Proseminar: Virtuelle Präsenz

Vortrag:

Steve Rechtenbach

- Einführung
- Was ist OpenGL? / Entwicklung
- Fakten
- Kleines(!) Beispiel
  - Hello OpenGL
  - Shader
- Zusammenfassung
  - OpenGL vs Direct3D

- Kurz nach Einführung von OpenGL unterstützen fast alle kommerziellen Spiele OpenGL
  - Je nach Implementierung/Hardware Geschwindigkeitsvorteil
- Quake => 1996 neben VESA, D3D auch OpenGL

OpenGL

nur für

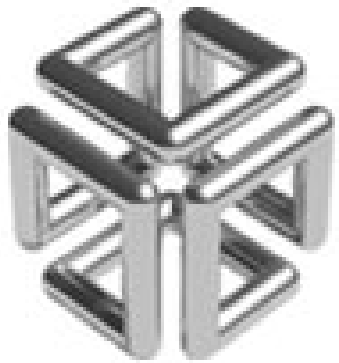
Computerspiele???

- (CAD) Computer Aided Design
- (CAE) Computer Aided Engineering
- (CAM) Computer Aided Manufacturing
- Virtual Reality
- Film und Fernsehen
- Simulation
- Medizin

- API zur 2D/3D Programmierung
- Renderer
  - Zeichnet im wesentlichen nur Dreiecke
    - Infos über Farb-/Texturinformationen
    - Tiefeninformationen der Dreiecke(Vertices)
- State Machine

- **Früher**: Hardwarespezifische 2D/3D Bibliotheken der verschiedenen Hersteller
  - Problem?
- 1992: SGI(Silicon Graphics Inc.) entwickelt OpenGL
  - Erste standardisierte API für 2D/3D Programmierung
- OpenGL basierte stark auf der früheren Arbeit SGI‘s an ihrer IRIX-GL™ Bibliothek

Silicon Graphics



OpenGL ist ...

- Netzwerkfähig!
- Industriestandard
  - große Software und Hardwarehersteller haben Mitspracherecht
- Gratis
  - (DirectX ist jedoch auch kostenlos)
  - Finanziert durch Abgaben der Hardwarehersteller(wenn sie Treiber mit ihren Produkten liefern wollen, + Lizensierungskosten fürs Logo)
- Unterstützung durch viele Programmiersprachen:

Es existieren Bindungen für:

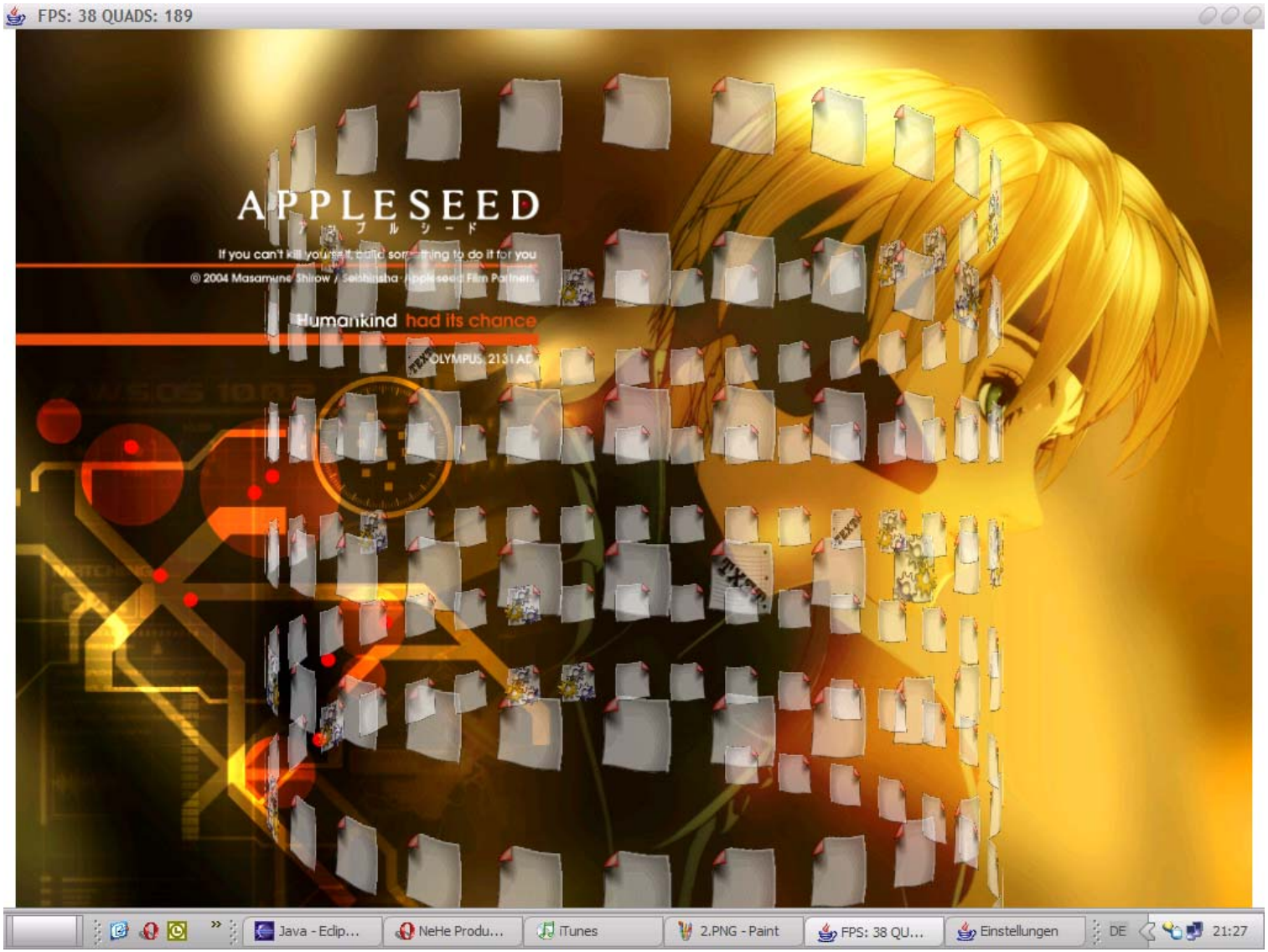
- C = C++
- Java
- Python
- Tcl/Tk
- Pascal
- Delphi
- Perl
- Scheme
- Guile
- Haskell
- Ada
- Fortran
- Modula-3

Unterstützte Betriebssysteme:

