OpenGL- Teil 1

Gliederung

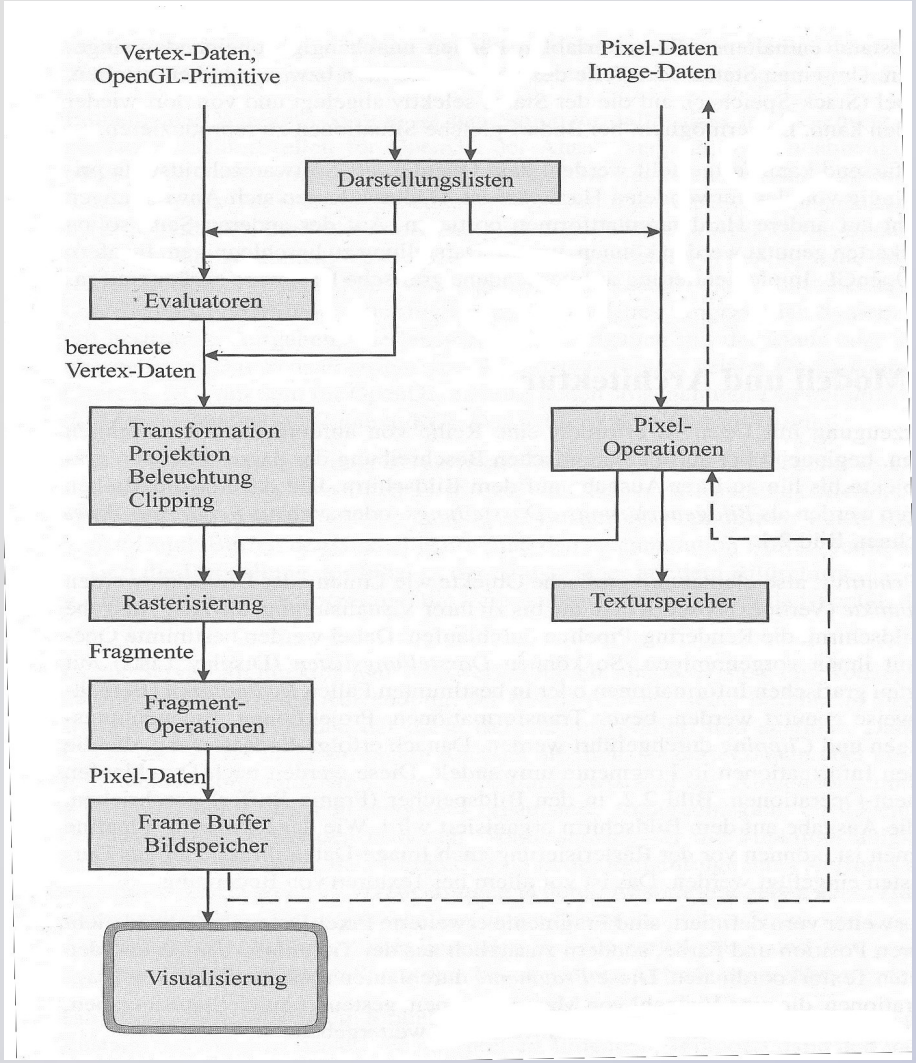
- Einführung
  - Geschichte
  - Renderpipeline
  - Konzepte
- Transformationen
- Primitive
- Farben
  - Farben setzen
  - Colorpicking
- Texturen
  - Mipmaps
- Quellen

Einführung- Geschichte

- Ursprünglich von Silicon Graphics entwickelt
- Von 1992 bis 2006 unter der Leitung des OpenGL ARB
  - 01.07.1992: OpenGL 1.0
  - 07.09.2004: OpenGL 2.0
- 2006 übernahm die Khronos Group die Weiterentwicklung
  - 02.08.2006: OpenGL 3.0
  - 11.03.2010: OpenGL 4.0

