Day7

#include <string.h>

#include <stdlib.h>

#include <stdio.h>

#include <GL/glut.h>

GLenum doubleBuffer;

GLubyte ubImage[65536];

static void

Init(void)

{

  int j;

  GLubyte \*img;

  GLsizei imgWidth = 128;

  glMatrixMode(GL\_PROJECTION);

  glLoadIdentity();

  gluPerspective(60.0, 1.0, 0.1, 1000.0);

  glMatrixMode(GL\_MODELVIEW);

  glDisable(GL\_DITHER);

  /\* Create image \*/

  img = ubImage;

  for (j = 0; j < 32 \* imgWidth; j++) {

    \*img++ = 0xff;

    \*img++ = 0x00;

    \*img++ = 0x00;

    \*img++ = 0xff;

  }

  for (j = 0; j < 32 \* imgWidth; j++) {

    \*img++ = 0xff;

    \*img++ = 0x00;

    \*img++ = 0xff;

    \*img++ = 0x00;

  }

  for (j = 0; j < 32 \* imgWidth; j++) {

    \*img++ = 0xff;

    \*img++ = 0xff;

    \*img++ = 0x00;

    \*img++ = 0x00;

  }

  for (j = 0; j < 32 \* imgWidth; j++) {

    \*img++ = 0x00;

    \*img++ = 0xff;

    \*img++ = 0x00;

    \*img++ = 0xff;

  }

}

/\* ARGSUSED1 \*/

static void

Key(unsigned char key, int x, int y)

{

  switch (key) {

  case 27:

    exit(0);

  }

}

void

TexFunc(void)

{

  glEnable(GL\_TEXTURE\_2D);

  glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_S, GL\_REPEAT);

  glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_T, GL\_REPEAT);

  glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_NEAREST);

  glTexEnvf(GL\_TEXTURE\_ENV, GL\_TEXTURE\_ENV\_MODE, GL\_DECAL);

#if GL\_EXT\_abgr

  glTexImage2D(GL\_TEXTURE\_2D, 0, 3, 128, 128, 0, GL\_ABGR\_EXT,

    GL\_UNSIGNED\_BYTE, ubImage);

  glBegin(GL\_POLYGON);

  glTexCoord2f(1.0, 1.0);

  glVertex3f(-0.2, 0.8, -100.0);

  glTexCoord2f(0.0, 1.0);

  glVertex3f(-0.8, 0.8, -2.0);

  glTexCoord2f(0.0, 0.0);

  glVertex3f(-0.8, 0.2, -2.0);

  glTexCoord2f(1.0, 0.0);

  glVertex3f(-0.2, 0.2, -100.0);

  glEnd();

#endif

  glTexImage2D(GL\_TEXTURE\_2D, 0, 3, 128, 128, 0, GL\_RGBA,

    GL\_UNSIGNED\_BYTE, ubImage);

  glBegin(GL\_POLYGON);

  glTexCoord2f(1.0, 1.0);

  glVertex3f(0.8, 0.8, -2.0);

  glTexCoord2f(0.0, 1.0);

  glVertex3f(0.2, 0.8, -100.0);

  glTexCoord2f(0.0, 0.0);

  glVertex3f(0.2, 0.2, -100.0);

  glTexCoord2f(1.0, 0.0);

  glVertex3f(0.8, 0.2, -2.0);

  glEnd();

  glDisable(GL\_TEXTURE\_2D);

}

static void

Draw(void)

{

  glClearColor(0.0, 0.0, 0.0, 1.0);

  glClear(GL\_COLOR\_BUFFER\_BIT);

#if GL\_EXT\_abgr

  glRasterPos3f(-0.8, -0.8, -1.5);

  glDrawPixels(128, 128, GL\_ABGR\_EXT, GL\_UNSIGNED\_BYTE, ubImage);

#endif

  glRasterPos3f(0.2, -0.8, -1.5);

  glDrawPixels(128, 128, GL\_RGBA, GL\_UNSIGNED\_BYTE, ubImage);

  TexFunc();

  if (doubleBuffer) {

    glutSwapBuffers();

  } else {

    glFlush();

  }

}

static void

Args(int argc, char \*\*argv)

{

  GLint i;

  doubleBuffer = GL\_TRUE;

  for (i = 1; i < argc; i++) {

    if (strcmp(argv[i], "-sb") == 0) {

      doubleBuffer = GL\_FALSE;

    } else if (strcmp(argv[i], "-db") == 0) {

      doubleBuffer = GL\_TRUE;

    }

  }

}

int

main(int argc, char \*\*argv)

{

  GLenum type;

  glutInit(&argc, argv);

  Args(argc, argv);

  type = GLUT\_RGB;

  type |= (doubleBuffer) ? GLUT\_DOUBLE : GLUT\_SINGLE;

  glutInitDisplayMode(type);

  glutCreateWindow("ABGR extension");

  if (!glutExtensionSupported("GL\_EXT\_abgr")) {

    printf("Couldn't find abgr extension.\n");

    exit(0);

  }

#if !GL\_EXT\_abgr

  printf("WARNING: client-side OpenGL has no ABGR extension support!\n");

  printf("         Drawing only RGBA (and not ABGR) images and textures.\n");

#endif

  Init();

  glutKeyboardFunc(Key);

  glutDisplayFunc(Draw);

  glutMainLoop();

  return 0;             /\* ANSI C requires main to return int. \*/

}

Output:

