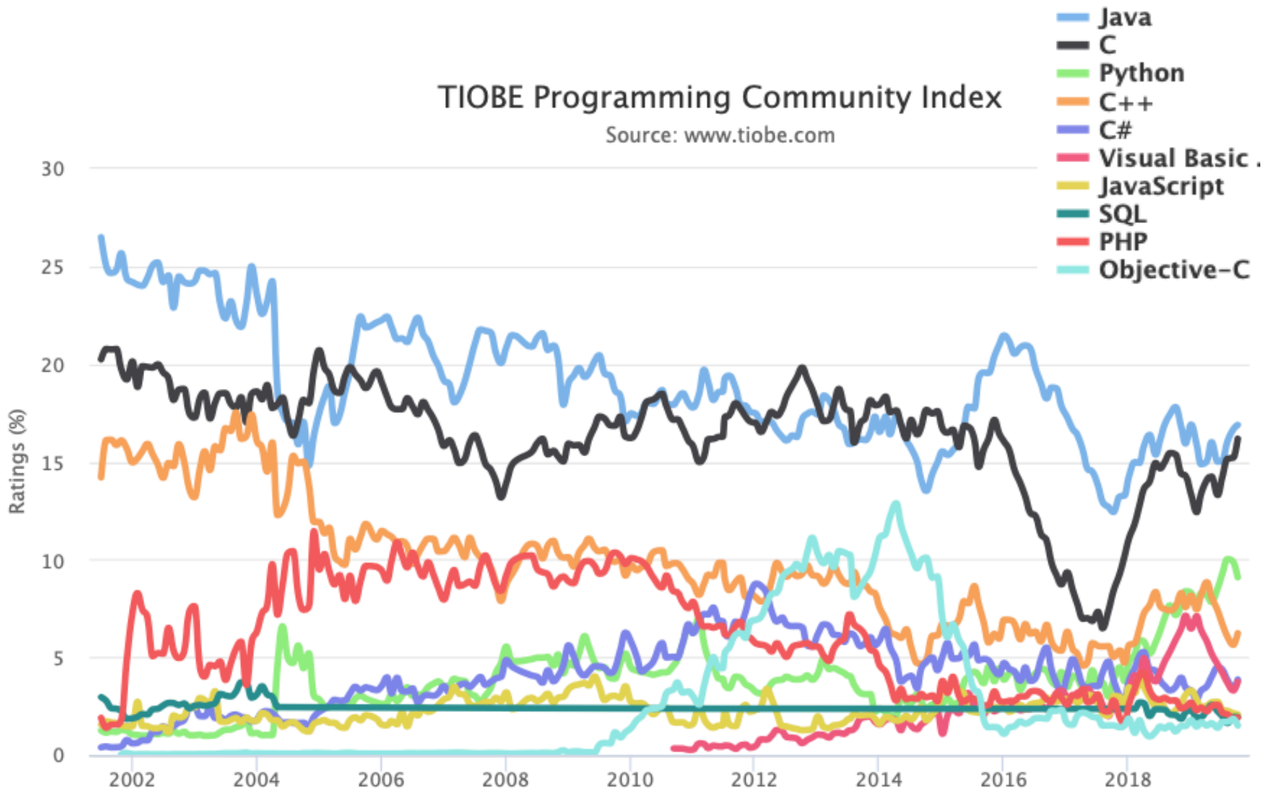
Computergraphik I
Organisatorisches

G. Zachmann
University of Bremen, Germany
cgvr.cs.uni-bremen.de

Voraussetzungen

- Ein wenig Mathematik
  - Trigonometrie
  - Lineare Algebra: Rechnen mit Vektoren und Matrizen
- Für die Programmieraufgaben:
  - Ein wenig Programmierkenntnisse in C/C++ (gute Gelegenheit, dieses wieder aufzufrischen)



Webseite zur Computergraphik

- Alle wichtigen Informationen zur VL stellen wir Ihnen auch im Internet zur Verfügung :

  http://cgvr.cs.uni-bremen.de/

  → "Teaching" → "Computergraphik"
- Folien & Übungsblätter
- Literaturhinweise, Online-Doku
- Evtl. aktuelle Meldungen
- Bitte anmelden in StudIP!

