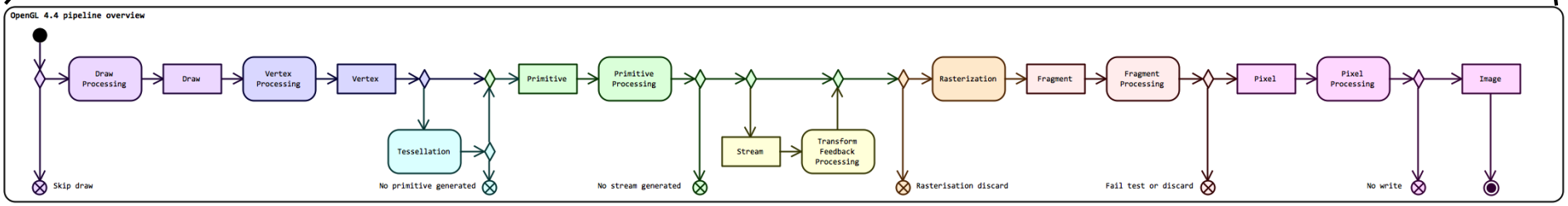
Abstrakte Rasterisierungs-Pipeline



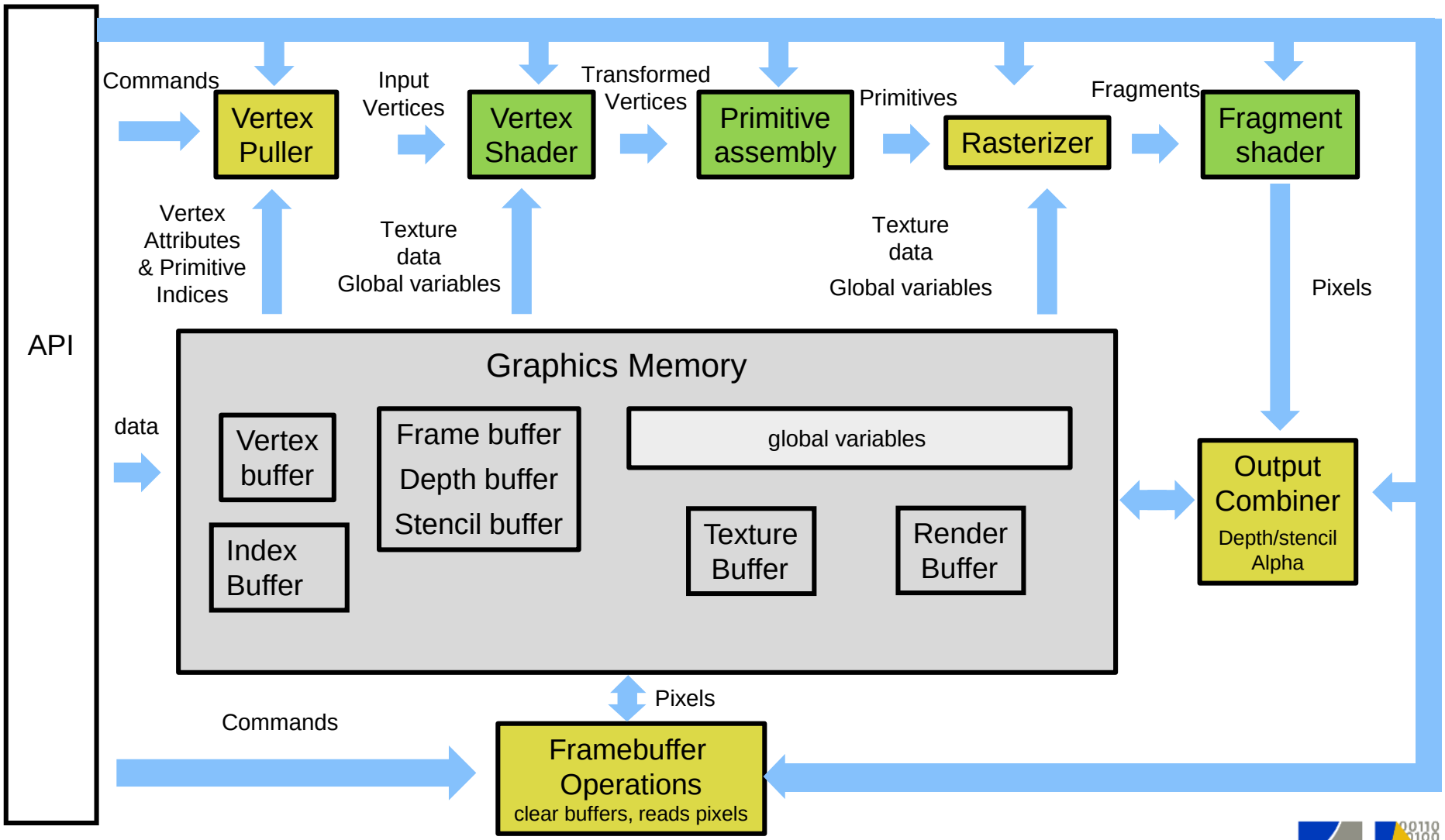
OpenGL 4.4 Pipeline