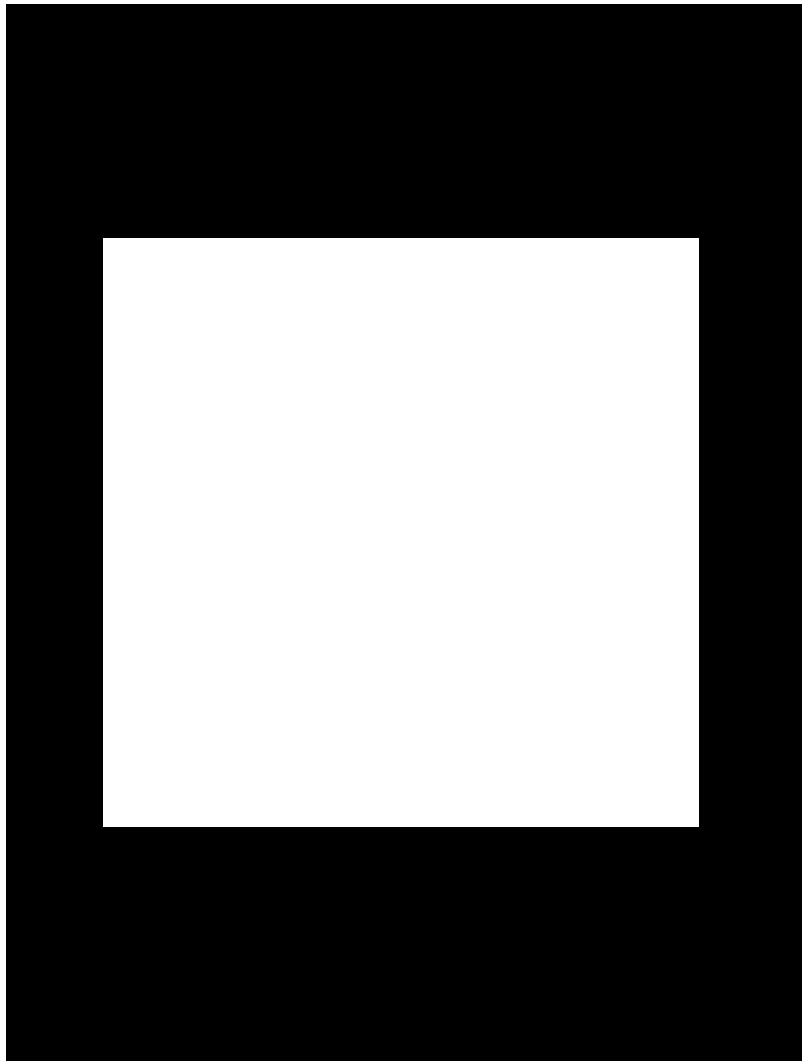
- AIX
- IRIX
- FreeBSD
- NetBSD
- OpenBSD
- Linux
- Windows 9x
- WindowsNT
- OS/2
- Amiga OS
- BeOS
- MacOS
- Und andere UNIX BS...

- Das OpenGL Standarisierungs-Komitee verwaltet die API
  - Treffen sich alle 3 Monate um über neue „features“ und „extensions“ zu entscheiden
- **Promoter-level ➔➔➔➔➔➔**
  - können direkt über neue Standards abstimmen
    - Direktes mitspracherecht
- Contributor-level
  - Evans and Sutherland, Imagination Technologies, Matrox, Quantum3D, S3 Graphics, Spinor GmbH, Tungsten Graphics, and Xi Graphics