→ sehr schnelle Weiterentwiklung

Einführung- Renderpipeline



- Rendern: Bearbeiten von Rohdaten in neue Daten

  3D → 2D

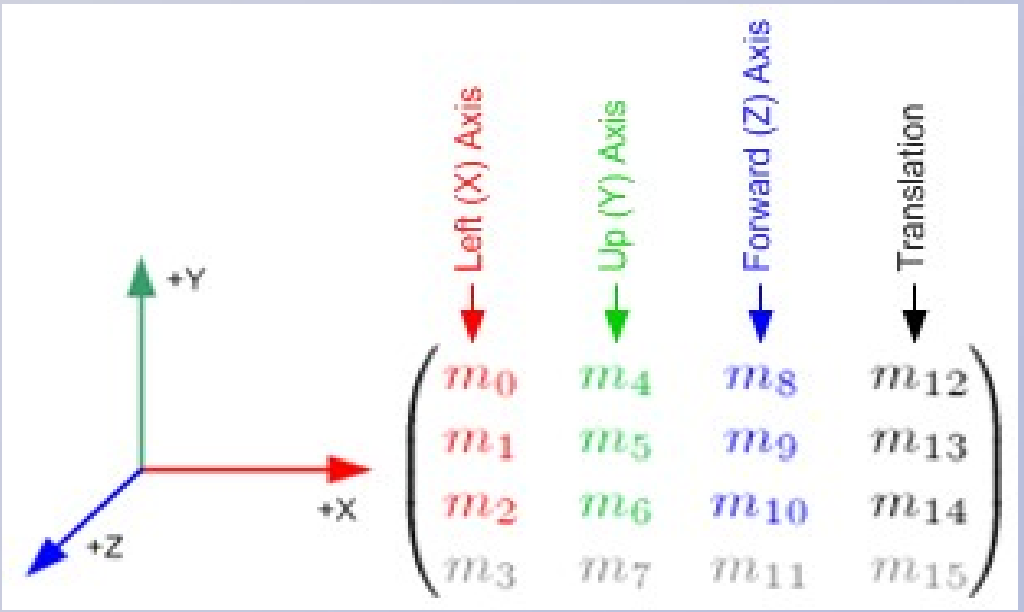
- Pipeline: Menge von Berechnungseinheiten, welche „Pseudoparallelität“ ermöglichen
- Der Monitor holt sich das Bild direkt aus dem Framebuffer

Einführung- Konzepte

- Plattformunabhänigkeit
- OpenGL ist als State Maschine entwickelt

```
glColor3f(1, 0, 0);     //Farbe auf Rot setzen
ZeichneDreieck();
glColor3f(0, 0, 1);     //Farbe auf Blau setzen
ZeichneDreieck();
```

- 3 Matrizen
  - Modelmatrix
  - Perspektivmatrix
  - Texturmatrix

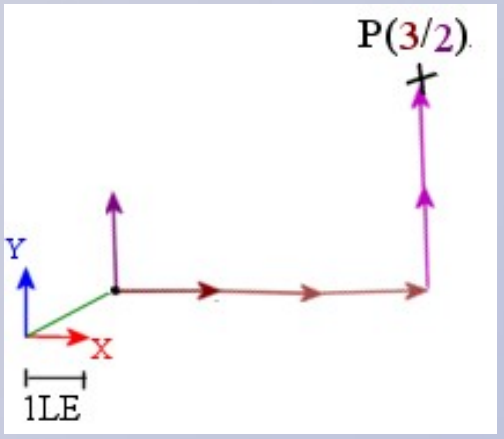


Transformationen

- Translation

glTranslatef(a, b, c);

$$\begin{pmatrix} x_{neu} \\ y_{neu} \\ z_{neu} \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{pmatrix} . \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$



