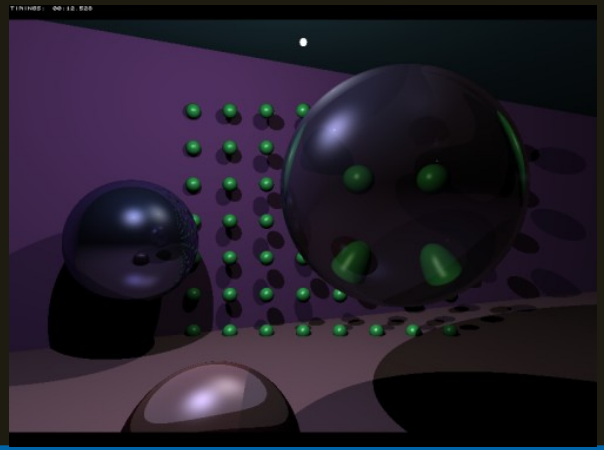
TECHNISCHE UNIVERSITÄT
BERGAKADEMIE FREIBERG
Die Ressourcenuniversität. Seit 1765.

Multimedia: OpenGL I



Ben Lorenz / Jonas Treumer

Themen

- Motivation
  - Computergrafik
- Theorie
  - Grafikpipeline
- OpenGL
  - Überblick
  - Einrichten der Programmierumgebung
  - Hello, Triangle

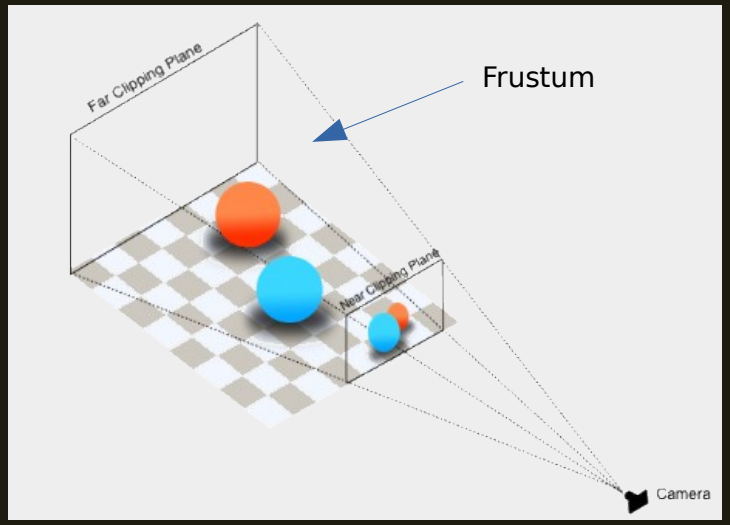
Motivation

- Computergrafik:
  - Bisher: Repräsentation von Bildern (BMP, JPEG) als „Pixel-Matrizen“
  - Problem: keine Dynamik möglich!
  - Beispiel:

Bildquelle
https://ichef.bbci.co.uk/news/624/cpsprodpb/821C/production/_89780333_blizzard_overwatch-game.jpg

Theorie

- Geometrie:
  - Darstellung dreidimensionaler Objekte auf einem zweidimensionalen Bildschirm: Projektion

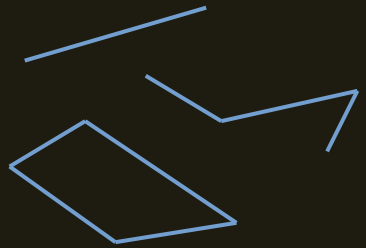


Bildquelle
http://www.falloutsoftware.com/tutorials/gl/perspective-transform-visual-diagram-opengl-3d-to-2d.png

Theorie

- Geometrie:
  - Modellierung einer geometrischen 3D-Welt
  - Aufbau aus Primitiven:

Punkte



Linien

Flächen

→ 2D-Primitiven im 3D-Raum!