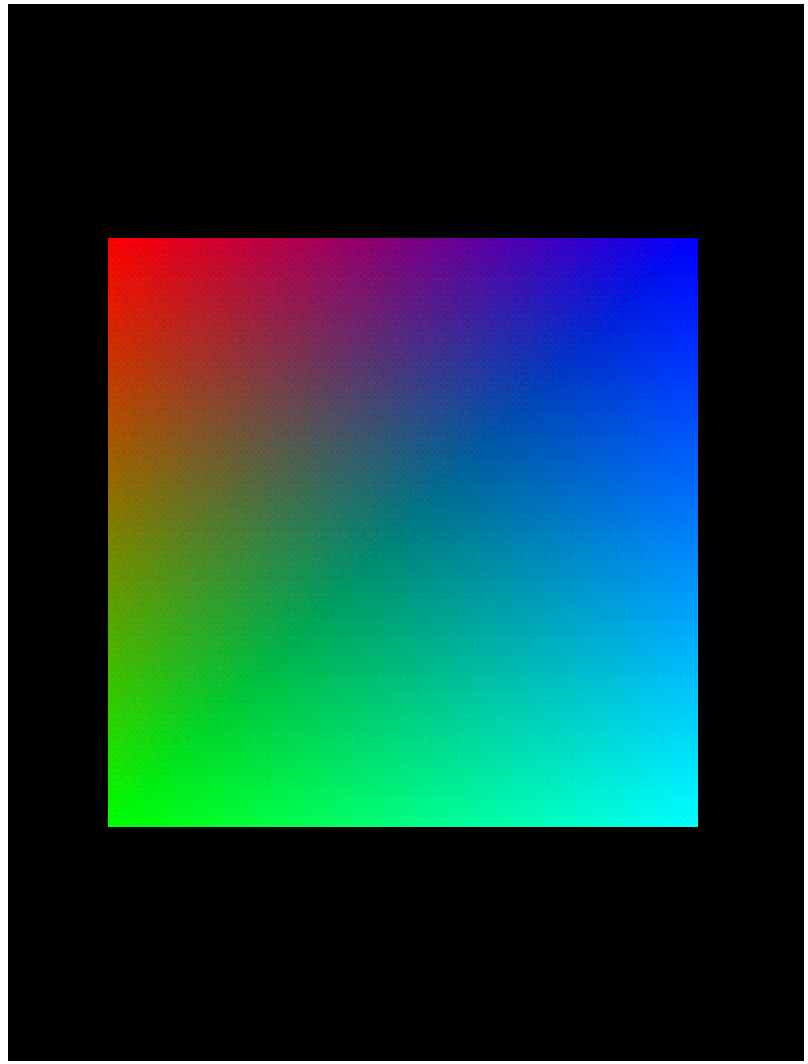
Beispiel


FPS: 38 QUADS: 189
APPLESEED
Humankind had its chance
OLYMPUS, 2131 AD
Java - Eclip...
NeHe Produ...
iTunes
2.PNG - Paint
FPS: 38 QU...
Einstellungen
DE
21:27