- Rotation

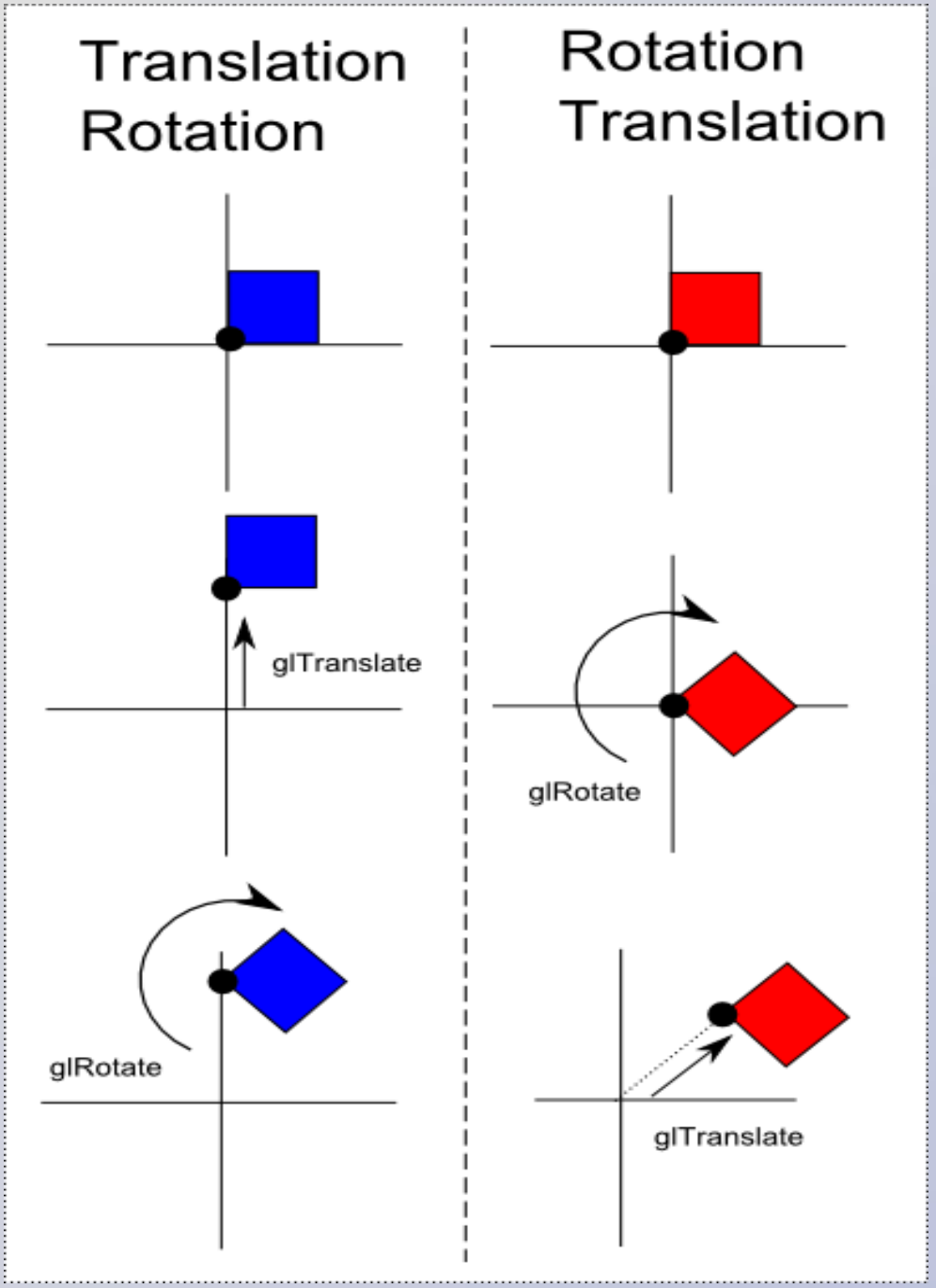
glRotatef(α, 0, 0, 1);

$$\begin{pmatrix} x_{neu} \\ y_{neu} \\ z_{neu} \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} . \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

- Skalieren

glScalef(a, b, c);

$$\begin{pmatrix} x_{neu} \\ y_{neu} \\ z_{neu} \\ 1 \end{pmatrix} = \begin{pmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & c & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} . \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$


Translation
Rotation
glTranslate
glRotate
Rotation
Translation
glRotate
glTranslate

Primitive

→ Kleine Einführung in den Code...

[*] main.cpp

```
1  #include <SDL/SDL.h>
2  #include <GL/gl.h>
3  #include <GL/glu.h>
4
5  #include "init.h"
6
7  #define MAXKEY 50
8
9
10 void points();
11 void lines();
12 void triangle();
13 void triangles();
14 void quad();
15
16 int main( int argc, char* argv[] )
17 {
18     int done=0, key[MAXKEY];
19     float alpha=0;
20     for( int i=0; i<MAXKEY; i++ ) key[i]=0;
21     gl_init(800,600);
22
23
24 // Hauptschleife
25     while( !key[0] ){
26     //Eingabe
27         input(key);
28
29
30     //Zeichnen
31         glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );
32
33         //Verschieben, Drehen, etc.
34         glLoadIdentity();
35         glTranslatef(0,0,-3);
36         //glRotatef(alpha,0,0,1);
37
38         glTranslatef(-1,0,0);
39         glColor3f(1,0,0);       //Farbe setzen
40         triangle();             //zeichnen
41
42         glTranslatef(2,0,0);
43         glColor3f(0,0,1);       //Farbe setzen
44         triangle();             //zeichnen
45
46         SDL_GL_SwapBuffers();
47     }
48     SDL_Quit();
49     return 0;
50 }
```