Modus der Vorlesung

- Vorlesung: Dienstag 10 ct — 14 Uhr
  - Modus ?
- Übungs-/Tutoriums-Slots:
  - Dienstag 16 ct – 18 Uhr
  - Mittwoch 8 ct – 10 Uhr
  - Donnerstag 10 ct – 12 Uhr
- Abgabe der Aufgaben: Montag 23:59
  - Vermutlich über git (bitte mit Tutor abklären)
- Tutoren: Philipp Dittmann, Max Kaluschke, Roland Fischer

Ziel der Vorlesung

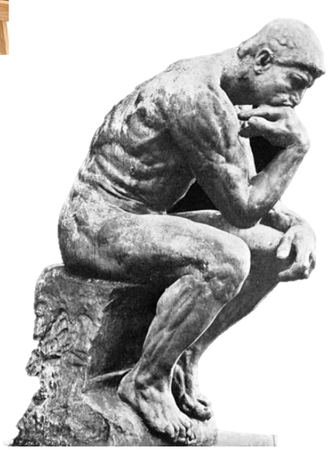
The mind is not a vessel to be filled, but a fire to be kindled.

(Plutarch)

Ziele der Vorlesung

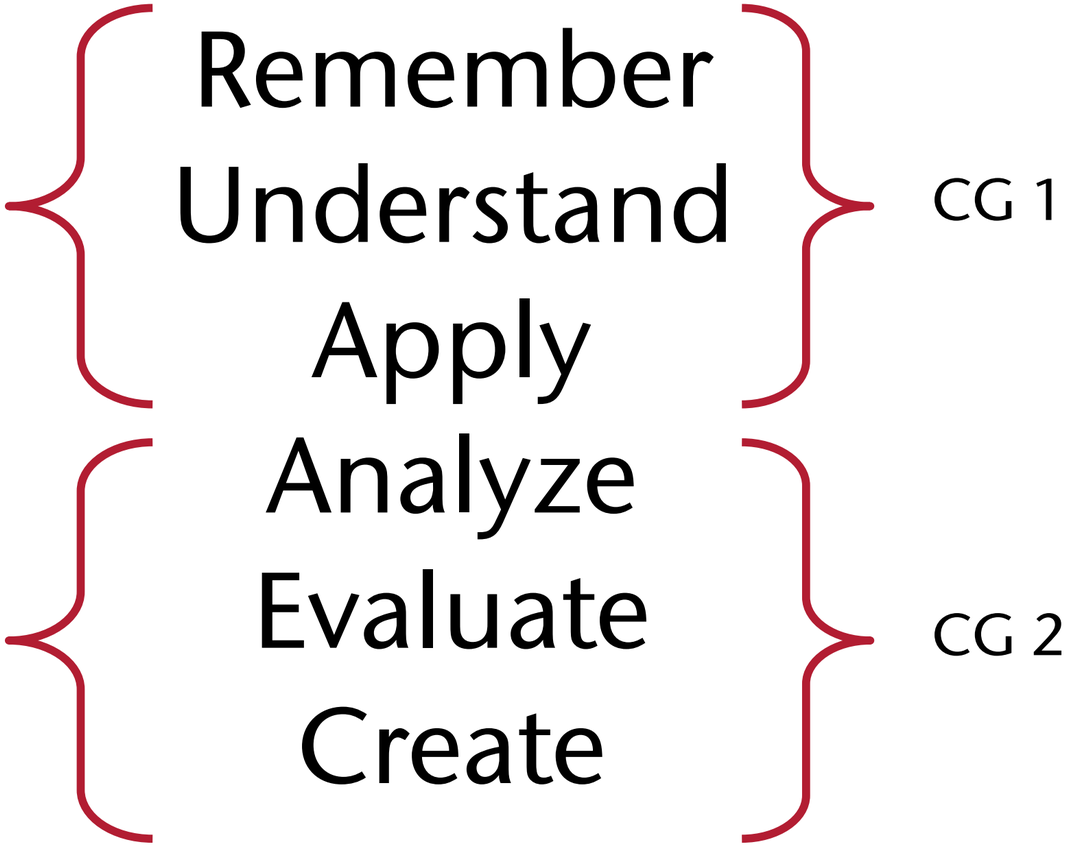
- Praxis: Sei in der Lage, einfache interaktive 3D Graphik-programme zu schreiben (in OpenGL)

- Theorie: Verstehe den mathematischen Hintergrund und einige grundlegende Verfahren moderner 3D Graphiksysteme



- Diese Vorlesung behandelt *nicht* Graphikprogramme/Modeler wie Blender, 3DStudio Max, Cinema4D, AfterEffects, Photoshop, ...

Cognitive Processes



Die Übungen

- Hat jemand keinen eigenen Laptop?
- Übungsblätter gibt es jede Woche, jeweils am Montag vormittag
- Teils theoretisch, teils praktisch
- Praktische Aufgaben = Programmieren in C++ und OpenGL
  - Eigentlich nur "C mit Klassen"
  - Bedarf für C/C++ Refresher?
- In 4er-Gruppen ! (zur Not auch 3-er oder 5-er)

"Vorstellung ersetzt erst dann das Handeln,
wenn jene von diesem
ausreichend Erkenntnisse gewonnen hat."