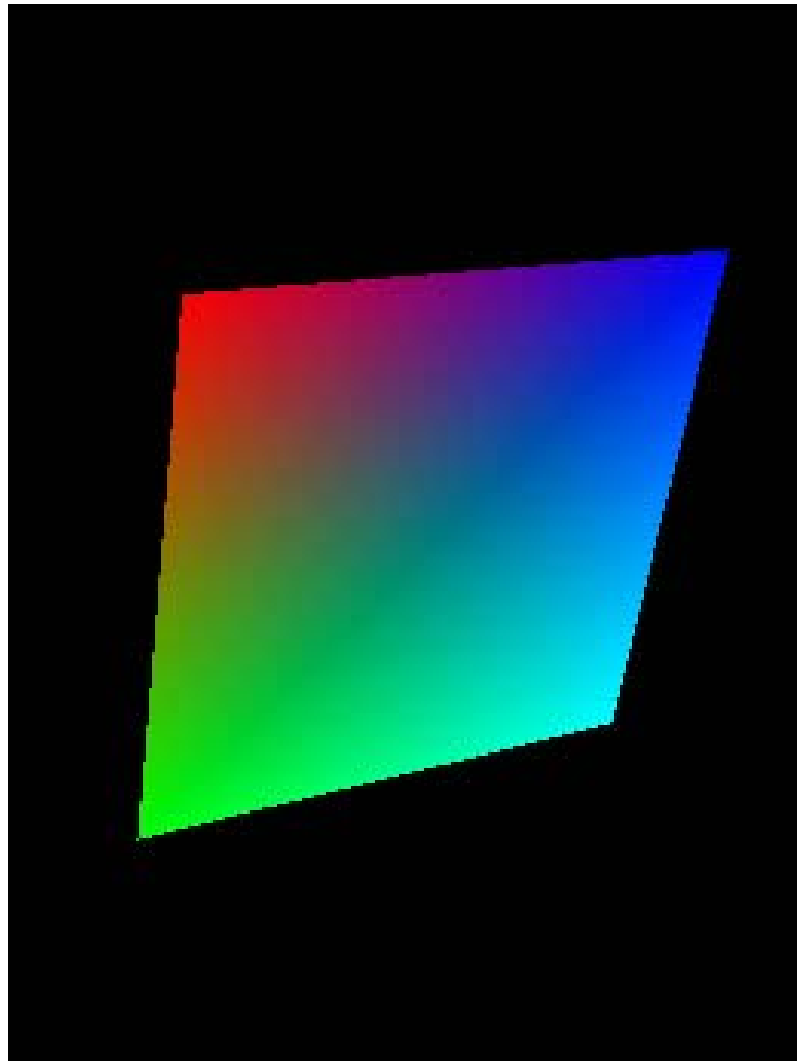
```
glTranslatef(0.0f,0.0f,-6.0f);

glBegin(GL_QUADS);
    glVertex2f(-1.0f,-1.0f);
    glVertex2f( 1.0f,-1.0f);
    glVertex2f( 1.0f, 1.0f);
    glVertex2f(-1.0f, 1.0f);
glEnd();
```



