FH Bielefeld
University of
Applied Sciences
Campus Minden

# Einstieg in OpenGL 4

Dr.-Ing. Christoph Fünfzig

OpenGL ist eine Spezifikation *(Khronos.org)* einer plattform- und programmiersprachen-übergreifende Programmierschnittstelle *(API)* zur Entwicklung von 2D- und 3D-Computergrafik-anwendungen.

*https://www.khronos.org/registry/OpenGL-Refpages/gl4/*

*https://www.khronos.org/developers/reference-cards/*

# Pipeline


Bild von http://www.carbodydesign.com/gallery/2011/01/design-story-the-fiat-uno-cabrio-concept/10/

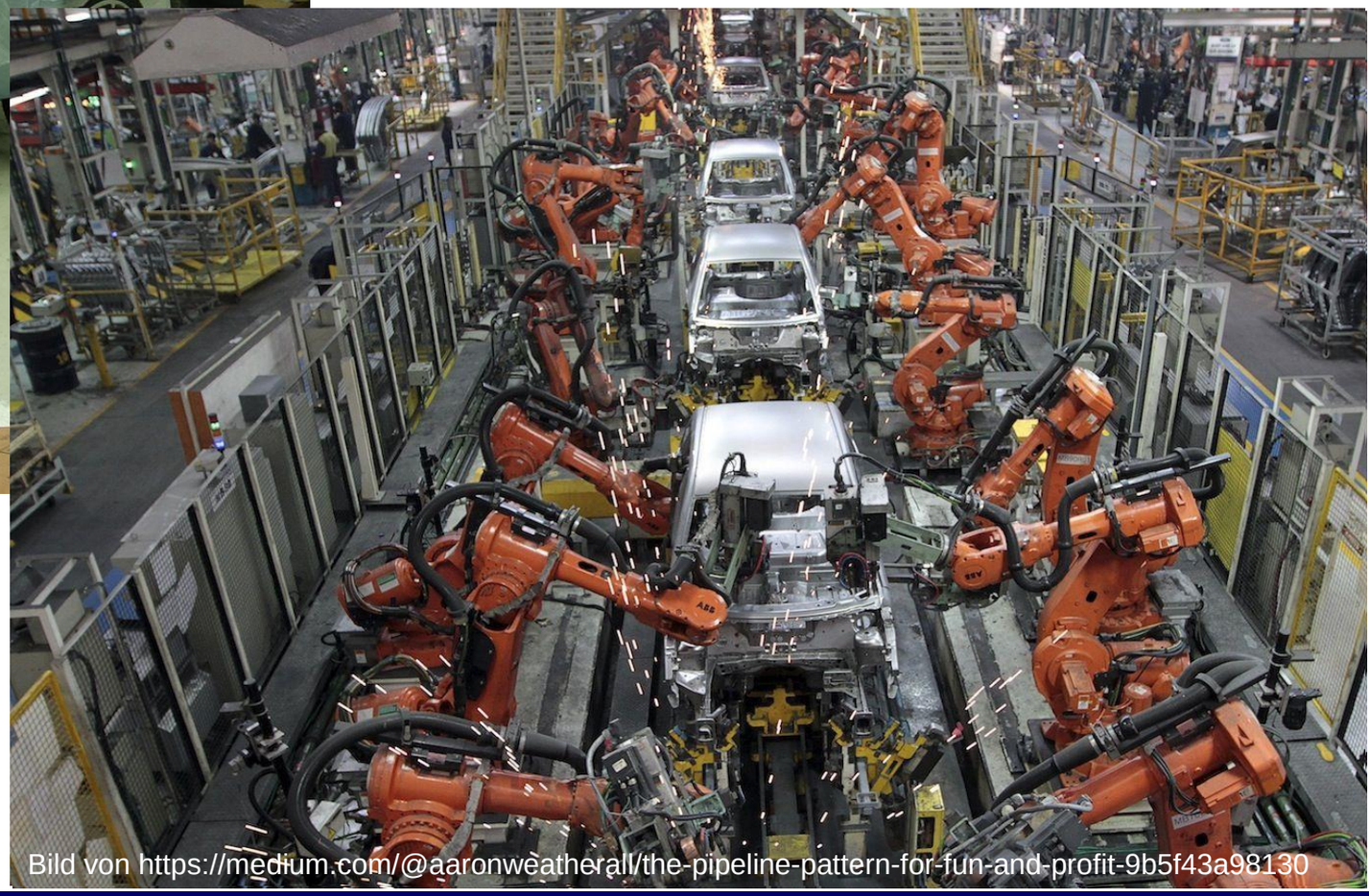


Bild von https://medium.com/@aaronweatherall/the-pipeline-pattern-for-fun-and-profit-9b5f43a98130