Quelle: openglinsights.com

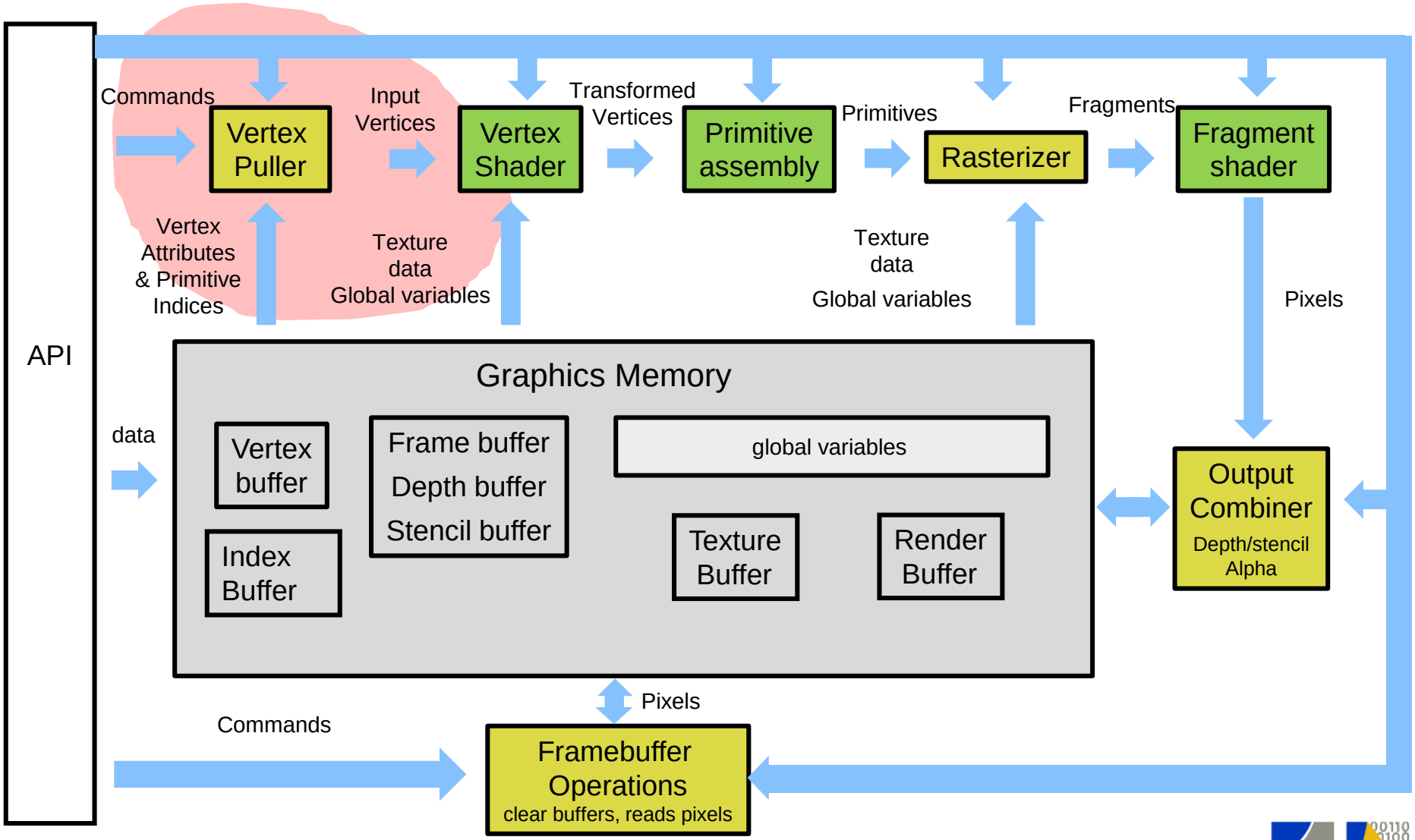




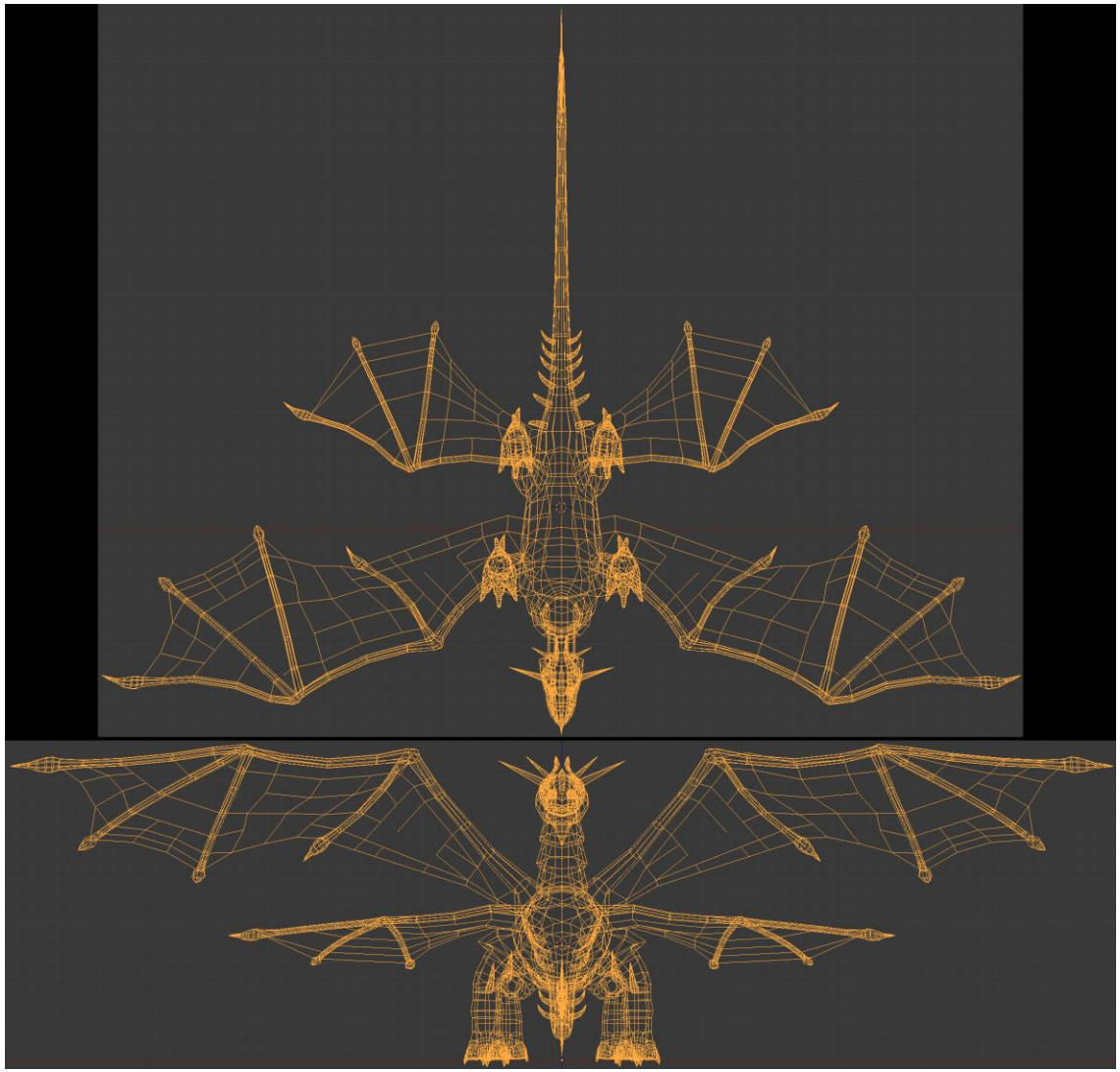
OpenGL-Pipeline (vereinfacht)