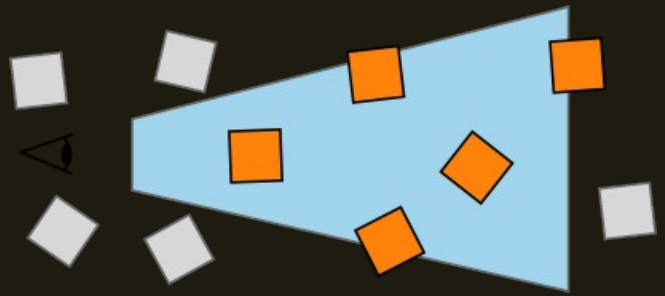
Theorie

- Optimierungen:
  - Ausschließen nicht sichtbarer Elemente: Culling
    - Auf Ebene der Primitiven: z. B. Backface Culling

Bildquelle
http://img06.deviantart.net/7bb1/i/2014/020/8/b/backface_culling_issues_by_theartistictiger-d730q04.jpg

Theorie

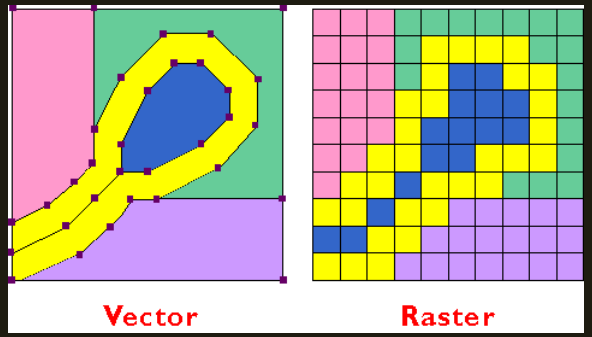
- Optimierungen:
  - Ausschließen nicht sichtbarer Elemente: Culling
    - Auf Ebene der Objekte: z. B. Frustum Culling



Bildquelle
http://www.txutxi.com/wp-content/uploads/2015/07/frustum_culling.png

Theorie

- Diskretisierung der Primitiven:
  - Vektor- zu Rasterdaten: Fragmente bzw. Pixel

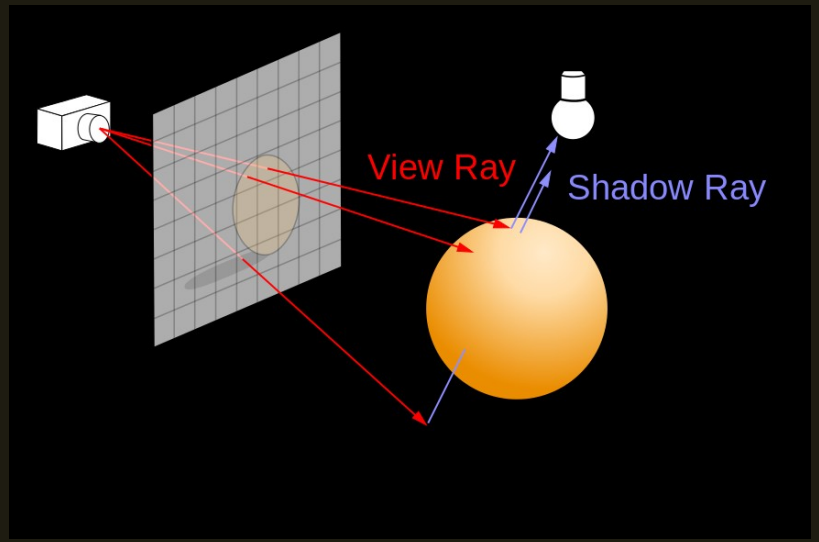


  - Zwei fundamental unterschiedliche Ansätze:
    - Ray Tracing
    - Rasterung

Bildquelle http://libdiz.tk/wp-content/uploads/2015/11/convert-vector-to-raster.gif