# OpenGL Pipeline

- Rasterisierung durch OpenGL

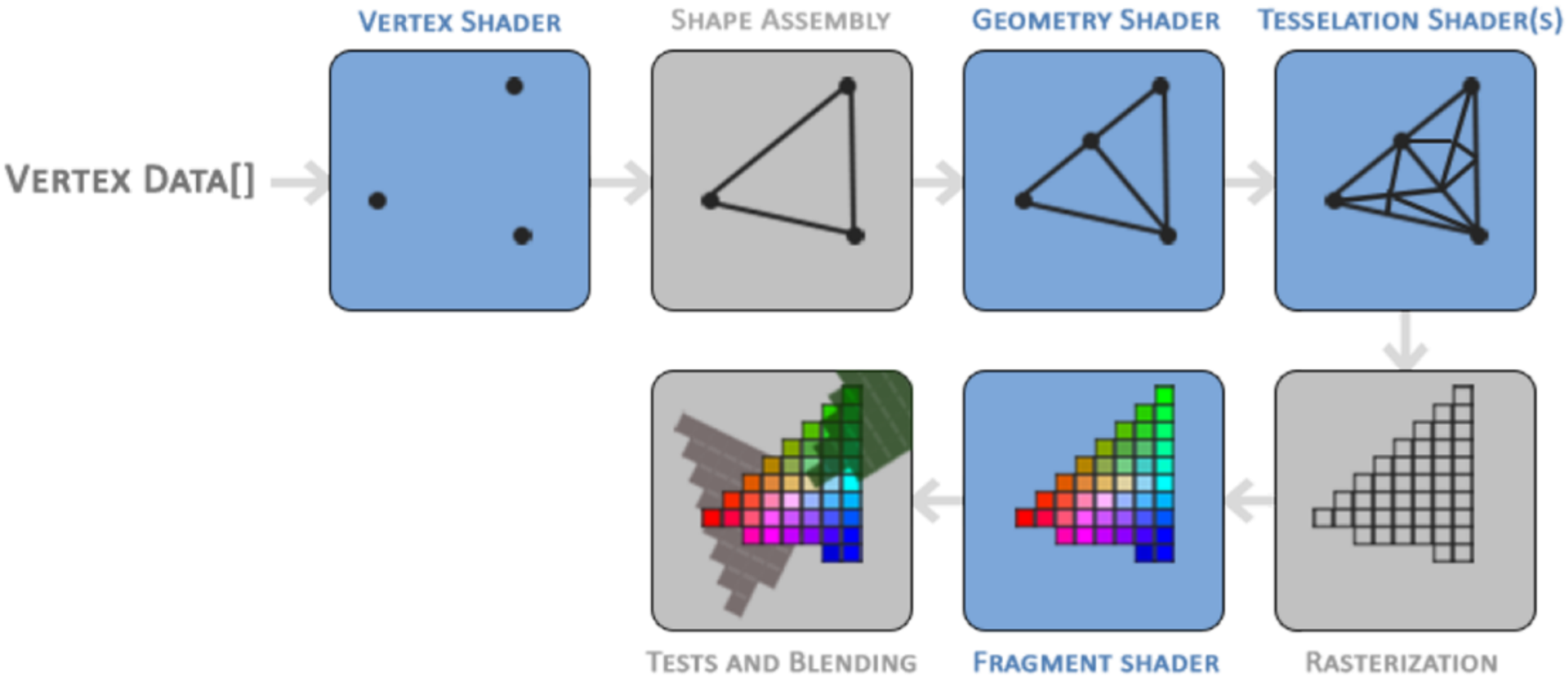



Bild von https://learnopengl.com/

# OpenGL Versionen

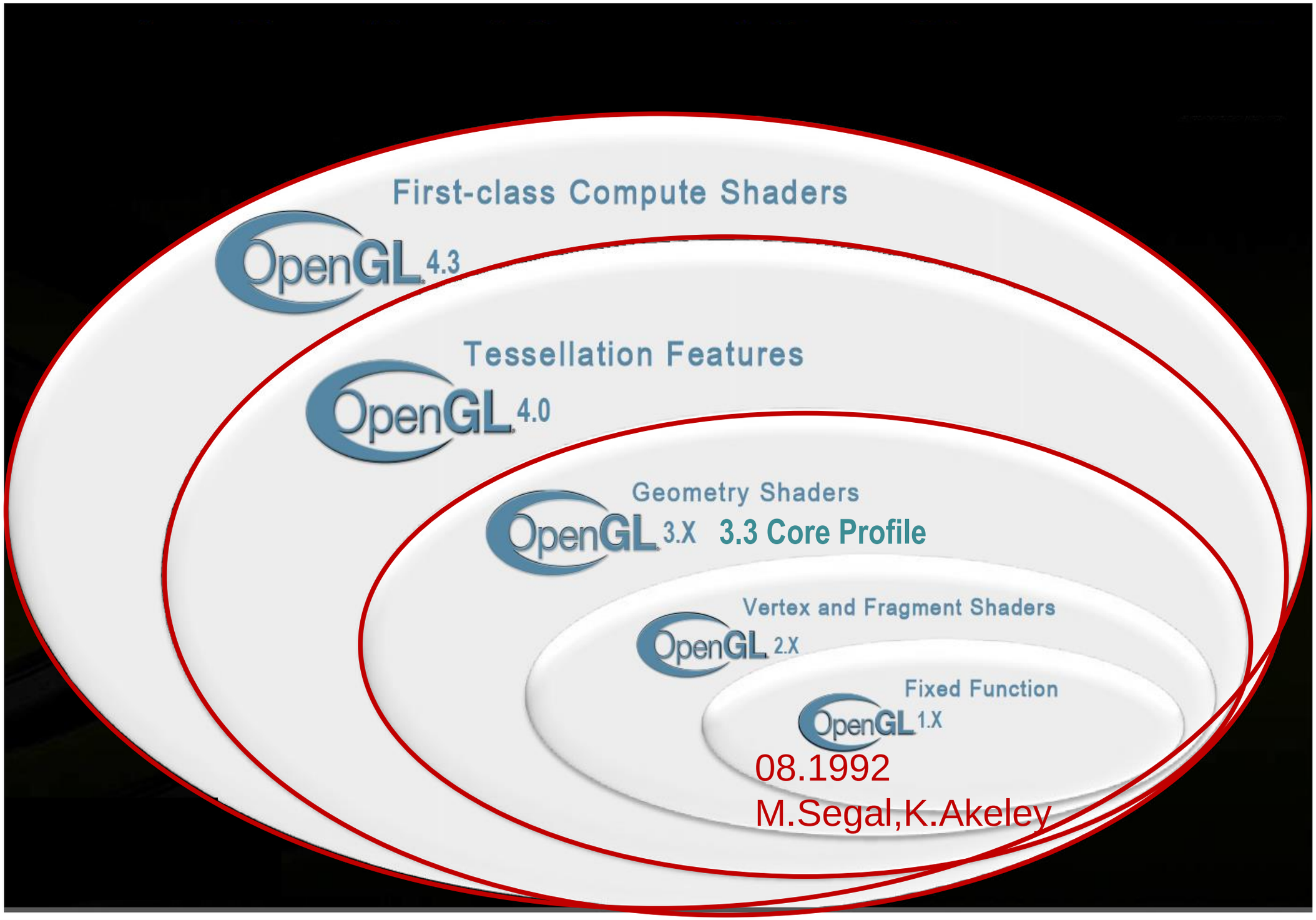
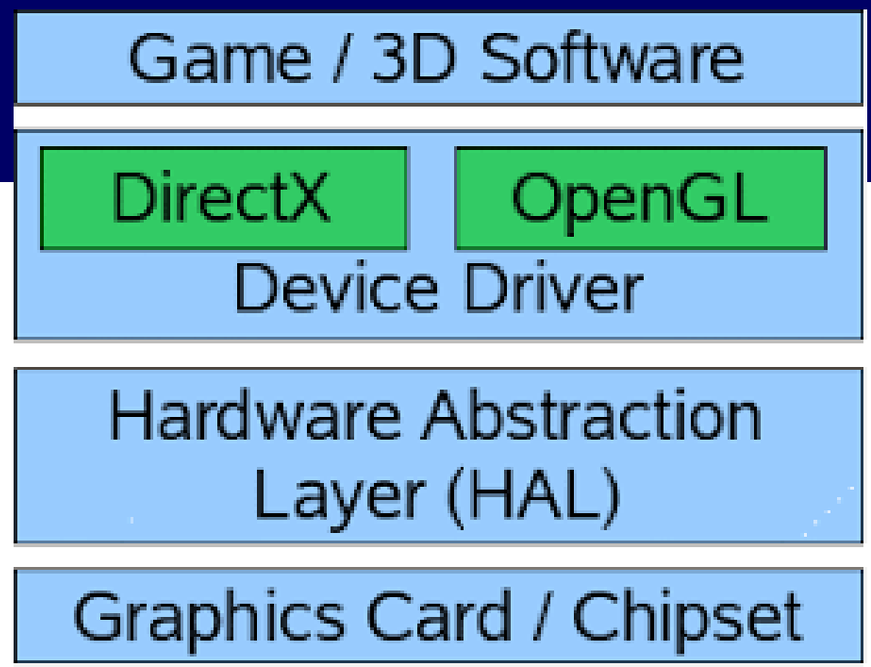