OpenGL-Pipeline

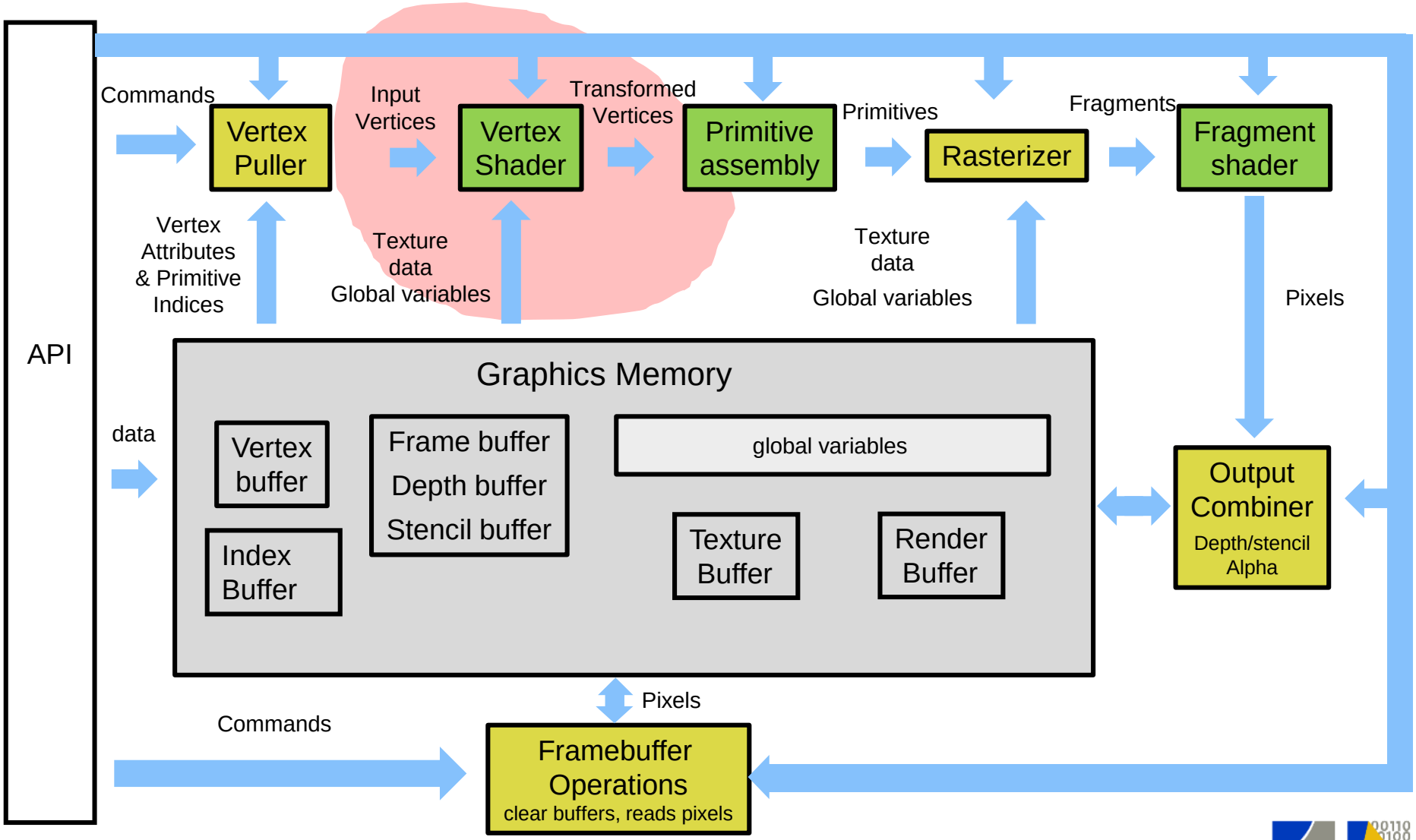


OpenGL-Pipeline

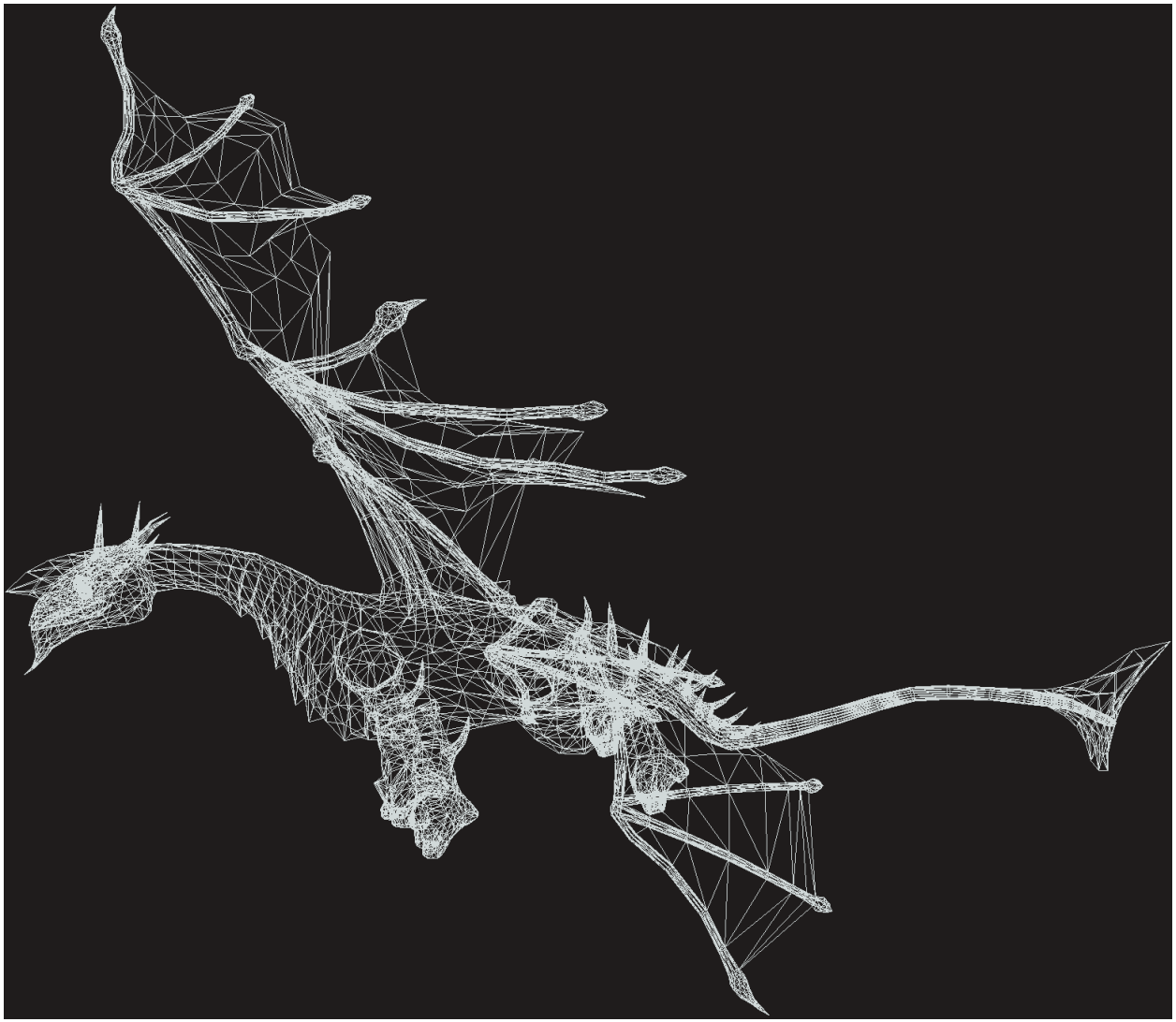


Post Vertex Puller

OpenGL-Pipeline