[Piaget]

Persönliche Vorab-Installationen

- Bitte folgende Software schon vorab auf dem eigenen Laptop installieren, falls nicht schon vorhanden

1. Qt 5.x :
   - Webseite: https://www.qt.io/download-qt-installer
   - Mac: dead-simple
   - Linux: 32- oder 64-Bit Version
   - Windows
     - Achtung: OpenGL-Version nehmen!
     - Beispiel: Qt 5.1.1 for Windows 64-bit (VS 2012, OpenGL, 522 MB)
   - Achtung: man darf QtCreator (eine IDE) verwenden, wir können aber keinen Support dafür geben!

2. Programmierumgebung:

- Es reicht im Prinzip: einfacher ASCII-Editor + Compiler
- Mac: XCode
  - Unter https://developer.apple.com/
    bzw. im Mac App Store
- Linux: dort sollte eigtl schon alles bereit stehen
  - Falls nicht: Kdevelop, oder geeigneter ASCII-Editor
- Windows: Visual Studio 2017
  - Verfügbar über https://visualstudio.microsoft.com/thank-you-downloading-visual-studio/?sku=Community

Die Prüfung

- Gesamtnote:
  - Notespiegel: 95% $\longrightarrow$ 1.0 , 40% $\longrightarrow$ 4.0
  - Wer die Übungsaufgaben gemacht hat:

$$\text{Gesamtnote} = \min\left\{\frac{1}{2}\cdot\ddot{\text{U}}\text{bungsaufgaben} + \frac{1}{2}\cdot\text{Klausur}\,,\ \text{Klausur}\right\}$$

  - Voraussetzung: beide Noten ≥ 4.0

    (Allgemeiner Teil der Bachelorprüfungsordnungen der Universität Bremen, 2010)

  - Wer die Übungsaufgaben nicht gemacht hat:

$$\text{Gesamtnote} = \text{Klausurnote}$$

Prüfungsmodalitäten

- Klausur:
  - Kein Taschenrechner, kein Handy, keine Smartwatch, etc.
  - Erlaubt ist "persönlicher Spickzettel": 3 DIN-A4-Blätter mit beliebigen Notizen, persönlich in eigener(!) Handschrift geschrieben

- Bewertungskriterien der Programmier-Aufgaben:
  1. Gute Variablen- und Funktionsnamen
  2. Genügend in-line Kommentare
  3. Dokumentation der Funktion und deren Parameter (in/out, pre-/post-condition, was tut die Funktion, ...)
  4. Funktionalität (Aufgabe vollständig gelöst? Bug-frei? ...)

Documentation: minimum

```
// Convert HSV color space to RGB
// input:  h must be in [0,360); s,v must be in [0,1]
// output: r,g,b (out) will be in [0,1]
// Warning: no parameter range checks are done!
__device__
void HSVtoRGB( float *r, float *g, float *b,
               float h, float s, float v )
{
```

Documentation: better

```
/**  Compute point nearest to q and on an edge of SIG
 *
 * @param q             the current query point
 * @param points        the point cloud
 * @param delaunay      delaunay diagram of the point cloud
 * @param pstar         NN of q (out)
 * @param pstar2        neighbor of pstar (in SIG) (out)
 * @param phat          point closest to q on edge (pstar,pstar2) (out)
 * @param d             distance between q and phat (out)
 *
 * @warning
 *   Assumes that a SIG has been computed!!
 * @bug
 *   Bis jetzt ist pstar2 nur NN zu pstar im Delaunay-Graph, nicht im SIG!!
 **/

void nearest_on_graph( const FPoint & q, const std::vector<FPoint> & points,
                       const Proximity & proximity,
                       unsigned int * const pstar, unsigned int * const pstar2,
                       FPoint * const phat, float * const d )
{
```

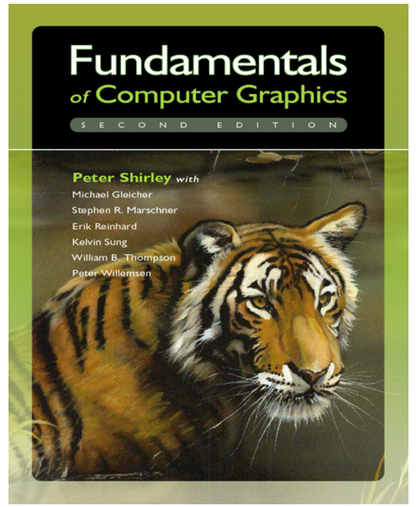
Plagiarismus

- Ernsthafter Fall von akademischem Fehlverhalten!