Theorie

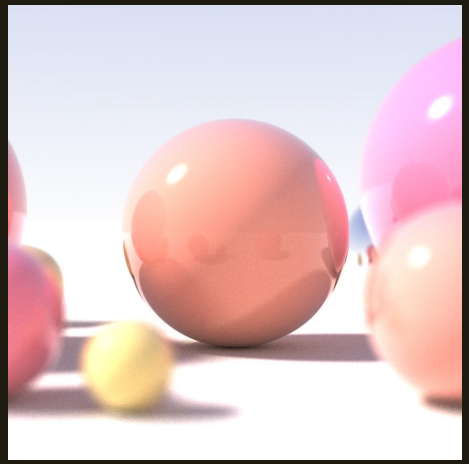
- Ray Tracing:
  - Rückverfolgung der Lichtstrahlen vom Auge zu den Objekten der Szene



Bildquelle
https://upload.wikimedia.org/wikipedia/commons/thumb/8/83/Ray_trace_diagram.svg/875px-Ray_trace_diagram.svg.png

Theorie

- Ray Tracing:
  - Rückverfolgung der Lichtstrahlen vom Auge zu den Objekten der Szene
  - Berücksichtigung von Materialeigenschaften, Reflexion und Refraktion

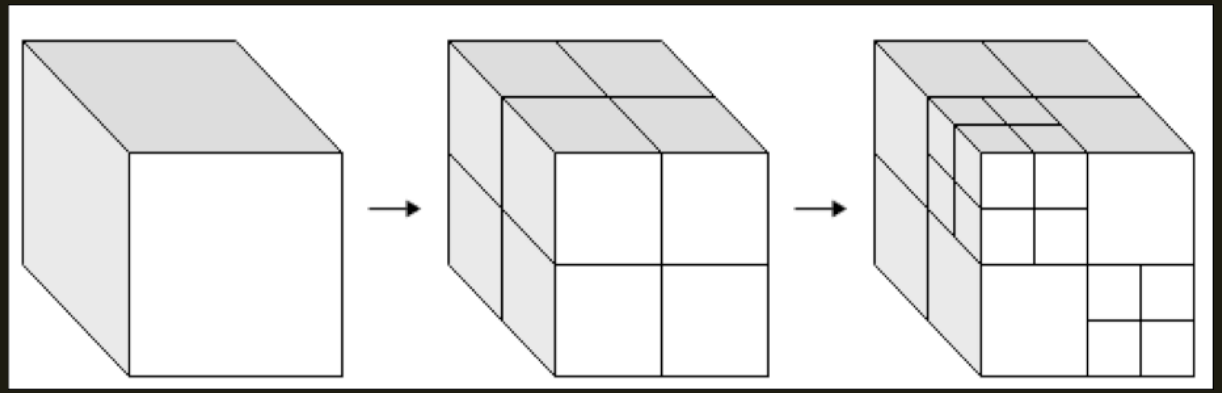


Bildquelle
https://upload.wikimedia.org/wikipedia/commons/3/32/Recursive_raytrace_of_a_sphere.png

Bildquelle
https://upload.wikimedia.org/wikipedia/commons/thumb/e/ec/Glasses_800_edit.png/1280px-Glasses_800_edit.png

Theorie

- Ray Tracing:
  - Problem: Im Prinzip Betrachtung aller Primitiven pro Strahl notwendig
  - Optimierung: Räumliche Datenstrukturen (Kd-Tree, Octree, ...)



