main.c

```
#include <windows.h>   /* Required on windows platforms */
#include <math.h>      /* To access sin(), cos() etc. */
#include <GL/glut.h>   /* GLUT includes glu.h and gl.h automatically */
#include "jpeg.h"      /* Where the loadJPEG() function is declared */

/* Animation variables, updated by idle() */
float anim = 0.0f;
float hopp = 0.0f;

/* Variable to hold a texture object ID */
GLuint texID;

void createTexture(void) {
      unsigned char *pixels; /* Unsigned char = unsigned byte in C */
      unsigned int width, height, components;

      /* Load a JPEG image from disk and obtain a
         pointer to the memory location where its
         decoded pixels are. loadJPEG writes the
         width, height, and number of components
         (3 for RGB pictures) into the three
         variables "width", "height" and "components",
         respectively.
      */
      pixels = loadJPEG("wood.jpg", &width, &height, &components);

      /* Write a new texture object ID into variable "texID"... */
      glGenTextures(1, &texID);

      /* ...and make it the active texture object. */
      glBindTexture(GL_TEXTURE_2D, texID);

      /* Set the magnification and minification filters for
         the active texture object. Also, set the texture to
         wrap in the s and t directions.
      */
      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);

      /* Pass the loaded image to OpenGL and store it
         in the active texture object.
      */
      glTexImage2D(GL_TEXTURE_2D,
            0, /* MIP-map level - set to 0 unless you have mip-maps. */
            GL_RGB, /* The image is a RGB image */
            width,  /* Witdh must be a power of 2! */
            height, /* Height must be a power of 2! */
            0, /* Border - always set to 0. */
            GL_RGB, /* Same as above */
            GL_UNSIGNED_BYTE, /* See loadJPEG() */
            pixels);
}
```

```
/* Update animation parameters and force a redraw of the window */
void idle(void) {
      anim += 0.05f;
      hopp = sin(0.1 * anim);

      glutPostRedisplay();    /* Force GLUT to redraw the window */
}

/* Called when window changes size */
void reshape(int w, int h) {
      /* Set a perspective projection */
      glMatrixMode(GL_PROJECTION);
      glLoadIdentity();
      gluPerspective(45.0f, (float)w / (float)h, 0.1f, 50.0f);

      /* Make sure the OpenGL drawing fills the entire window */
      glViewport(0, 0, w, h);
}

/* Set parameters for light number 0 */
void setLight(void) {
      float lamb[4] = { 1.0, 1.0, 1.0, 1.0 };
      float ldif[4] = { 1.0, 1.0, 1.0, 1.0 };
      float lspc[4] = { 1.0, 1.0, 1.0, 1.0 };

      /* w = 0 means directional light source */
      float lpos[4] = { 0.0f, 1.0f, 0.0f, 0.0f };

      glEnable(GL_LIGHT0);
      glLightfv(GL_LIGHT0, GL_AMBIENT, lamb);
      glLightfv(GL_LIGHT0, GL_DIFFUSE, ldif);
      glLightfv(GL_LIGHT0, GL_SPECULAR, lspc);
      glLightfv(GL_LIGHT0, GL_POSITION, lpos);
}

/* Set a red plastic material */
void setMaterial(void) {
      float amb[4] = { 0.1f, 0.0f, 0.0f, 0.0f };
      float dif[4] = { 1.0f, 0.0f, 0.0f, 0.0f };
      float spc[4] = { 1.0f, 1.0f, 1.0f, 1.0f };

      glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT, amb);
      glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, dif);
      glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, spc);
      glMaterialf(GL_FRONT_AND_BACK, GL_SHININESS, 10.0f);
}

/* Called when user presses a key */
void keyboard(unsigned char key, int x, int y) {
      if (key == 'q' || key == 27) { /* 27 = Escape */
            exit(0);
      }
}
```

```
void drawRobot(void) {
      glutSolidSphere(0.5f, 20, 20);