Primitive

- Points

```
glBegin(GL_POINTS);
    glVertex3f(0.5, 0, 0);
    glVertex3f(0, 0.5, 0);
glEnd();
```

- Lines
  - GL_LINES
  - GL_LINE_STRIP
  - GL_LINE_LOOP

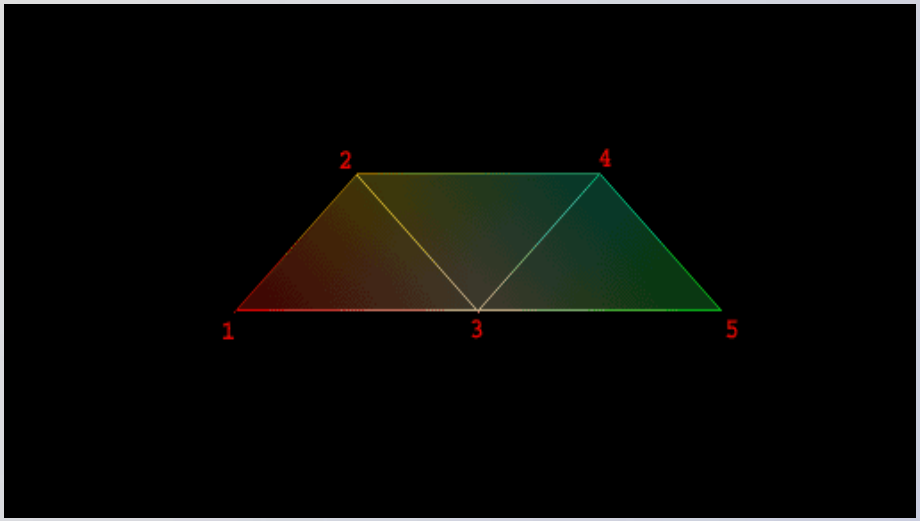
```
glBegin(GL_LINES);
    glVertex3f(0.5, 0, 0);
    glVertex3f(0, 0.5, 0);
glEnd();
```

- Triangles
  - GL_TRIANGLES
  - GL_TRIANGLE_STRIP
  - GL_TRIANGLE_FAN

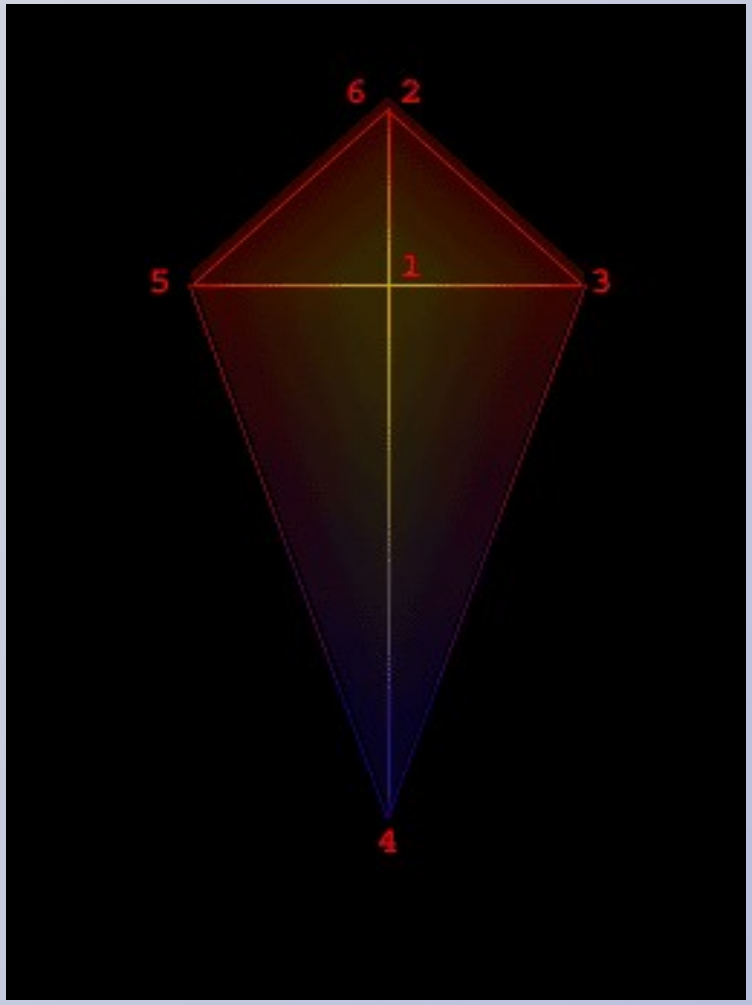
```
glBegin(GL_TRIANGLES);
      glVertex3f(-1,0.3,0);
      glVertex3f(-1.5,-0.5,0);
      glVertex3f(-0.5,-0.5,0);
glEnd();
```