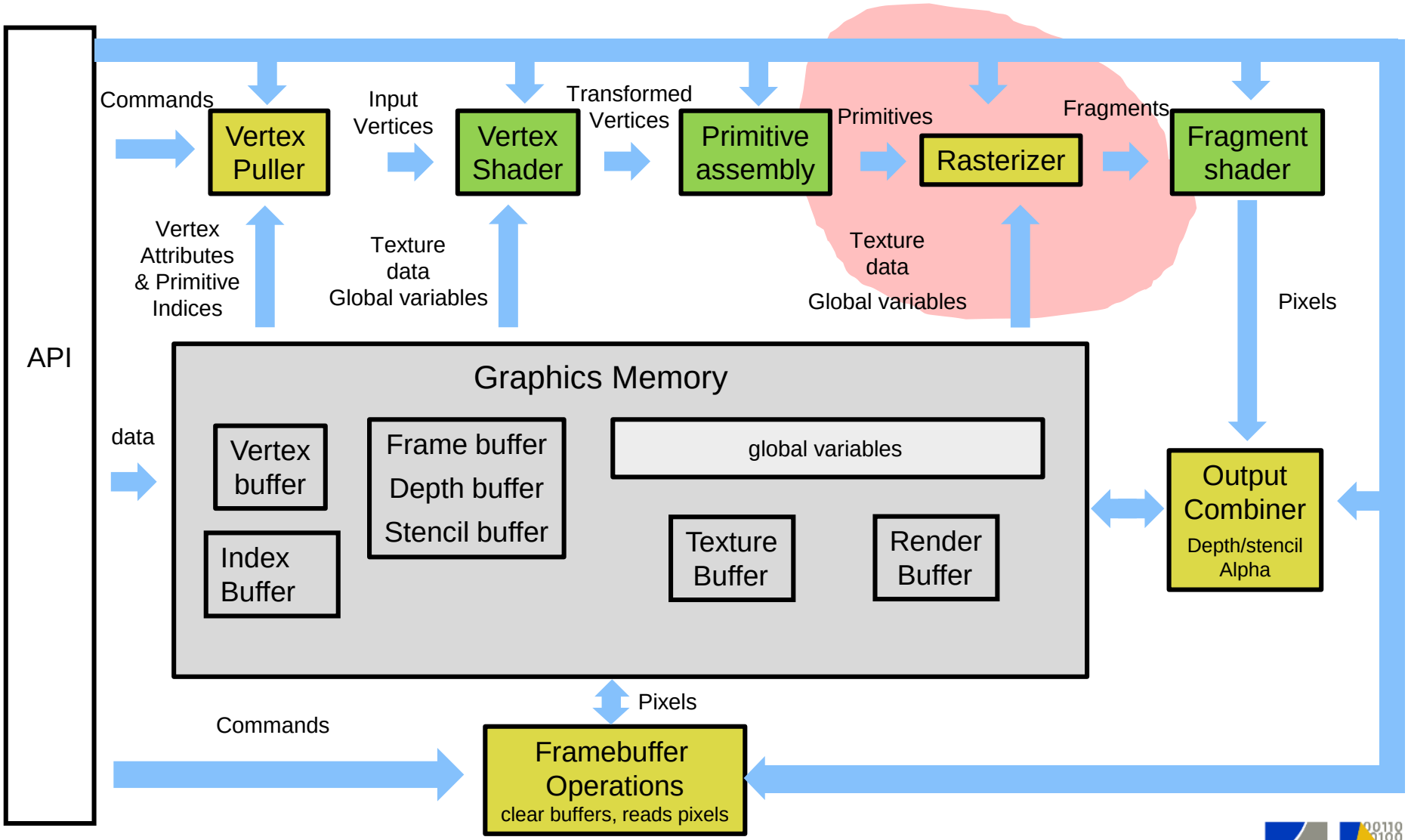


OpenGL-Pipeline



Post Vertex Shader

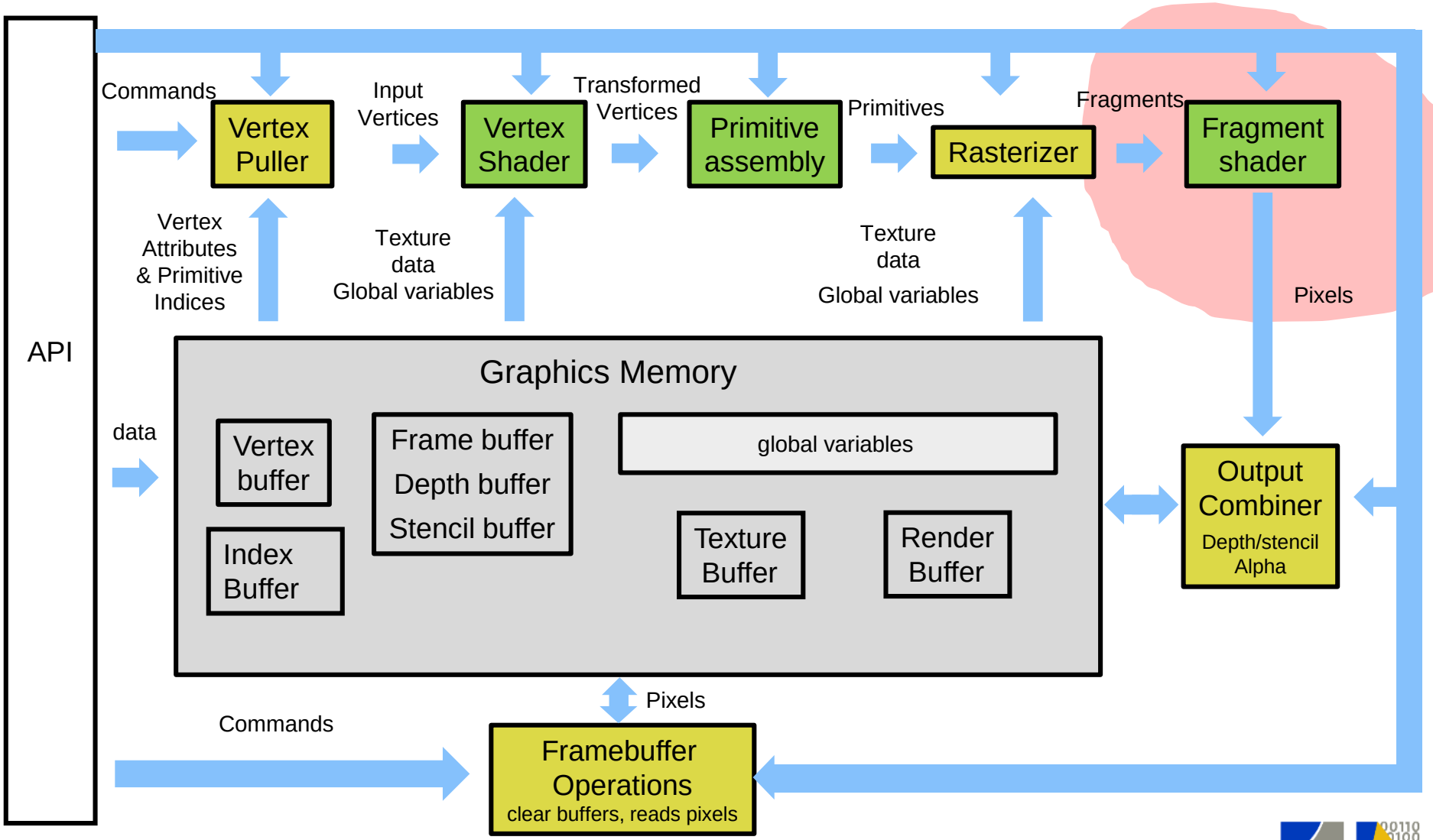
OpenGL-Pipeline



OpenGL-Pipeline