      /* Translate to position of upper arm pivot */
      glTranslatef(0.5, 0, 0);
      /* Rotate around upper arm pivot */
      glRotatef(anim, 1, 0, 0);

      glPushMatrix();
      /* Move halfway down the upper arm - the reason is
         that the cube drawn by glutSolidCube() is
         centered around the current origo */
      glTranslatef(0, 0, 0.5);
      /* Scale the upper arm and draw it */
      glScalef(0.25f, 0.25f, 1.0f);
      glutSolidCube(1.0f);
      /* Return to pivot */
      glPopMatrix();

      /* Draw lower arm: first move down to
         the position of the lower arm pivot */
      glTranslatef(0, 0, 1.0);
      /* Rotate around lower arm pivot */
      glRotatef(-anim * 2.0f, 1, 0, 0);
      /* Move halfway down the lower arm */
      glTranslatef(0, 0, 0.5);
      /* Scale the lower arm and draw it */
      glScalef(0.24f, 0.24f, 0.9f);
      glutSolidCube(1.0f);
}

void display(void) {
      /* Clear the window and the depth buffer */
      glClearColor(0.1, 0.2, 0.3, 1.0);
      glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

      /* Enable the depth test */
      glEnable(GL_DEPTH_TEST);

      /* Make sure normals are always of length 1, even
         if we're scaling */
      glEnable(GL_NORMALIZE);

      /* Set the modelview matrix */
      glMatrixMode(GL_MODELVIEW);
      glLoadIdentity();

      setLight(); /* Set light pos before changing modelview matrix */

      gluLookAt(
            3, 5, 15,
            0, 0, 0,
            0, 1, 0);
```

```
        /* Draw the ground polygon */

        /* Enable texturing */
        glEnable(GL_TEXTURE_2D);
        /* Activate our texture object */
        glBindTexture(GL_TEXTURE_2D, texID);
        /* Replace colors with the contents of the texture */
        glTexEnvi(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_REPLACE);

        /* Draw a quadrilateral (4-sided polygon) */
        glBegin(GL_QUADS);
        glTexCoord2f(0.0f, 0.0f);
        glVertex3f(-10.0f, 0.0f, 10.0f);
        glTexCoord2f(1.0f, 0.0f);
        glVertex3f(10.0f, 0.0f, 10.0f);
        glTexCoord2f(1.0f, 1.0f);
        glVertex3f(10.0f, 0.0f, -10.0f);
        glTexCoord2f(0.0f, 1.0f);
        glVertex3f(-10.0f, 0.0f, -10.0f);
        glEnd();

        /* Disable texturing */
        glDisable(GL_TEXTURE_2D);


        /* Position the robot */
        glRotatef(anim, 0, 1, 0);
        glTranslatef(5, 0, 0);

        /* Draw the robot's shadow: */
        glPushMatrix();
        /* Move up a bit so shadow doesn't interfer with ground poly */
        glTranslatef(0, 0.01, 0);
        /* Multiply all y coordinates with zero */
        glScalef(1, 0, 1);
        glColor3f(0, 0, 0);
        drawRobot();
        glPopMatrix();

        /* Move up a bit */
        glTranslatef(0, 2, 0);
        /* Make robot jump */
        glTranslatef(0, hopp, 0);
        glEnable(GL_LIGHTING);
        setMaterial();
        drawRobot();
        glDisable(GL_LIGHTING);

        /* Show the back buffer */
        glutSwapBuffers();
}
```

```
int main(int argc, char *argv[]) {
	/* Initialize GLUT */
	glutInit(&argc, argv);

	/* Tell GLUT we want a depth buffer and a back buffer */
	glutInitDisplayMode(GLUT_RGB | GLUT_DEPTH | GLUT_DOUBLE);

	/* Suggest a window size */
	glutInitWindowSize(500, 500);

	/* Create a window */
	glutCreateWindow("Mitt fina foenster");

	/* Set the redisplay callback */
	glutDisplayFunc(display);

	/* Set the reshape callback */
	glutReshapeFunc(reshape);

	/* Set the keyboard callback */
	glutKeyboardFunc(keyboard);

	/* Set an idle callback for animation */
	glutIdleFunc(idle);

	/* Load the texture */
	createTexture();

	/* Leave control to GLUT */
	glutMainLoop();

	return 0;
}
```