Primitive

Triangle Strip



Triangle Fan



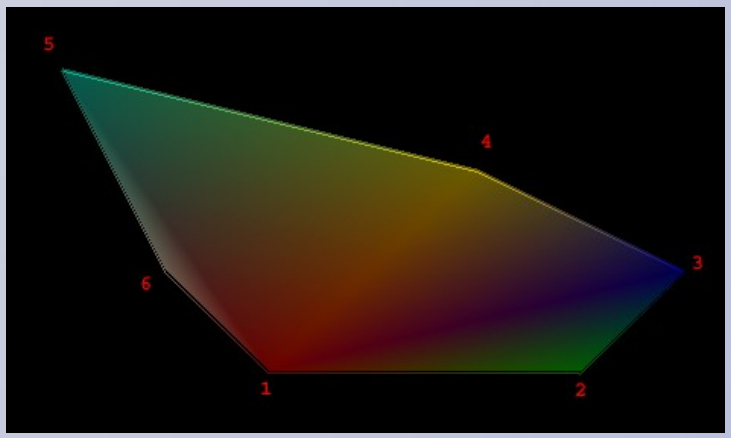
Primitive

- Quads
  - GL_QUADS
  - GL_QUAD_STRIP

```
glBegin(GL_QUADS);
        glVertex3f(-0.5,0.5,0);
        glVertex3f(0.5,0.5,0);
        glVertex3f(0.5,-0.5,0);
        glVertex3f(-0.5,-0.5,0);
    glEnd();
```

- Polygone

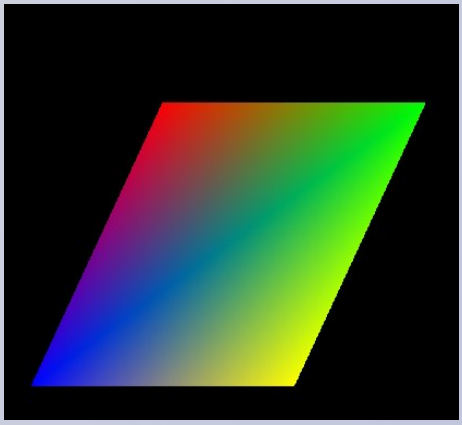
```
glBegin(GL_POLYGON);
    glVertex3f(0,0,0);
    glVertex3f(3,0,0);
    glVertex3f(4,1,0);
    glVertex3f(2,2,0);
    glVertex3f(-2,3,0);
    glVertex3f(-1,1,0);
  glEnd;
```



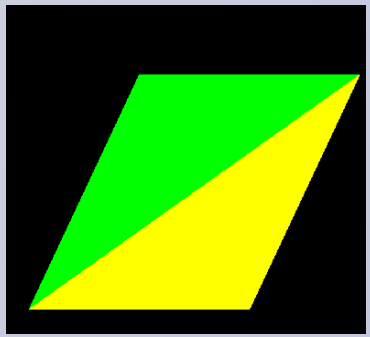
Farben

Farben setzten

```
glColor3f(1,1,1);
glColor4f(1,1,1,1);
glColor3ub(255,255,255);
```