Bildquelle https://assistly-production.s3.amazonaws.com/188862/portal_attachments/504528/Octree_depth_original.png?AWSAccessKeyId=AKIAJNSFWOZ6ZS23BMKQ&Expires=1496370905&Signature=gmvXpHNdc%2B87nI21V2j87oOj21Y%3D&response-content-disposition=filename%3D%22Octree_depth.png%22&response-content-type=image%2Fpng

Theorie

- Rasterung:
  - Vertices (= Stützpunkte) über Koordinatentransformationen aus dem Frustum in eine quaderförmige Box projizieren: Zentralprojektion → Fluchtpunkt
  - X und Y als Bildschirmkoordinaten, Z für den Depth-Buffer

Bildquelle https://assistly-https://hostr.co/file/ixF4pHfrMruR/LogDepthNew.png

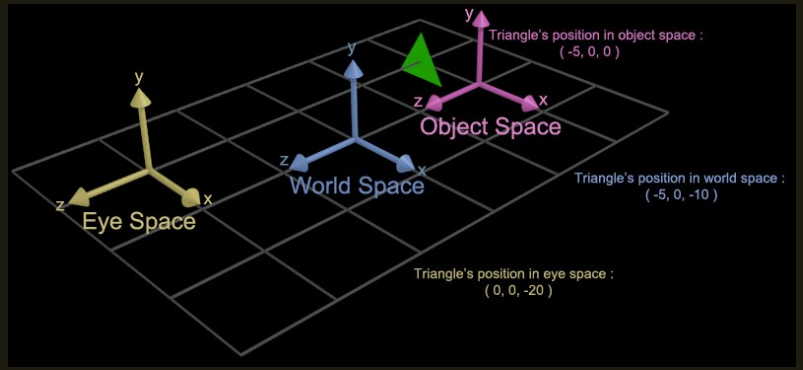
Theorie

- Rasterung:
  - Vertices (= Stützpunkte) über Koordinatentransformationen aus dem Frustum in eine quaderförmige Box projizieren: Zentralprojektion → Fluchtpunkt
  - X und Y als Bildschirmkoordinaten, Z für den Depth-Buffer
  - Baryzentrische Interpolation mit Shadern
  - Problem: Beleuchtung!
  - Transformationen liefern nur die Abbildung der Koordinaten, keine Lichtverhältnisse.

Bildquelle https://assistly-
https://hostr.co/file/ixF4pHfr
MruR/LogDepthNew.png

Theorie

- Koordinatentransformationen:
  - Primitiven in der Welt platzieren
  - Welt aus Sicht der Kamera betrachten
  - Kamerabild auf den Bildschirm projizieren



Bildquelle
http://theamazingking.com/images/opengl/matrix1.JPG

Theorie

- Koordinatentransformationen:
  - Realisierung als affine Transformationen:

$$f(\vec{x}) = A \cdot \vec{x} + \vec{t}, \quad \vec{x} \in \mathbb{R}^3$$

  - Kollinearität bleibt erhalten.
  - Parallelität bleibt erhalten.
  - Teilverhältnisse kollinearer Punkte bleiben erhalten.

Theorie

- Koordinatentransformationen:
  - Einfachere Handhabung über homogene Koordinaten:
    - Ergänze jeden Vektor um eine konstante vierte Komponente:

$$\vec{x} = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ 1 \end{pmatrix}$$

Theorie

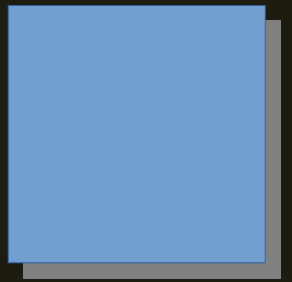
- Koordinatentransformationen:
  - Einfachere Handhabung über homogene Koordinaten:
    - Erweitere die Transformationsmatrix auf 4x4:

$$A = \begin{pmatrix} A_{00} & A_{01} & A_{02} & t_0 \\ A_{10} & A_{11} & A_{12} & t_1 \\ A_{20} & A_{21} & A_{22} & t_2 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \vec{x} = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ 1 \end{pmatrix}$$

Theorie

- Koordinatentransformationen:
  - Wichtige Transformationen:

Einheitsabbildung: $A = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

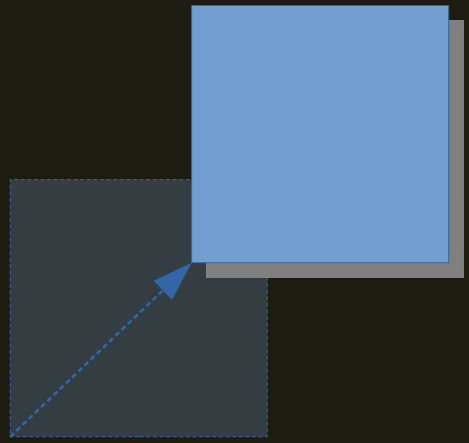


Theorie

- Koordinatentransformationen:
  - Wichtige Transformationen:

Translation:

$$A = \begin{pmatrix} 1 & 0 & 0 & t_0 \\ 0 & 1 & 0 & t_1 \\ 0 & 0 & 1 & t_2 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

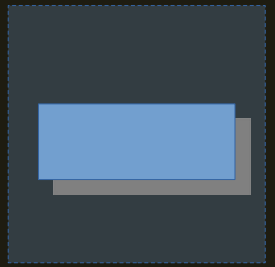


Theorie

- Koordinatentransformationen:
  - Wichtige Transformationen:

Skalierung:

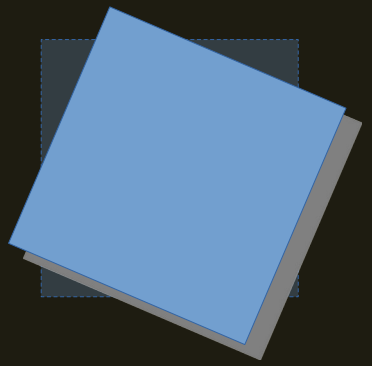
$$A = \begin{pmatrix} s_0 & 0 & 0 & 0 \\ 0 & s_1 & 0 & 0 \\ 0 & 0 & s_2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



Theorie

- Koordinatentransformationen:
  - Wichtige Transformationen:

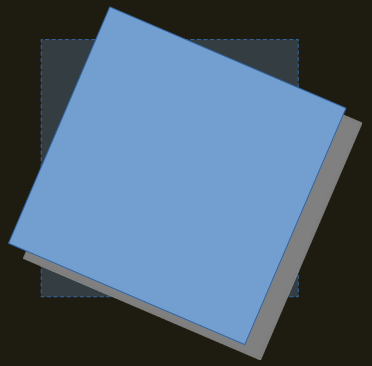
Rotation um x: $$A = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(v) & -\sin(v) & 0 \\ 0 & \sin(v) & \cos(v) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



Theorie

- Koordinatentransformationen:
  - Wichtige Transformationen:

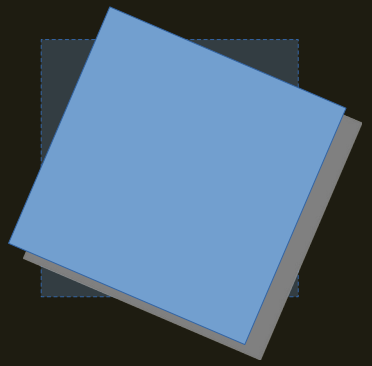
Rotation um y: $$A = \begin{pmatrix} \cos(v) & 0 & \sin(v) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(v) & 0 & \cos(v) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



Theorie

- Koordinatentransformationen:
  - Wichtige Transformationen:

Rotation um z: $A = \begin{pmatrix} \cos(v) & -\sin(v) & 0 & 0 \\ \sin(v) & \cos(v) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$



OpenGL

- Überblick:
  - Open Graphics Library: 1991 von Silicon Graphics entwickelt
  - Unabhängig von der Programmiersprache
  - Implementierungen für zahlreiche Systeme (Windows, Linux, macOS X)
  - Realisierung als State Machine
  - Ab OpenGL 3.0: Programmable Pipeline per Shader-Sourcecode
  - Aktuell: Version 4.5

OpenGL

- Überblick:
  - OpenGL ES: Vereinfachte Version für Embedded Systems
  - Version 3.0 entspricht ungefähr Version 3.3 des OpenGL-API
  - Einige Unterschiede:
    - Keine Tessellation(-Shader)
    - Keine doppelte Gleitkommapräzision
    - Keine Geometry-Shader
    - Programmable Pipeline muss verwendet werden.
    - ...