Post Rasterizer

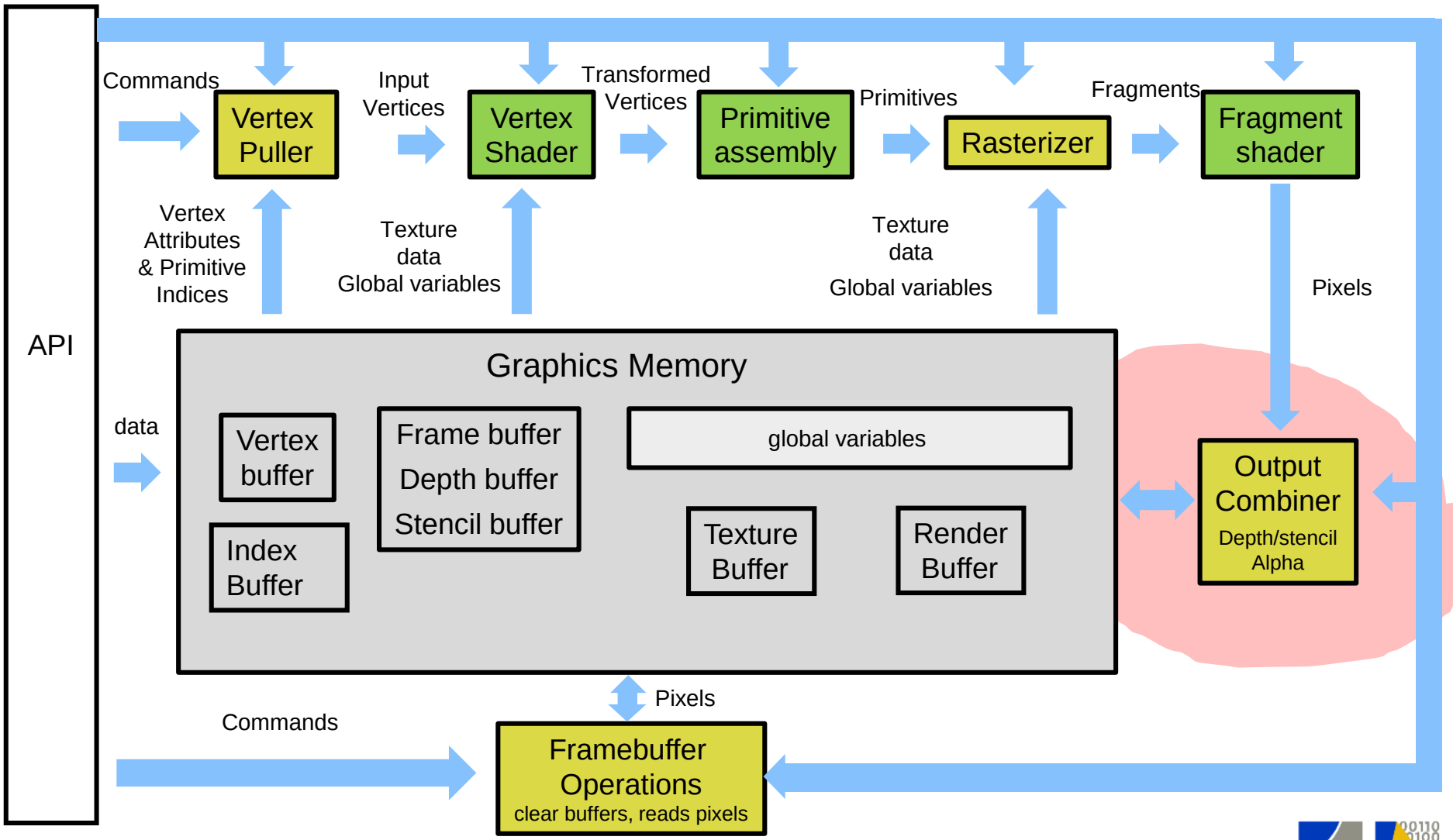
OpenGL-Pipeline



OpenGL-Pipeline

Post Fragment Shader

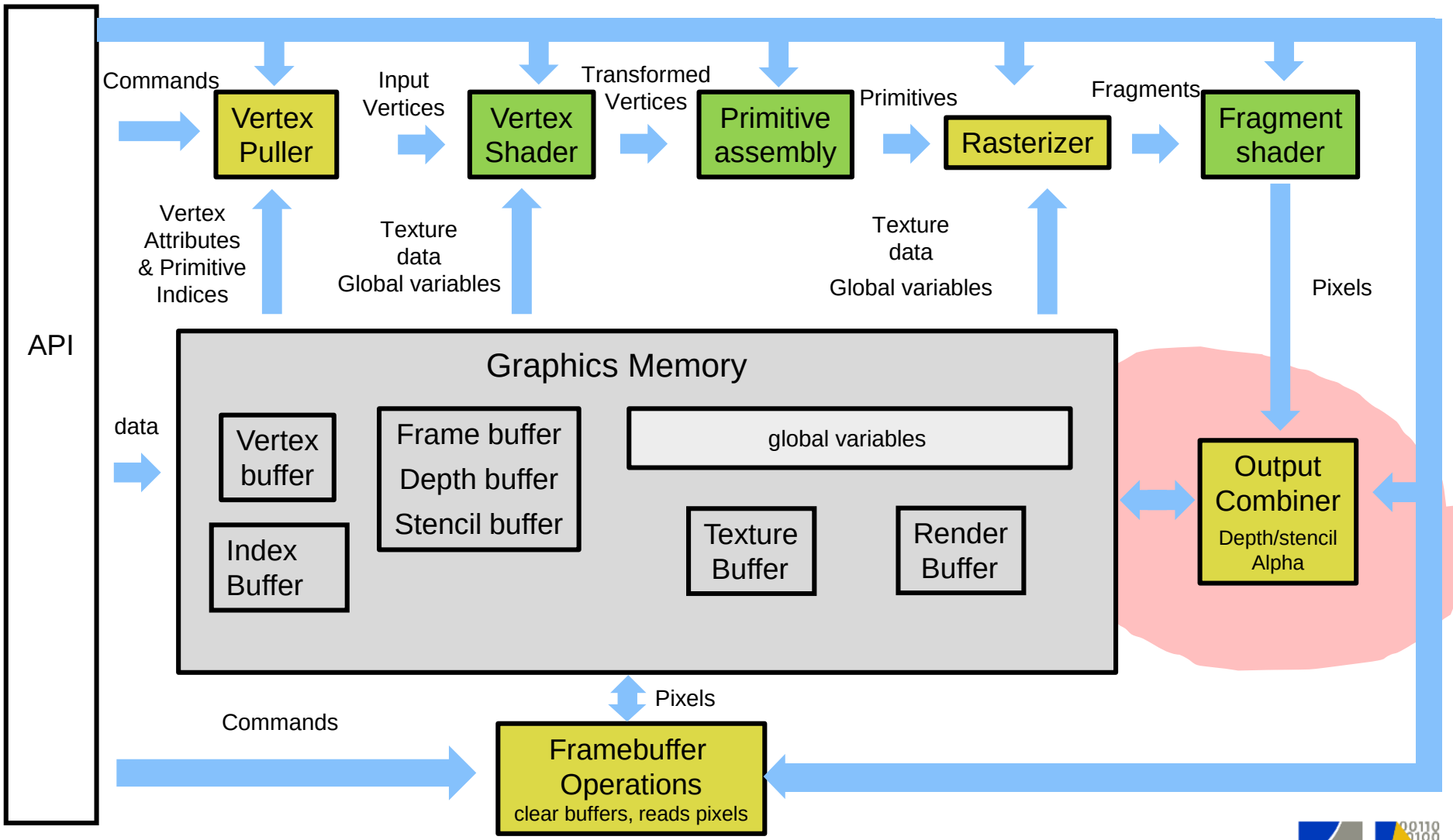