- GL_SMOOTH: weicher Farbverlauf



- GL_FLAT: keine Interpolation

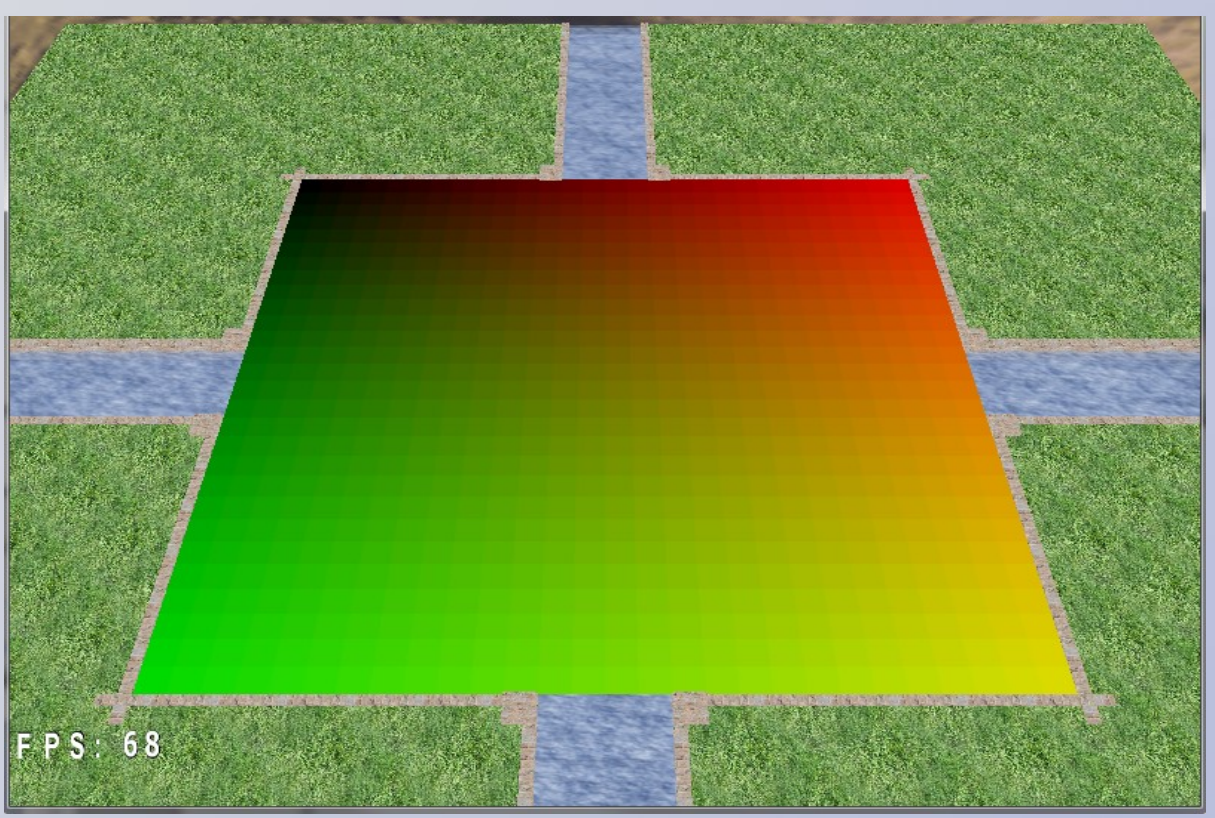
- Bei gesetzter Farbe wird diese mit einer Textur multipliziert

Farben

Farbe auslesen (Colorpicking)

- Idee: Die Scene wird ein 2. Mal nur mit charakteristischen Farben gerendert
- Auslesen der Farbinfomation