OpenGL

- Programmierumgebung:
  - Eine OpenGL-Implementierung muss unterschiedliche API-Versionen (3.0, 4.1, …) unterstützen.
  - Problem: Gleichnamige Funktionen verhalten sich abhängig von der API-Version teilweise unterschiedlich.
  - Lösung: Die Betriebssysteme (Windows, Linux, macOS) stellen in ihren Header-Dateien nur eine alte Version der API zur Verfügung (z. B. OpenGL 1.1 unter Windows), deren Funktionsumfang stark eingeschränkt ist. Neuere Funktionen sind zwar im entsprechenden Grafiktreiber verfügbar, müssen aber zur Laufzeit „geladen“ werden:

OpenGL

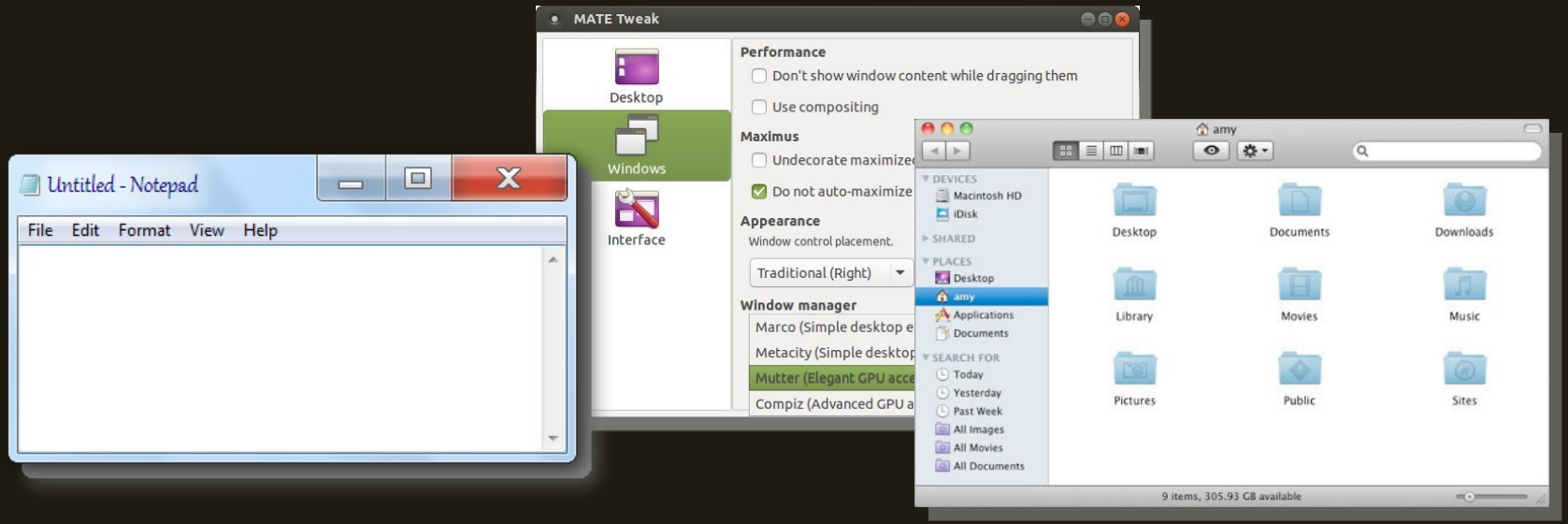
```
//In a header somewhere.
#include <glext.h>
PFNGLUSEPROGRAMPROC glUseProgram;

//In an initialization routine
glUseProgram = (PFNGLUSEPROGRAMPROC)wglGetProcAddress("glUseProgram");
```

- Automatisierung durch Loader wie glew oder glad
- Zusätzlich: Laden von plattform- und hardwareabhängigen Extensions

OpenGL

- Programmierumgebung:
  - Plattformunabhängigkeit? Erst nach der Fenster- bzw. Kontexterzeugung ...