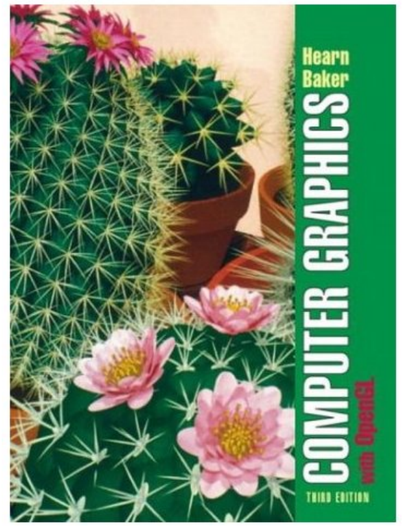
1. Versuch: "gelbe Karte"
   - 0 Punkte für **beide** Gruppen auf das gesamte Übungsblatt
   - Plagiats-Eintrag in der Akte im Prüfungsamt
2. Versuch: "rote Karte"
   - Übungsvornote = 5.0
   - Zweite Meldung ans Prüfungsamt

Literatur

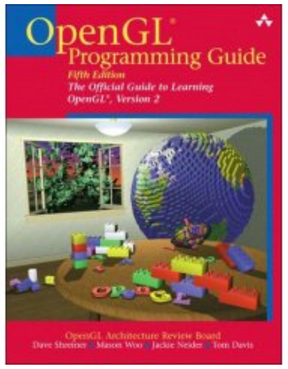
- Peter Shirley: *Fundamentals of Computer Graphics*. Francis & Taylor



- Donald Hearn, M. Pauline Baker: *Computer Graphics with OpenGL*. Prentice Hall

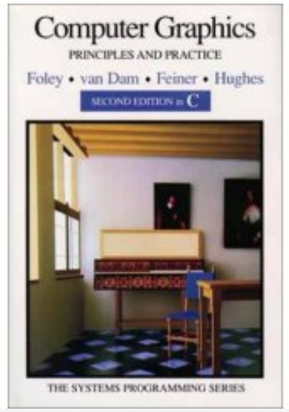


Literatur, nicht mehr ganz aktuell, aber teilweise brauchbar



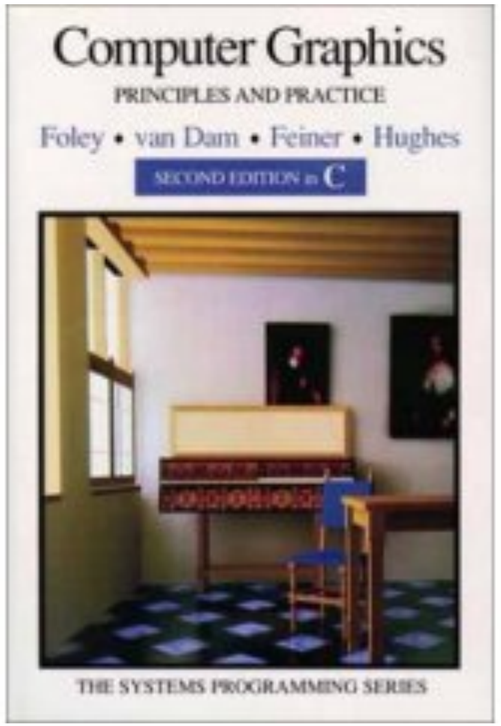
- Dave Shreiner: The *OpenGL Programming Guide: The Official Guide to Learning OpenG.* Addison-Wesley
  - Auf der VL-Homepage als PDF !
  - Passwort ...

- J. L. Encarnação, W. Strasser, R. Klein: *Graphische Datenverarbeitung 1 und 2.* Oldenbourg, 1996



- J. Foley, A. van Dam, S. Feiner, J. Hughes: *Computer Graphics: Principles and Practice*. Addison-Wesley Professional; 2nd Edition, 1995

Exkurs: Jan Vermeer



Jan (Johannes) Vermeer
The Music Lesson
Ca. 1662-1665
Oil on Canvas
75x64 cm

For the Creatively Inclined

Deadline:
Wednesday
November 13th,
2019 - Midnight PST

"In this “Woodville” Art Challenge, you’ll get to work with a fantastic scene from Pixar's Undergraduate Program and SpeedTree, providing you with production-quality assets to showcase your shading, lighting, rendering, and compositing skills for the possibility of winning amazing prizes"