Bild von NVidia




# OpenGL Architektur

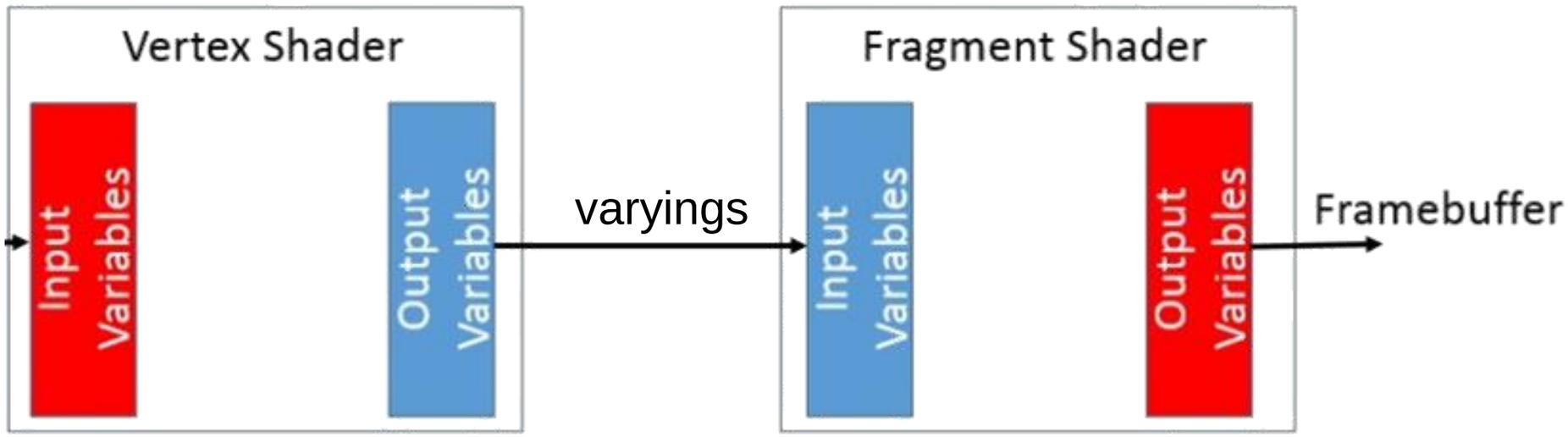
- OpenGL ist C Bibliothek
- Client (App)-Server (GPU) System:
  OpenGL Kontext hält Zustand und Datenobjekte
- Verschiedene Versionen, wie geht das?
  GLEW: Funktionssammlung vom OGL Treiber
- FREEGLUT/GLFW öffnet Fenster mit OpenGL Kontext
  - Gewünschte Version (bei uns: OpenGL 4.3 Core Profile)
  - Renderschleife mit Ereignisverarbeitung (Windows, Linux/X, MacOS/X)

# Welche OpenGL Version auf ihrem Rechner?

- OpenGL Extensions Viewer
  http://realtech-vr.com/admin/glview
- GPU Caps Viewer
  http://www.ozone3d.net/gpu_caps_viewer/
- GLEW
  Blatt01\libs\glew\bin\vs2015_x64\Release\glewinfo.exe

# Ohne Shader kein Bild ;-)

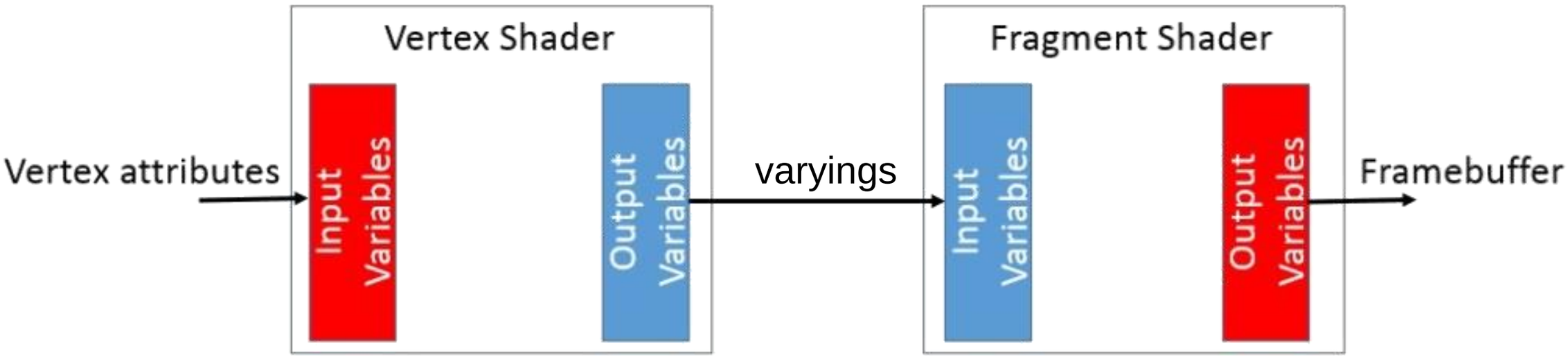
- simple.vert, simple.frag
  Compiler paart out-Variablen des VS mit in-Variablen des FS