OpenGL-Pipeline



OpenGL-Pipeline

Post Output Combiner

OpenGL-Pipeline

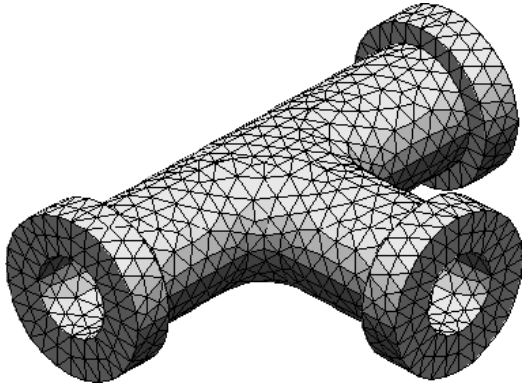


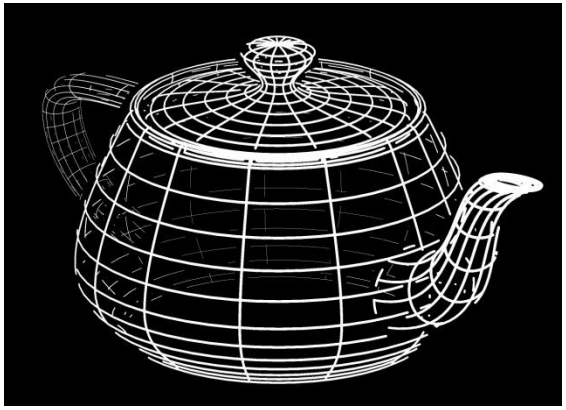
OpenGL-Pipeline

Post Output Combiner (2)

Geometrierepräsentationen

- Oberflächenrepräsentationen von Objekten
  - Polyeder, Dreiecke, Vierecke (Quads)

