JPEG-bilder och OpenGL

2007-03-21

Gustav Taxén, gustavt@csc.kth.se

Till OpenGL-labben använder vi *Independent JPEG Groups* bibliotek `libjpeg` för att ladda texturfiler från disk. (Källkoden finns att hämta från http://www.ijg.org/.)

En JPEG-bild har oftast 1 (grayscale) eller 3 (RGB) komponenter per pixel. OpenGL kan arbeta med 1 komponent per pixel (då antar man normalt att texturen inte beskriver färg utan har formatet `GL_LUMINANCE`), men det vanligaste är att man använder RGB. Om din textur är i grayscale, rekommenderas att du konverterar den till RGB i Photoshop (eller motsvarande program) först.

Utdata från `libjpeg` är av typen `unsigned byte` (som motsvaras av `unsigned char` i C/C++). Det betyder att du ska ange `type`-parametern `GL_UNSIGNED_BYTE` då bilden skickas till OpenGL med `glTexImage2D()`.

JPEG-bilder läses scanline-för-scanline från disk, med start från första raden (överst i bilden). OpenGL placerar dock bildorigo i nedre vänstra hörnet. Det betyder att bilderna måste vändas upp och ned innan de skickas till OpenGL (eller att texturkoordinaters y-värden måste inverteras). På nästa sida finns ett exempel som vänder bilden då den läses från disk. Exempelkoden finns inkopierad i skalprogrammet som hör till labben.

Eftersom `libjpeg` kan vara knepigt att kompilera under Windows finns i kurskatalogen också ett arkiv där en färdigkompilerad version ingår.

```
#include <jpeglib.h>

...

unsigned char *loadJPEG(const char *filename,
                        unsigned int *w,
                        unsigned int *h,
                        unsigned int *components) {

  struct jpeg_decompress_struct cinfo;
  struct jpeg_error_mgr jerr;
  FILE * infile;
  unsigned char *buffer, *ptr;
  unsigned int nval;

  cinfo.err = jpeg_std_error(&jerr);
  jpeg_create_decompress(&cinfo);

  if ((infile = fopen(filename, "rb")) == NULL) {
    fprintf(stderr, "Can't open '%s'\n", filename);
    jpeg_destroy_decompress(&cinfo);
    return NULL;
  }

  jpeg_stdio_src(&cinfo, infile);
  jpeg_read_header(&cinfo, TRUE);
  jpeg_start_decompress(&cinfo);

  *w = cinfo.output_width;
  *h = cinfo.output_height;
  *components = cinfo.output_components;

  nval = cinfo.output_width * cinfo.output_components * cinfo.output_height;
  buffer = (unsigned char *)malloc(sizeof(unsigned char) * nval);
  if (!buffer) {
    fprintf(stderr, "Out of memory\n");
    jpeg_finish_decompress(&cinfo);
    jpeg_destroy_decompress(&cinfo);
    fclose(infile);
    return NULL;
  }

  /* We want to flip the image over the x-axis, so start reading at
     the last row and move backwards. */

  ptr = buffer +
    (cinfo.output_width * cinfo.output_components) * (cinfo.output_height - 1);

  while (cinfo.output_scanline < cinfo.output_height) {
    jpeg_read_scanlines(&cinfo, &ptr, 1);
    ptr -= cinfo.output_width * cinfo.output_components;
  }

  jpeg_finish_decompress(&cinfo);
  jpeg_destroy_decompress(&cinfo);
  fclose(infile);

  return buffer;
}
```