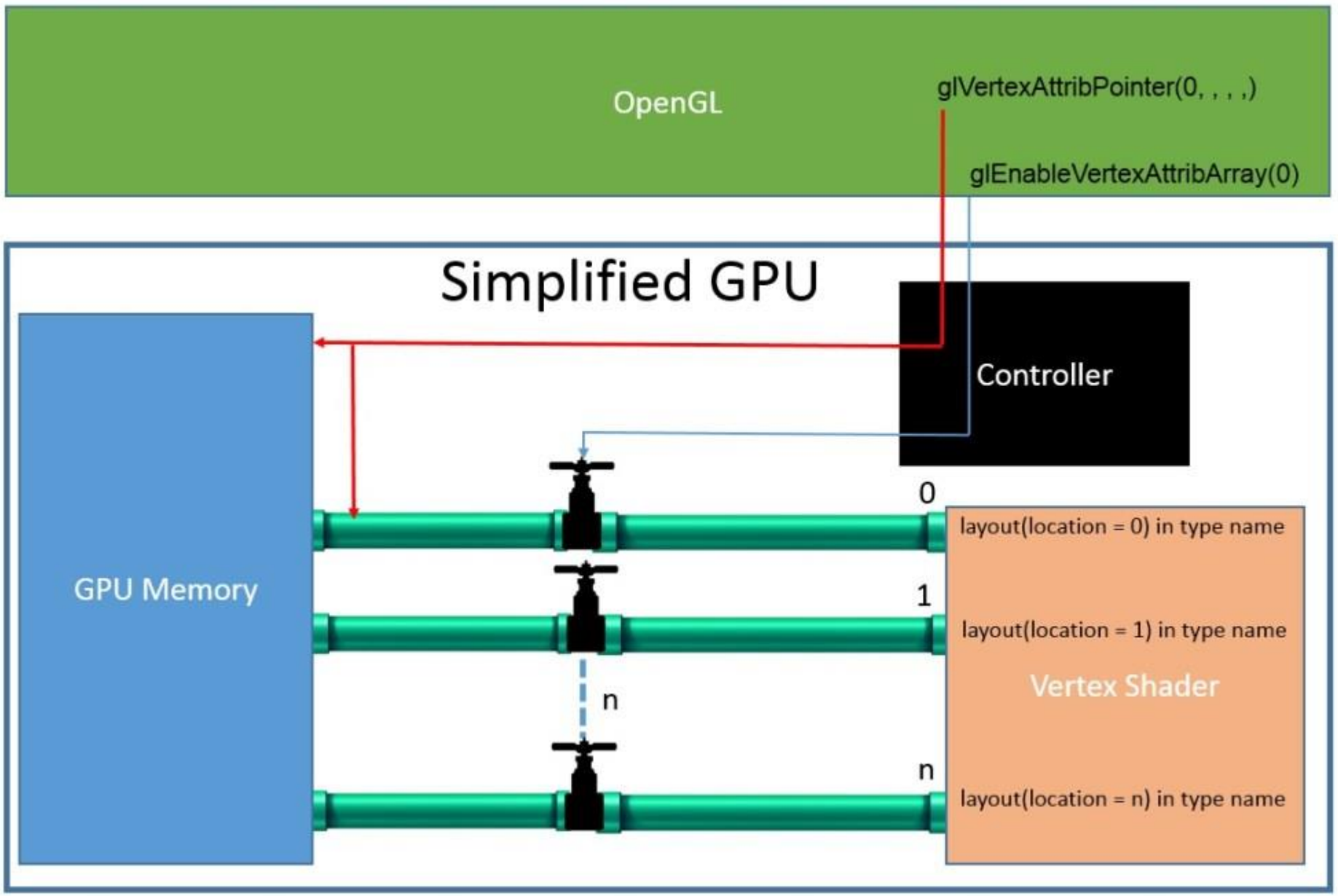

out fragmentColor
in fragmentColor
out fragColor

# in-Variablen des Vertex Shader (Attribute)

- simple.vert, simple.frag
  Vertex-Attribute: (position, color, ..) sind Eingabe des VS



# in-Variablen des Vertex Shader (Attribute)



# Vertex Buffer Object (OpenGL 3.3)

- Buffer sind Arrays auf der GPU
- Belegung mit Daten durch die CPU

Step 1: Create VBO

```
Gluint vbo; //declare
glGenBuffers(1, &vbo);//create
```

Step 2: Bind VBO

```
//Bind VBO to a binding point
//Binding point is GL_ARRAY_BUFFER
glBindBuffer(GL_ARRAY_BUFFER, vbo);
```

GL_ARRAY_BUFFER
Is the binding point for Vertex Attributes

Step 3:Allocate memory

```
glBufferData(GL_ARRAY_BUFFER, size, data, usage);
```

| | |
|---|---|
| size | 3*sizeof(VertexFormat)<br>VertexFormat->Position, Position is a Vertex attribute |
| data | Our 3 vertices array with XYZ values |
| usage | How to use this buffer |

Step 4: Delete VBO

```
glDeleteBuffers(1, &vbo);
//When you don't use it anymore
```

# Vertex Buffer Object (OpenGL 3.3)

## Beispiel

```
glGenBuffers(1, &vbo);
glBindBuffer(GL_ARRAY_BUFFER, vbo);
glBufferData(GL_ARRAY_BUFFER, vertices.size()*sizeof(VertexFormat),
vertices.data(), GL_STATIC_DRAW);
```

GL_STATIC_DRAW, GL_DYNAMIC_DRAW, GL_STREAM_DRAW
GL_STATIC_READ, GL_DYNAMIC_READ, GL_STREAM_READ

Performanzhinweis (Häufigkeit: STATIC, DYNAMIC, STREAM, Verwendungsart: READ, DRAW) aus Applikations-Sicht!