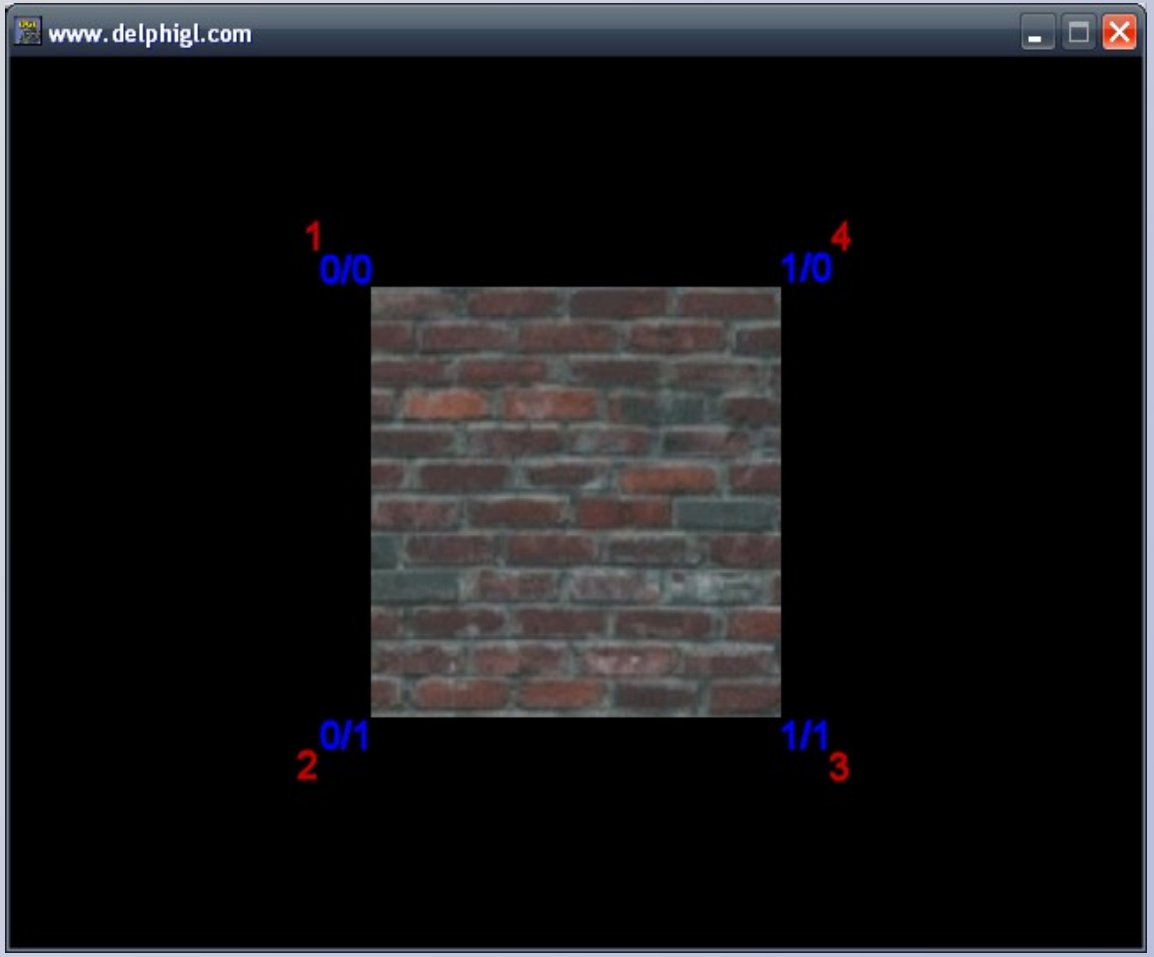
```
unsignet char red;
glReadPixels(x, y, 1, 1, GL_RED,
GL_UNSIGNED_BYTE, &red);
```



Texturen

- Zusätzliche u-v-Koordinaten
- Übergabe vor Eckpunkten

```
glBegin(GL_QUADS);
  glTexCoord2f(0,0);
    glVertex3f(-1,1,0);  //lo
  glTexCoord2f(0,1);
    glVertex3f(-1,-1,0); //lu
  glTexCoord2f(1,1);
    glVertex3f(1,-1,0);  //ru
  glTexCoord2f(1,0);
    glVertex3f(1,1,0);   //ro
glEnd;
```



Texturen- Mipmaps

- Verbesserte Darstellung von verkleinerten Bildern
- Umrechnen der Bilder durch Trilineare Filterung
- Mehrere Versionen eines Bildes im Speicher

Bild mit einfacher Verkleinerung

Bild durch Trilinearen Filter verkleinert

Quellen

- http://www.delphigl.com/
- http://www.opengl.org/
- http://www.gamedev.de/index.php
- www.c3t.de/pub/vortraege/20070404-computergrafik/computergrafik.pdf
- http://www.songho.ca/opengl/gl_anglestoaxes.html