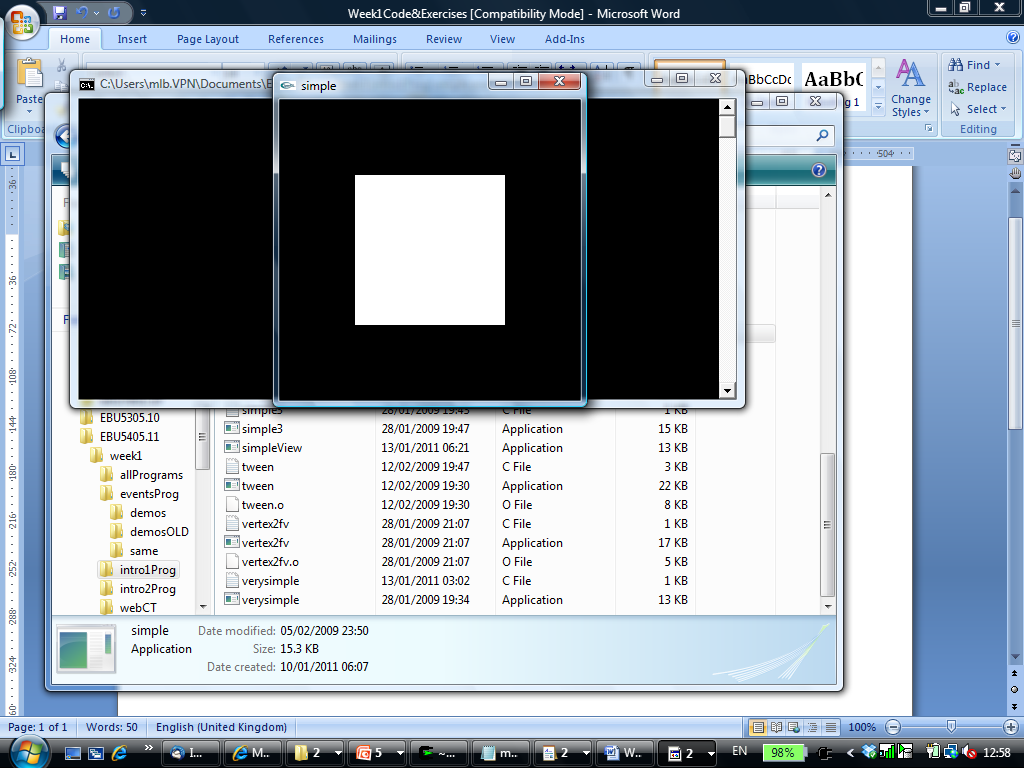
EBU5405 – week 1 – Code & Exercises

1. **simple.c**



#include <GL/glut.h>

void mydisplay(){

glClear(GL\_COLOR\_BUFFER\_BIT);

glBegin(GL\_POLYGON);

glVertex2f(-0.5, -0.5);

glVertex2f(-0.5, 0.5);

glVertex2f(0.5, 0.5);

glVertex2f(0.5, -0.5);

glEnd();

glFlush();

}

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

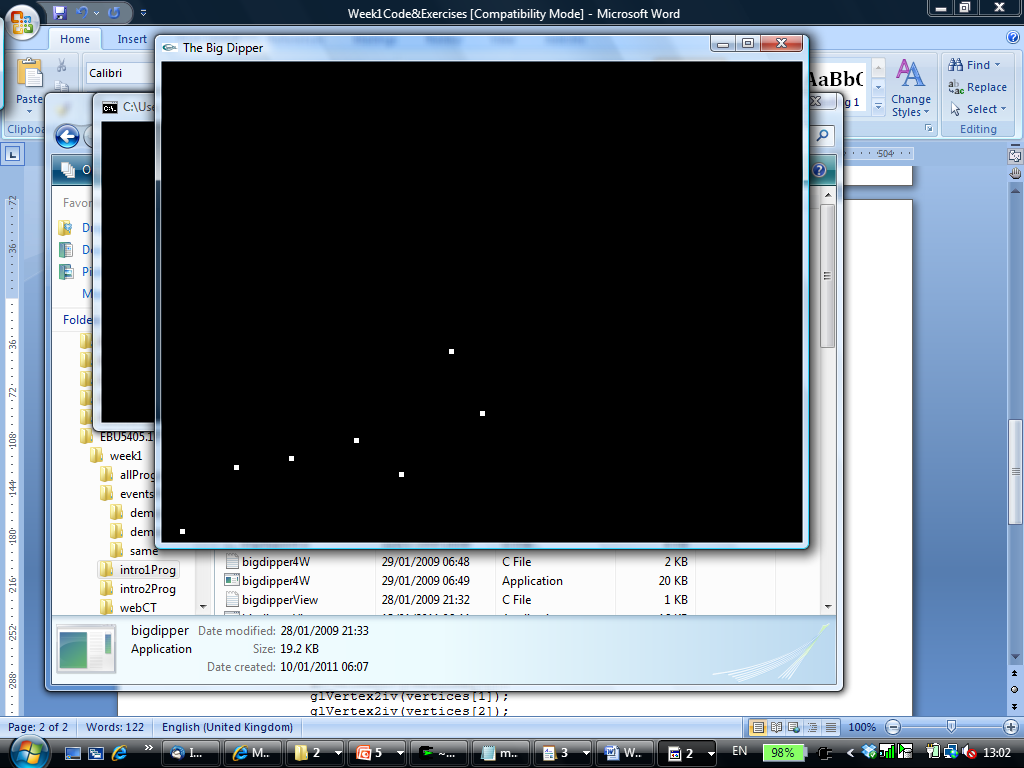
glutCreateWindow("simple");