# Vertex Buffer Object (OpenGL 3.3)

## Bindung der Einträge als Attribut 0

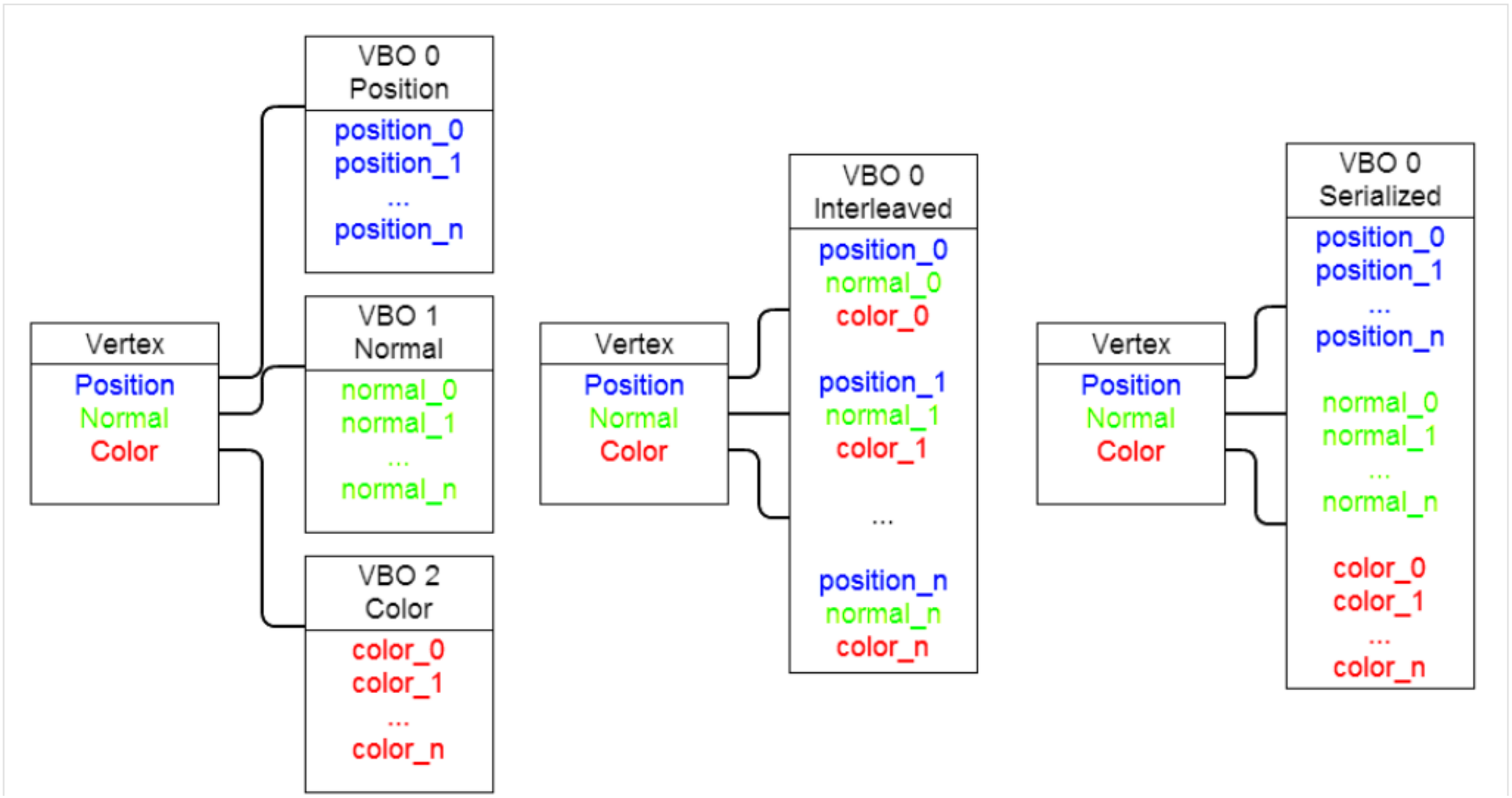
```
glUseProgram          (program0);
glBindBuffer          (GL_ARRAY_BUFFER, vbo);
glEnableVertexAttribArray(0);
glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE,    stride=0, (void*)0);
```

index=0, size=3, type=GL_FLOAT, normalized=GL_FALSE, stride=0, pointer=0

# glVertexAttribPointer?

3 Buffer (1 Buffer pro Attribut)

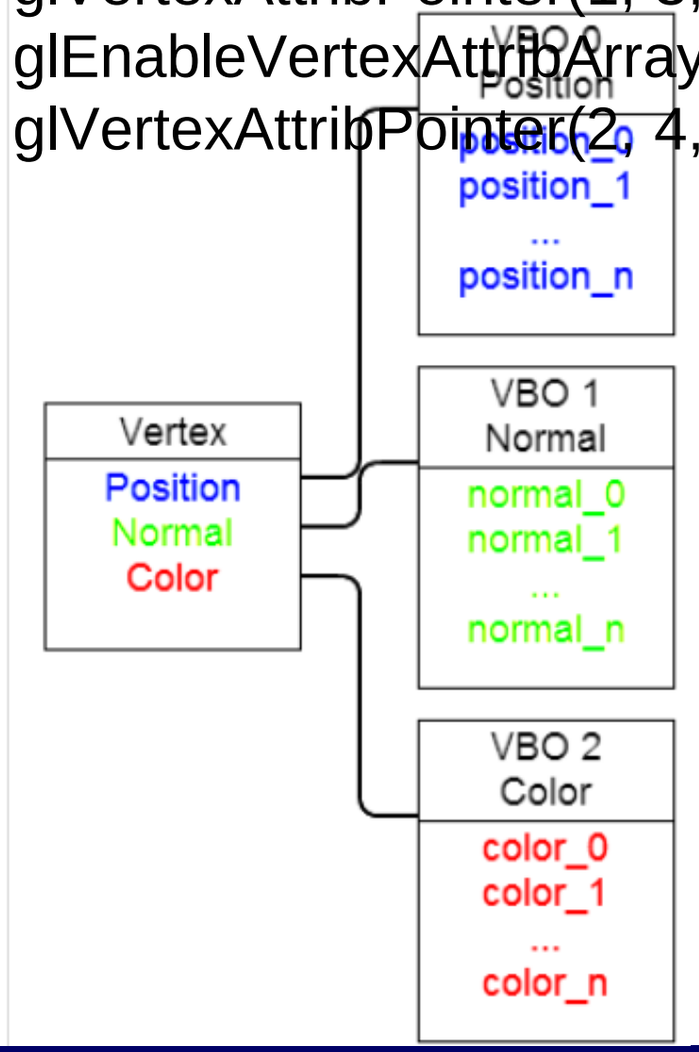
oder: 1 Buffer mit allen Attributen



# glVertexAttribPointer?

- 3 Buffer (1 Buffer pro Attribute)