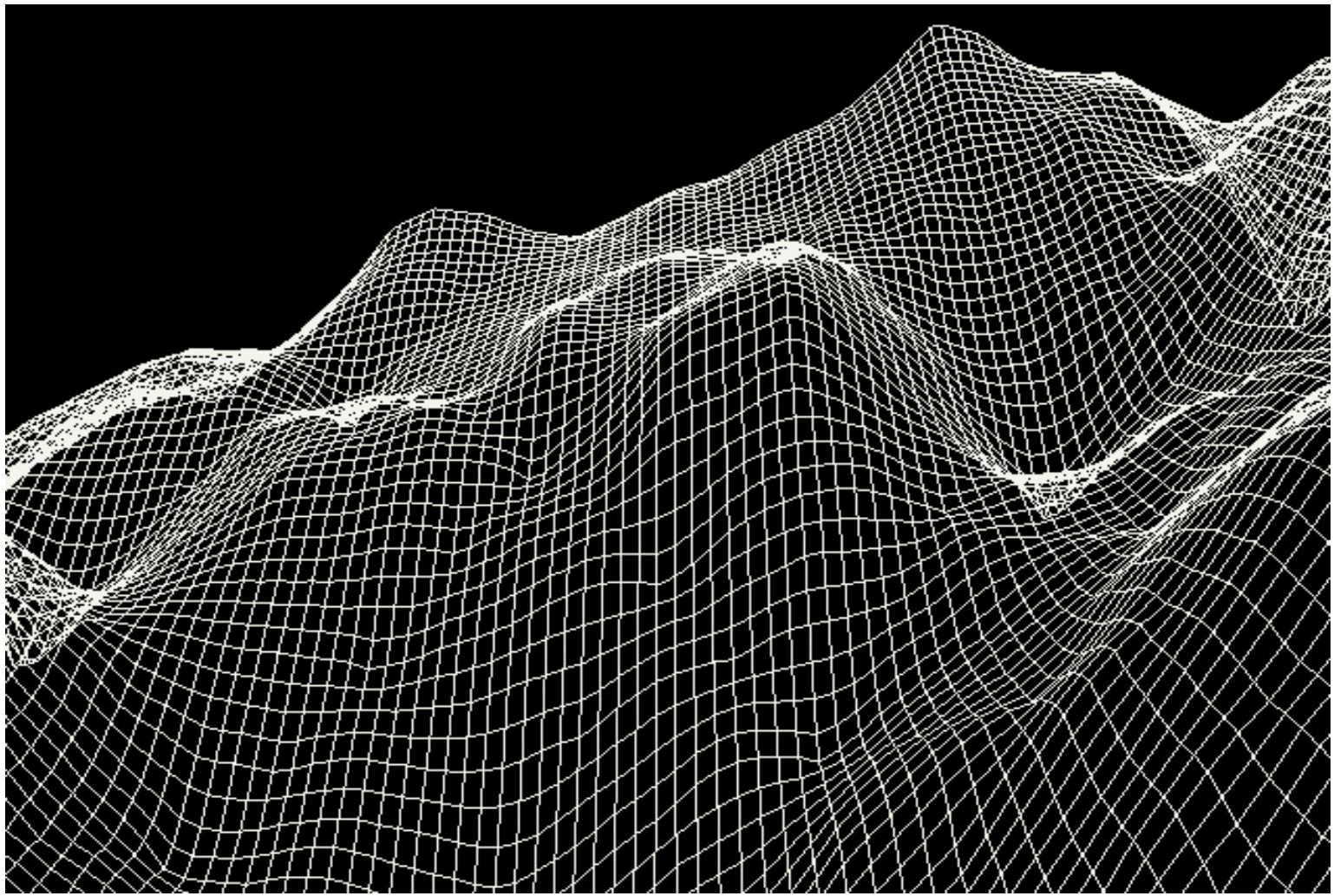
```
glTranslatef(0.0f,0.0f,-6.0f);

glBegin(GL_QUADS);
    glColor3f ( 0.0, 1.0, 0.0);
    glVertex2f(-1.0,-1.0);
    glColor3f ( 0.0, 1.0, 1.0);
    glVertex2f( 1.0,-1.0);
    glColor3f ( 0.0, 0.0, 1.0);
    glVertex2f( 1.0, 1.0);
    glColor3f ( 1.0, 0.0, 0.0);
    glVertex2f(-1.0, 1.0);
glEnd();
```



```
glTranslatef(0.0f,0.0f,-6.0f);
glRotatef(45,1.0,1.0,0.0);
glBegin(GL_QUADS);
    glColor3f ( 0.0, 1.0, 0.0);
    glVertex2f(-1.0,-1.0);
    glColor3f ( 0.0, 1.0, 1.0);
    glVertex2f( 1.0,-1.0);
    glColor3f ( 0.0, 0.0, 1.0);
    glVertex2f( 1.0, 1.0);
    glColor3f ( 1.0, 0.0, 0.0);
    glVertex2f(-1.0, 1.0);
glEnd();
```



