Grafische Datenverarbeitung: Texturen mit OpenGL

Einleitung

Es existieren zahlreiche Methoden um Texturen für die Verwendung in OpenGL zu laden. Im folgenden wird ein Weg gezeigt, der für mehr als 100 verschiedene Bildformate verwendet werden kann.

ImageMagick Einführung

Für folgende Erläuterung wird die ImageMagick Programmsuite genutzt. Mehr Informationen zu dieser Software sind auf der Webseite **www.imagemagick.org** zu finden. Für diese Software existiert ein C++-Wrapper (Magick++) der hier zum programmieren verwendet wird.

Kdevelop einrichten

Ein neues Projekt erstellen, alle Einstellungen für ein OpenGL Programm vornehmen und zusätzlich noch folgende Zeile mit in die LDFLAGS übernehmen:

LDFLAGS += `Magick++-config --cppflags --cxxflags --ldflags –libs`

Diese Zeile bewirkt, dass alle wichtigen Informationen, die zum Linken des Programms nötig sind, hinzugefügt werden.

Magick++ API

Die Dokumentation des Wrappers ist unter folgendem Link zum Download verfügbar:

http://www.imagemagick.org/Magick++/tutorial/Magick++_tutorial.pdf

```
#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glut.h>
#include <Magick++.h>

string filenames[] = {      "bilder/test.gif", "bilder/test.jpg",  "bilder/test.bmp" };
#define MAXTEXTURES 3
GLuint textures[MAXTEXTURES];
GLint textureid = 0;

void RenderScene() {
            glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
            glLoadIdentity();
            glTranslatef( xpos, ypos, zpos );
            glRotatef( xrot, 1.0f, 0.0f, 0.0f );
            glRotatef( yrot, 0.0f, 1.0f, 0.0f );
            glColor3f( 1.0f, 1.0f, 1.0f );

            #define HWIDTH 1.0f
            #define P1() glVertex3f(-HWIDTH, HWIDTH, HWIDTH)
            #define P2() glVertex3f(-HWIDTH, -HWIDTH, HWIDTH)
            #define P3() glVertex3f(HWIDTH, -HWIDTH, HWIDTH)
            #define P4() glVertex3f(HWIDTH, HWIDTH, HWIDTH)
            #define P5() glVertex3f(-HWIDTH, HWIDTH, -HWIDTH)
            #define P6() glVertex3f(-HWIDTH, -HWIDTH, -HWIDTH)
            #define P7() glVertex3f(HWIDTH, -HWIDTH, -HWIDTH)
            #define P8() glVertex3f(HWIDTH, HWIDTH, -HWIDTH)

            /* Bild auf die Fläche setzen. */
            #define T1() glTexCoord2f(0.0f, 1.0f)
            #define T2() glTexCoord2f(0.0f, 0.0f)
            #define T3() glTexCoord2f(1.0f, 0.0f)
            #define T4() glTexCoord2f(1.0f, 1.0f)

            /* Textur laden und an alle weiteren Flächen binden. */
            glBindTexture( GL_TEXTURE_2D, textures[textureid] );

            glBegin( GL_QUADS );
                        glNormal3f( 0.0f, 0.0f, 1.0f );
                        T1(); P1();
                        T2(); P2();
                        T3(); P3();
                        T4(); P4();
                        ///
                        /// weitere Seiten ...
                        ///
            glEnd();
            #undef HWIDTH
            glutSwapBuffers();
}

void ChangeSize(int w, int h) {  /* Bekannt! */   }

void KeyboardFunc( unsigned char key, int x, int y ) {
    switch(key) {
      case '+':
            textureid += 1;
            if( textureid >= MAXTEXTURES ) textureid = 0;
      break;
    }
    glutPostRedisplay();
}
```

```
void SetupRC() {
    glClearColor(0.0f, 0.0f, 0.0f, 0.0f);
    glEnable(GL_DEPTH_TEST);
    glEnable( GL_CULL_FACE );
    glShadeModel(GL_SMOOTH);

    Magick::Image img;        // Objekt das ein Bild bearbeitet.
    Magick::Blob blob;        // Objekt das die „raw“ Bilddaten beinhalten kann.

    glGenTextures( MAXTEXTURES, textures );

    for( int i=0; i<MAXTEXTURES; i++ )
    {
      try {
        /* Um die Verwendung zu vereinfachen wird das Bild geladen und im Speicher
         * ins RGBA Format umgewandelt.
         */
        img.read(filenames[i]);
        img.magick("RGBA");
        img.write(&blob);

        glBindTexture(GL_TEXTURE_2D, textures[i]);
        glTexParameteri( GL_TEXTURE_2D,
                         GL_TEXTURE_MAG_FILTER, GL_LINEAR );
        glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
                         GL_LINEAR_MIPMAP_LINEAR );
        glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_WRAP_S,
                         GL_CLAMP_TO_EDGE );
        glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_WRAP_T,
                         GL_CLAMP_TO_EDGE );
        gluBuild2DMipmaps( GL_TEXTURE_2D, 4,
                           img.columns(), img.rows(),
                           GL_RGBA, GL_UNSIGNED_BYTE, blob.data() );
      } catch( ... ) {
        cout << "Die Datei " << filenames[i] <<
                " konnte nicht geladen werden!" << endl;
      }
    }
    glEnable(GL_TEXTURE_2D);
}

void TimerFunc( int data ) {
            glutPostRedisplay();
            glutTimerFunc(100, TimerFunc, 0);
}

int main( int argc, char **argv ) {
            glutInit(&argc, argv);
            glutInitWindowSize( 600, 400 );
            glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
            glutCreateWindow("ImageMagick Beispiel");
            SetupRC();
            glutReshapeFunc(ChangeSize);
            glutDisplayFunc(RenderScene);
            glutKeyboardFunc(KeyboardFunc);
            glutTimerFunc(100, TimerFunc, 0);
            glutMainLoop();
            return 0;
}
```