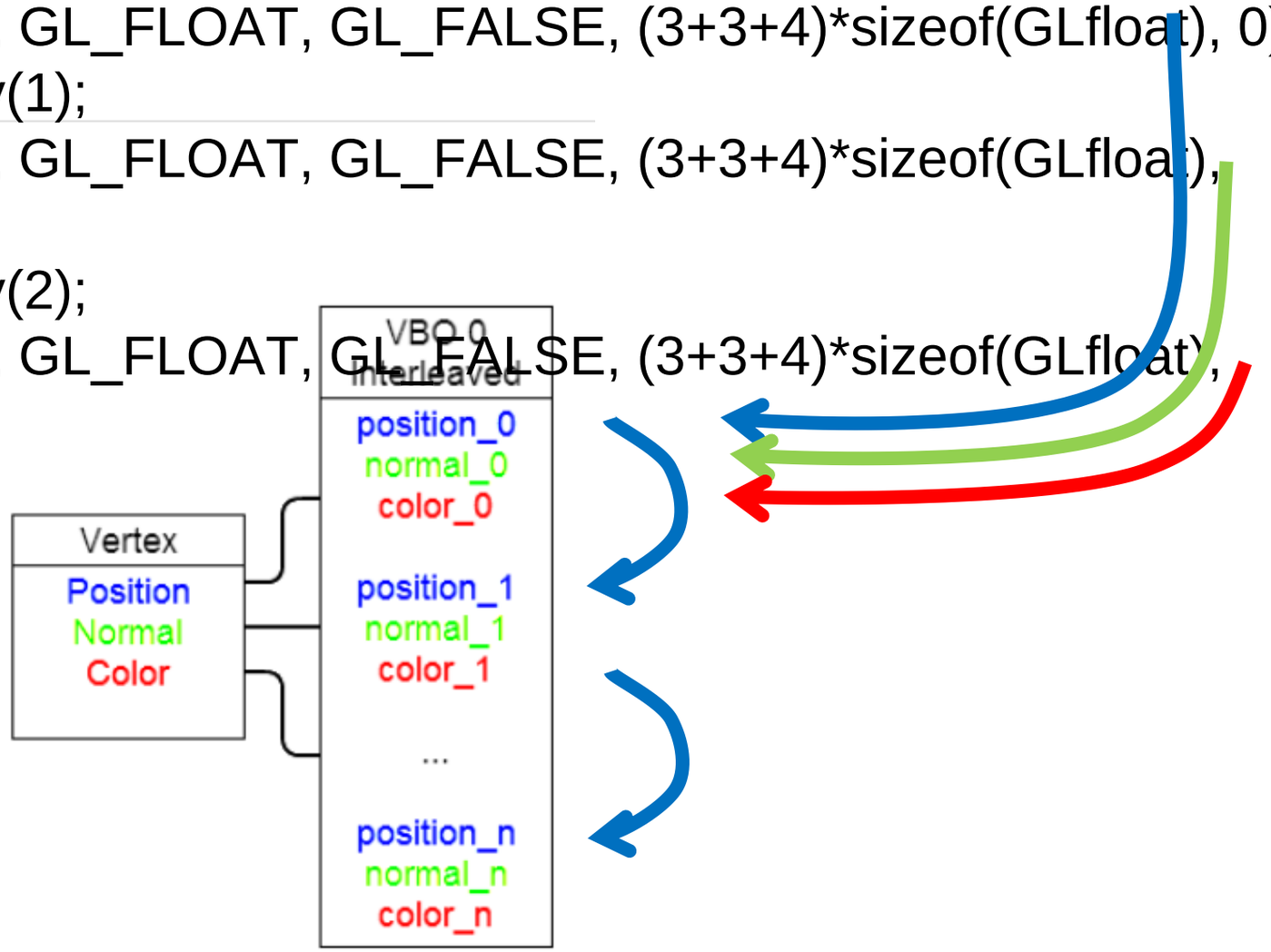
```
glEnableVertexAttribArray(0);
glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 0, 0); // stride=0: tightly
glEnableVertexAttribArray(1);
glVertexAttribPointer(1, 3, GL_FLOAT, GL_FALSE, 0, 0);
glEnableVertexAttribArray(2);
glVertexAttribPointer(2, 4, GL_FLOAT, GL_FALSE, 0, 0);
```



# glVertexAttribPointer?

- 1 Buffer mit allen Attributen

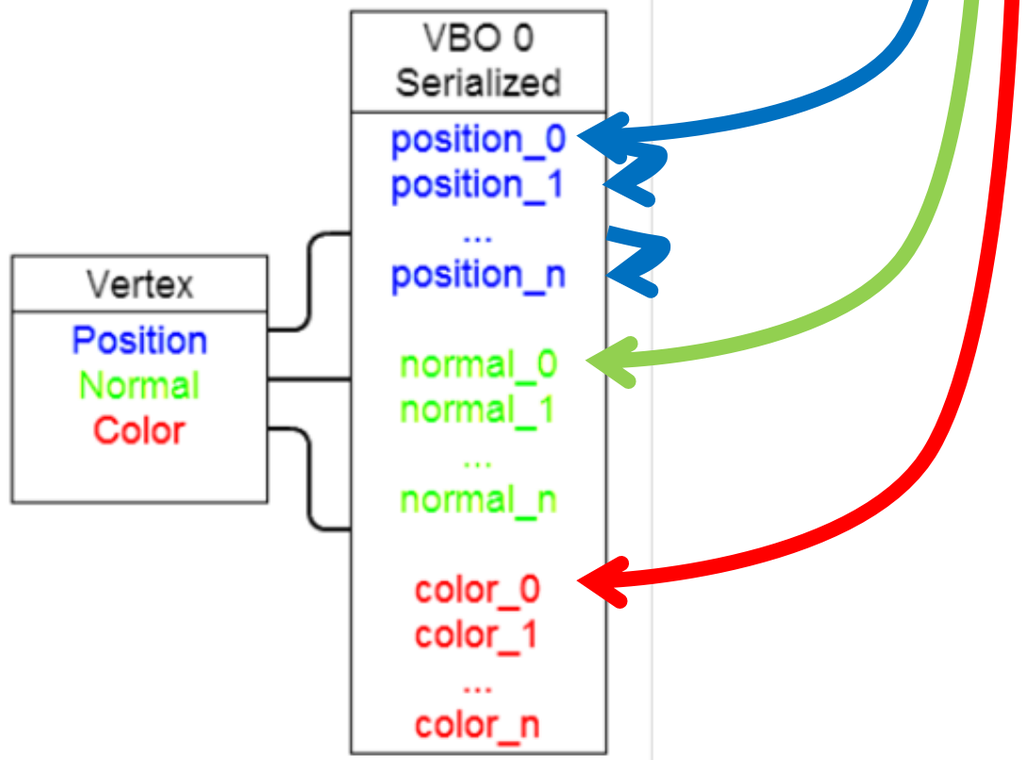
```
glEnableVertexAttribArray(0);
glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, (3+3+4)*sizeof(GLfloat), 0);
glEnableVertexAttribArray(1);
glVertexAttribPointer(1, 3, GL_FLOAT, GL_FALSE, (3+3+4)*sizeof(GLfloat),
3*sizeof(GLfloat));
glEnableVertexAttribArray(2);
glVertexAttribPointer(2, 4, GL_FLOAT, GL_FALSE, (3+3+4)*sizeof(GLfloat),
6*sizeof(GLfloat));
```



# glVertexAttribPointer?

- 1 Buffer mit allen Attributen

```
glEnableVertexAttribArray(0);
glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 0, 0); // stride=0: tightly
glEnableVertexAttribArray(1);
glVertexAttribPointer(1, 3, GL_FLOAT, GL_FALSE, 0, 3(n+1)*sizeof(GLfloat)); // stride=0: tightly
glEnableVertexAttribArray(2);
glVertexAttribPointer(2, 4, GL_FLOAT, GL_FALSE, 0, 2*3(n+1)*sizeof(GLfloat)); // stride=0: tightly
```



# Vertex Array Object (OpenGL 3.3)

- Vertex Array Object aktiviert alle Vertex Attribute

```
// glGenVertexArray (1, &vao);
glBindVertexArray    (vao);
glDrawArrays(GL_TRIANGLES, 0, nVertices);
```

model without VAO

```
glBindBuffer();
glEnableVertexAttribArray();
glVertexAttribPointer();

glBindBuffer();
glEnableVertexAttribArray();
glVertexAttribPointer();

glBindBuffer();
glEnableVertexAttribArray();
glVertexAttribPointer();

glBindBuffer();
glEnableVertexAttribArray();
glVertexAttribPointer();
```

or

model with VAO

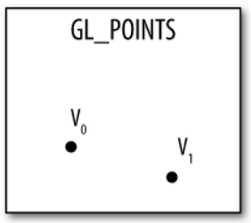
```
glBindVertexArray();
```

That's all
☺

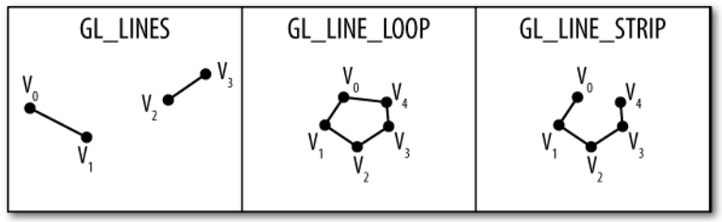
# Vertex Array Object für Viereck?

- Welche Möglichkeiten?
  Wieviele Vertices?
  - GL_TRIANGLES, nicht-indiziert
  - GL_TRIANGLES, indiziert

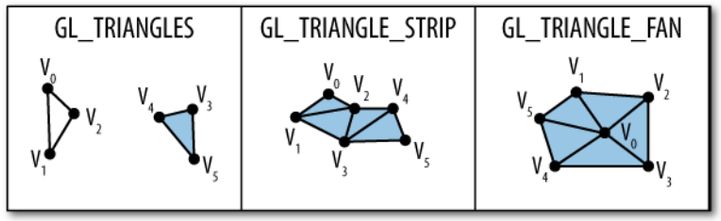
## Primitive: Punkte




## Linien




## Flächen

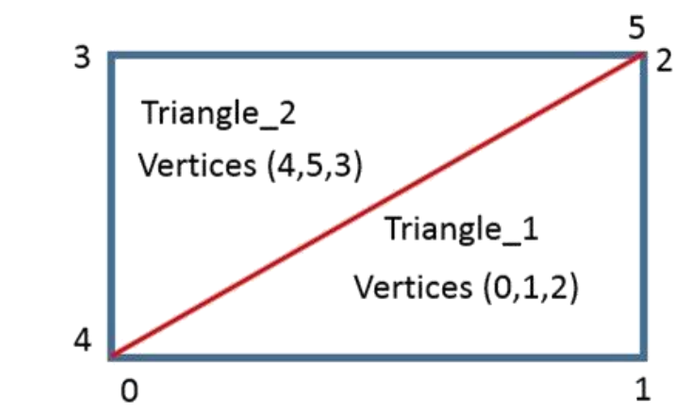


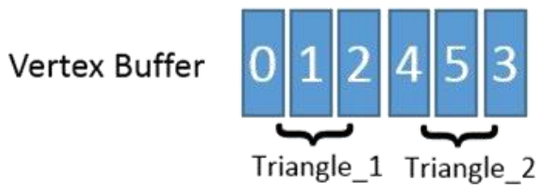
# Vertex Array Object für Viereck?

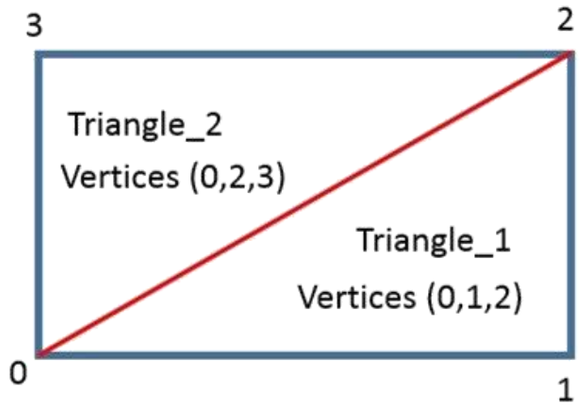
- GL_TRIANGLES, nicht-indiziert

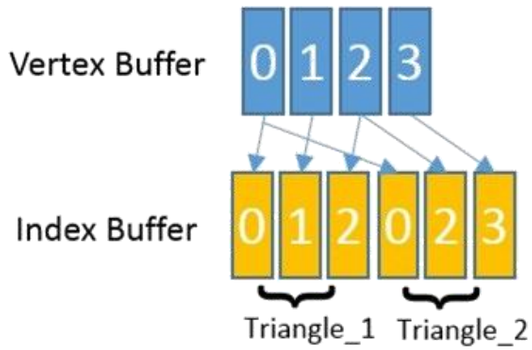
  glBufferData(GL_ARRAY_BUFFER, vertices.size()*sizeof(VertexFormat), vertices.data(), GL_STATIC_DRAW);
  glDrawArrays(GL_TRIANGLES, 0, 3); // beginnend bei 0, nVertices=3 macht 1 Dreieck

- Wieviele Vertices?