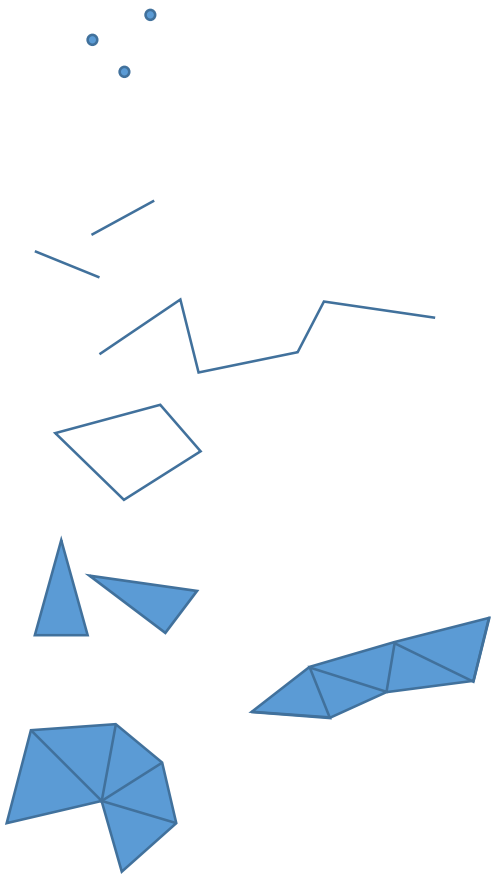




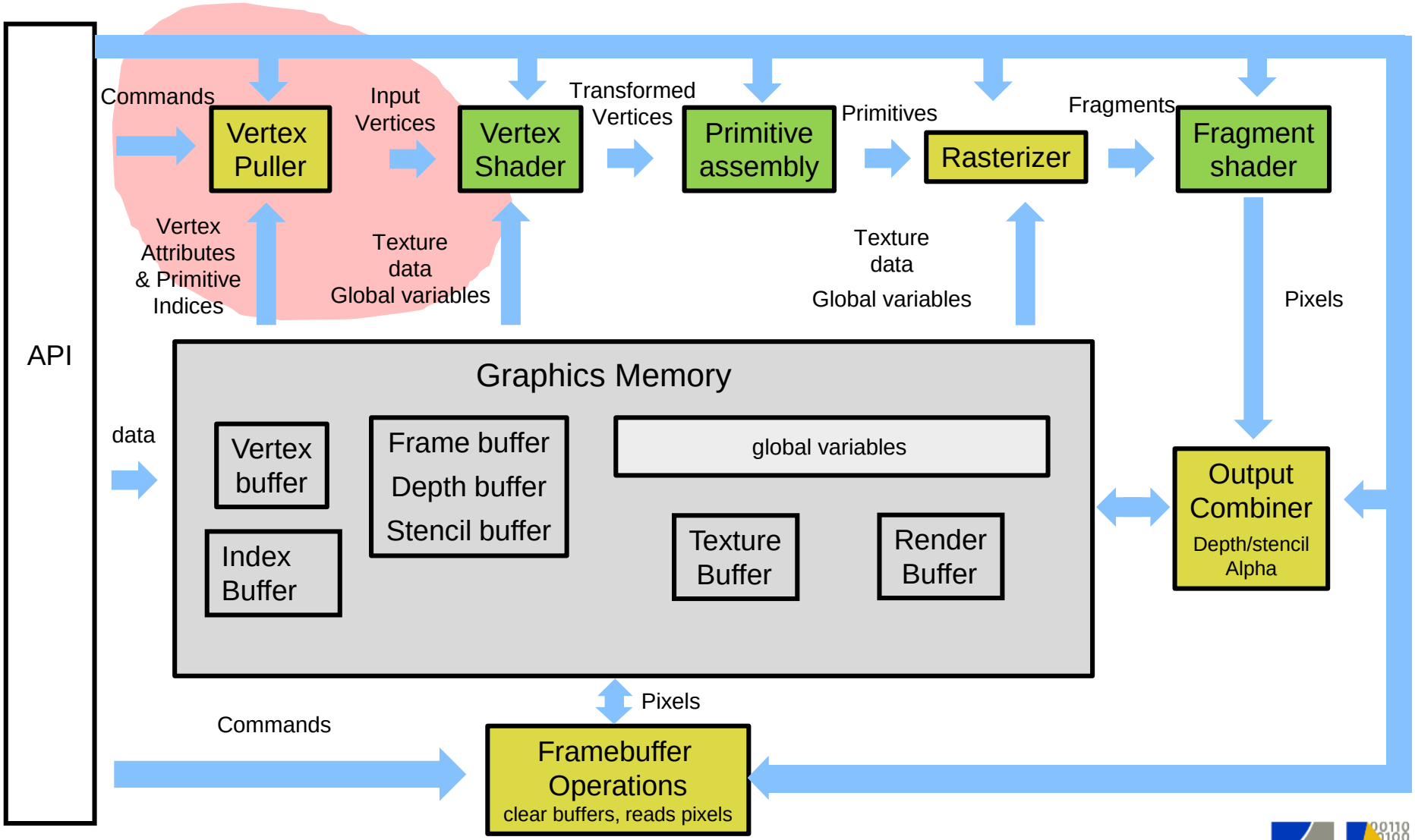
- Wie werden diese Objekte in OpenGL repräsentiert?

OpenGL-Primitives

- Punkte / Points
  - GL_POINTS
- Linien / Lines
  - GL_LINES
  - GL_LINE_STRIP
  - GL_LINE_LOOP
- Dreiecke / Triangles
  - GL_TRIANGLES
  - GL_TRIANGLE_STRIP
  - GL_TRIANGLE_FAN
- ~~Vierecke / Quads~~
  - ~~GL_QUADS~~
  - ~~GL_QUAD_STRIP~~



OpenGL-Pipeline



Vertex Puller

- Drei wichtige Arten von Objekten liefern Geometriedaten für die Pipeline:
- Vertex Array Object (VAO)
- Vertex Buffer Object (VBO)
  - Array von Vertexattributen
- (optional) Element Buffer Object (EBO)
  - Array von Vertexindizes (zur Erzeugung von indizierten Primitives)

Geometrie an die Pipeline schicken

Ziehe Vertexdaten aus VBOs:

```
glDrawArrays (GL_TRIANGLES, .....);
glDrawArrays (GL_TRIANGLE_STRIP, .....);
glDrawArrays (GL_TRIANGLE_STRIP, .....);
glDrawArrays (GL_TRIANGLE_FAN, .....);
glDrawArrays (GL_TRIANGLE_STRIP, .....);
glDrawArrays (GL_TRIANGLE_FAN, .....);
```

Viele "Draw Calls"; Vertexdaten nicht gecacht. Daher: lieber Strips und Fans mittels EBO in indizierte Dreiecke verwandeln, und dann ein gemeinsamer Aufruf:

```
glDrawElements (GL_TRIANGLES, .....);
```

Vorteile:

- weniger Draw Calls
- Vertices wiederverwenden – OpenGL prüft Cache über Indizes