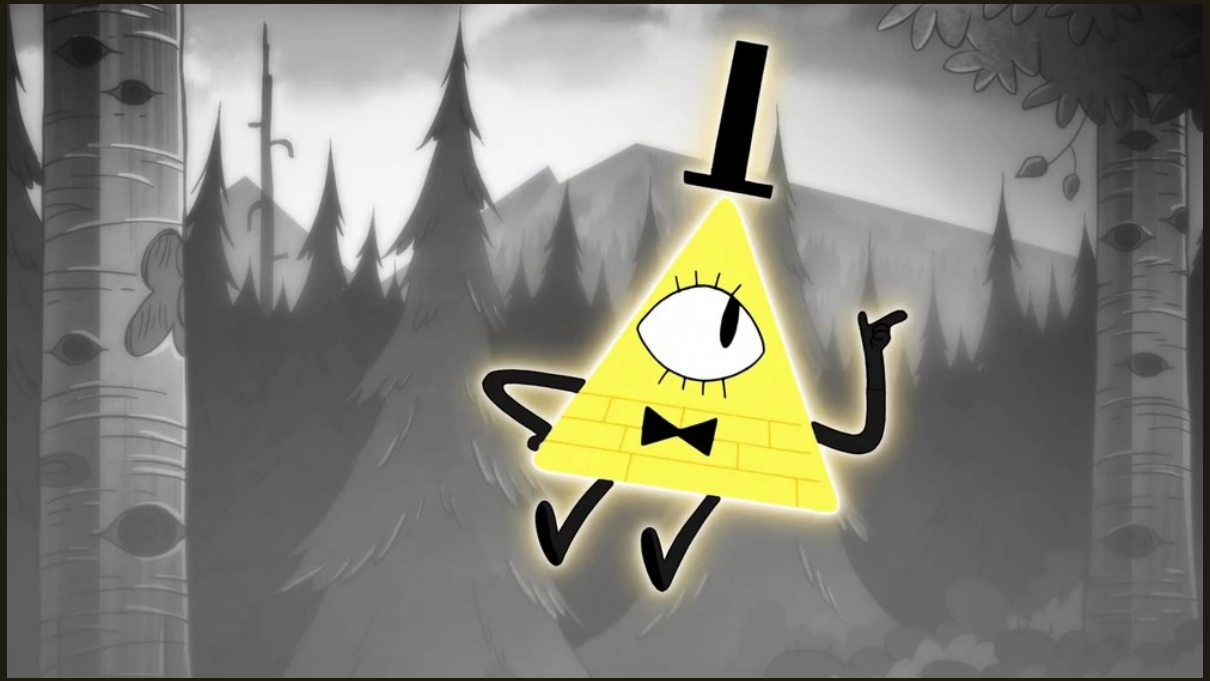


OpenGL

- Programmierumgebung:
  - Plattformunabhängigkeit? Erst nach der Fenster- bzw. Kontexterzeugung
  - Lösung: Hilfsbibliothek verwenden, z. B. (free)glut, sdl, glfw, ...

OpenGL ES

- Hello, Triangle



Bildquelle
https://vignette2.wikia.nocookie.net/gravityfalls/images/b/b9/S1E19_Bill_schnipst.png/revision/latest/scale-to-width-down/1000?cb=20150721161440&path-prefix=de