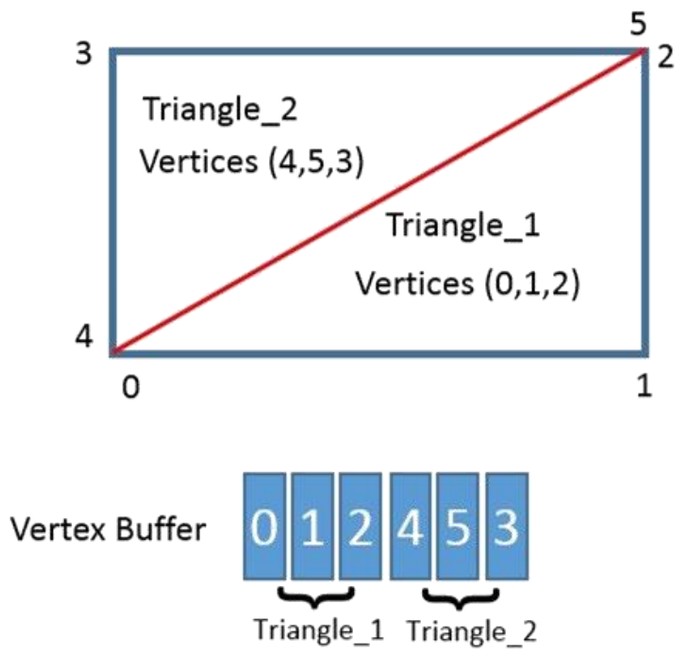



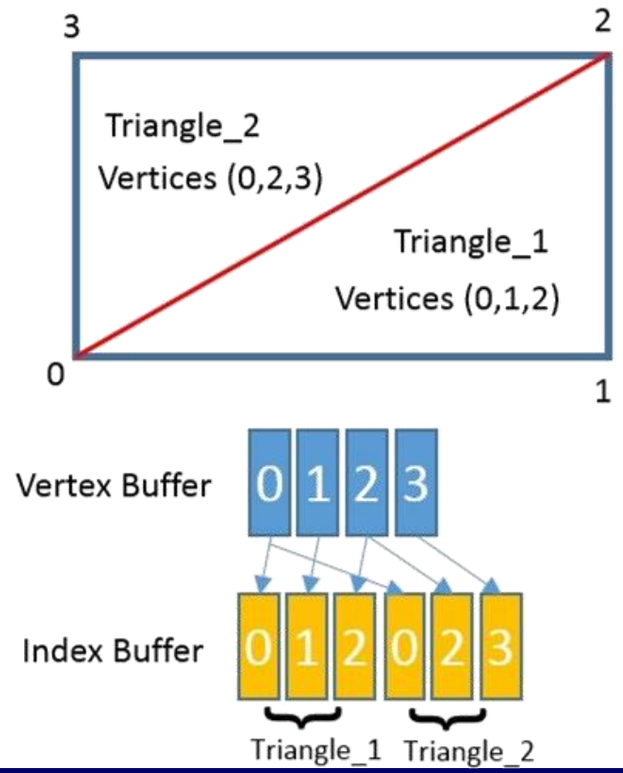
# Vertex Array Object für Viereck?

- GL_TRIANGLES, indiziert

  glBufferData(GL_ELEMENT_BUFFER, vertices.size()*sizeof(GLushort), indices.data(), GL_STATIC_DRAW);
  glDrawElements(GL_TRIANGLES, nVertices, GL_UNSIGNED_SHORT, 0);

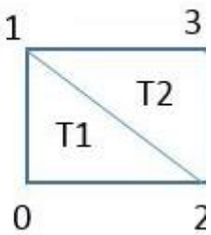
- Wieviele Vertices? Indices?

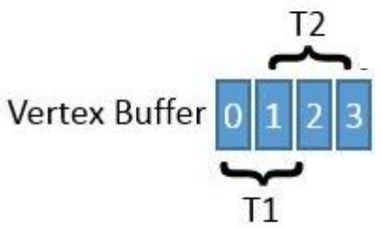


# Vertex Array Object für Viereck?

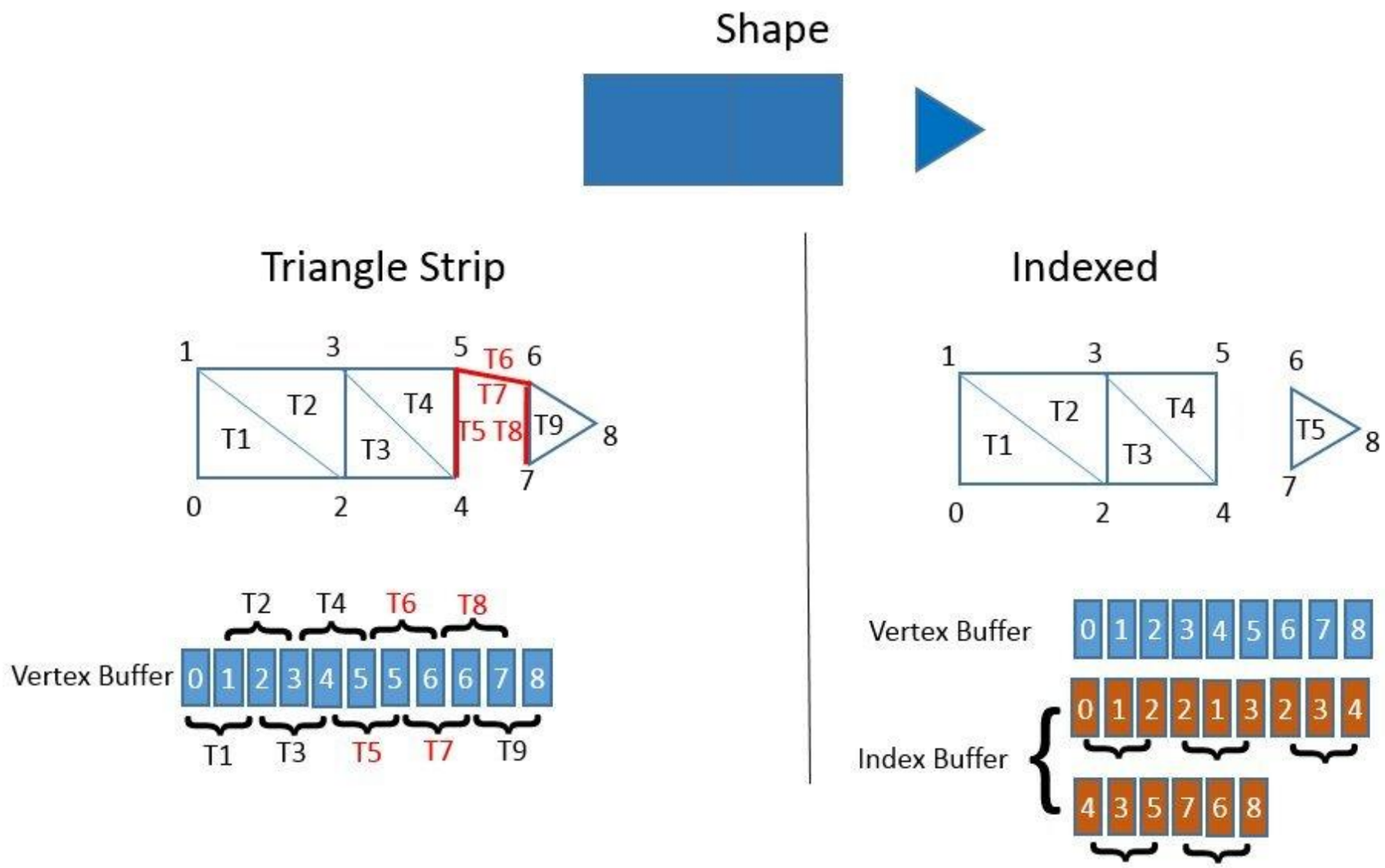
- GL_TRIANGLE_STRIPS, nicht-indiziert
- Wieviele Strips? Vertices?



# Vertex Array Object für Streifen?

- GL_TRIANGLE_STRIPS, nicht-indiziert
- Wieviele Strips? Wieviele Vertices insgesamt?



# Hilfreiche Ressourcen

- McKesson: Learning Modern 3D Graphics Programming (2012)
  https://paroj.github.io/gltut/index.html
- J. de Vries: Learn OpenGL (2014)
  https://learnopengl.com/
  https://learnopengl.com/book/offline%20learnopengl.pdf
- The OpenGL Mathematics Library
  https://glm.g-truc.net
- The OpenGL Extension Wrangler Library
  http://glew.sourceforge.net/
- The freeglut Project
  http://freeglut.sourceforge.net/