Schneller = Besser

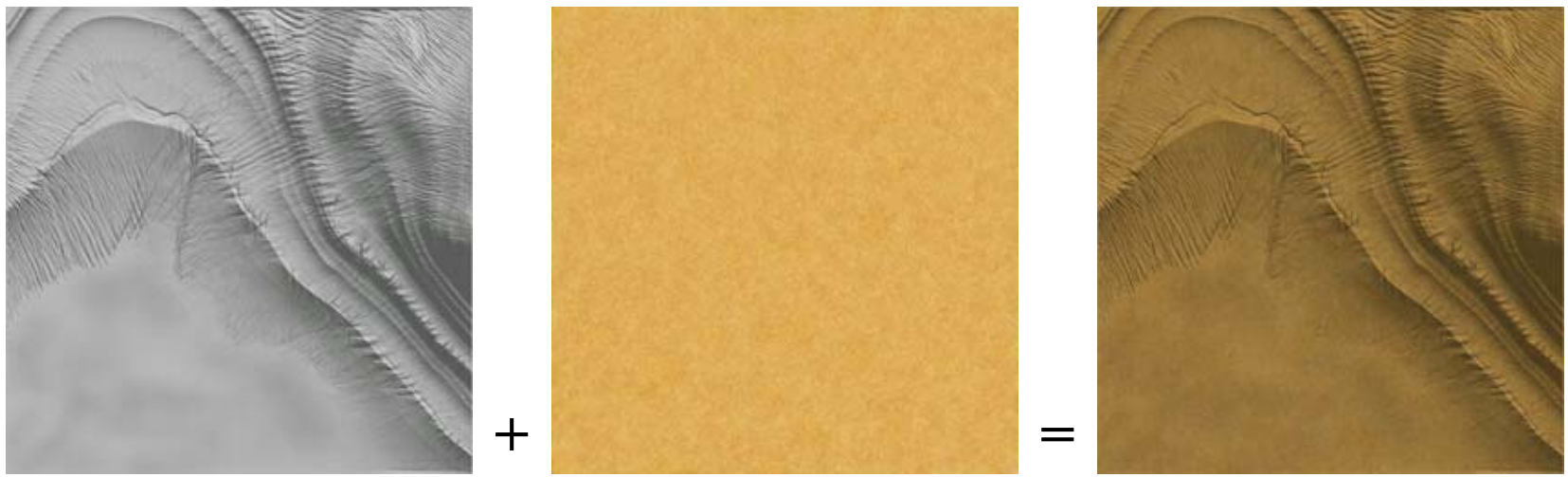


- Reduzierung der Vertices die gezeichnet werden:
  - Back Face Culling
  - Frustum Culling
- Optimierung des Zeichenvorgangs:
  - Display Listen
  - Vertex Arrays/Buffer

- Ab OpenGL 1.4 durch das ARB hinzugefügt
- Programmierbare Rendering-Pipeline
  - Realistische Lichtberechnungen
  - Flexiblere Überblendung von Texturen
- OpenGL Shading Language (GLSL)
- Unterteilung in Vertex- und Fragmentshader(Pixelshader)

- Wird auf jedem gezeichnetem Vertex ausgeführt
- Belastet nur die GPU der Grafikkarte
- Man kann Position und Aussehen jedes Vertex ändern
- Emulierbar
  - Bsp: Geforce4 MX
    - Echter Fragmentshader in Chip integriert
    - Jedoch Vertexshader nur durch Treiber simuliert

- Fragmente:
  - Enthalten Farb-,Tiefe-,Texturdaten
  - Materialeigenschaften
- Wird häufig für Multitexturing benutzt



OpenGL:

- BS unabhängig
- OpenSource
  - Selbst erweiterbar
- Weiterentwicklung durch ARB

- Beispielprogramm Dreieck:

```
glBegin (GL_TRIANGLES);
        glVertex (0,0,0);
        glVertex (1,1,0);
        glVertex (2,0,0);
glEnd ();
```

Direct3D:

- Windows
- Eigentum von M$

- Weiterentwicklung durch M$ alleine

- Beispielprogramm Dreieck:

```
D3DVERTEX g_pvTriangleVertices[3];
D3DVECTOR p1( 0.0f, 3.0f, 0.0f );
D3DVECTOR p2( 3.0f,-3.0f, 0.0f );
D3DVECTOR p3(-3.0f,-3.0f, 0.0f );
g_pvTriangleVertices[0] = D3DVERTEX( p1, vNormal, 0, 0 );
g_pvTriangleVertices[1] = D3DVERTEX( p2, vNormal, 0, 0 );
g_pvTriangleVertices[2] = D3DVERTEX( p3, vNormal, 0, 0 );
pd3dDevice->DrawPrimitive( D3DPT_TRIANGLELIST,
D3DFVF_VERTEX, g_pvTriangleVertices, 3, NULL
```

- OpenGL
- Vorteile:
  - plattformunabhängig
  - Open-source Referenzimplementierung
  - Meist bessere Treiber für professionelle Grafikhardware
- Nachteile:
  - Neuer Standard dauert lange
  - Extensions-Chaos
  - Keine bzw. nur schlechte Treiberunterstützung auf preiswerter Hardware

- Direct3D
- Vorteile:
  - Schnell neuer Standard verfügbar
  - Programmiersprachenunabhangig (COM)
  - DirectX immer auch verfügbar (Sound…)
  - Standard meist weiter als Hardwareentwicklung
  - Software-Emulator für alle Features
  - Meist bessere Treiber für Low-Cost Grafikkarten
- Nachteile
  - Plattformabhängig (Windows)
  - Proprietär (Microsoft)
  - Closed-source
  - Oft starke Änderungen bei Versions-wechseln

Welche API benutze ich jetzt?

Fragen?