glutDisplayFunc(mydisplay);

glutMainLoop();

}

1. **bigdipper.c**



#include <GL/glut.h>

void myInit(void) {

glClearColor(0.0, 0.0, 0.0, 0.0);

glColor3f(1.0, 1.0, 1.0);

glPointSize(5.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0.0, 640.0, 0.0, 480.0);

}

void mydisplay(){

GLint vertices[7][2] = {{20, 10}, {74, 74}, {129, 83}, {194, 101},

{239, 67}, {320, 128}, {289, 190}};

glClear(GL\_COLOR\_BUFFER\_BIT);

glBegin(GL\_POINTS);

glVertex2iv(vertices[0]);

glVertex2iv(vertices[1]);

glVertex2iv(vertices[2]);

glVertex2iv(vertices[3]);

glVertex2iv(vertices[4]);

glVertex2iv(vertices[5]);

glVertex2iv(vertices[6]);

glEnd();

glFlush();

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(640, 480);

glutCreateWindow("The Big Dipper");

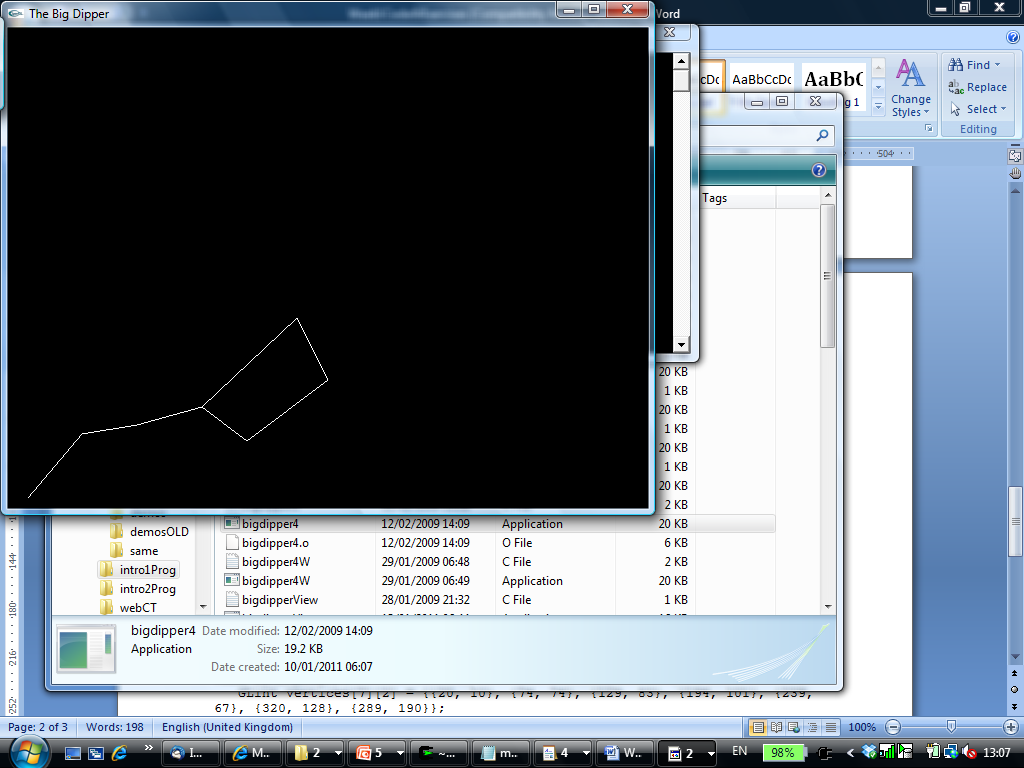
glutDisplayFunc(mydisplay);

myInit();

glutMainLoop();

}

1. **bigdipper2.c**



#include <GL/glut.h>

void myInit(void) {

glClearColor(0.0, 0.0, 0.0, 0.0);

glColor3f(1.0, 1.0, 1.0);

glPointSize(5.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0.0, 640.0, 0.0, 480.0);

}

void mydisplay(){

GLint vertices[7][2] = {{20, 10}, {74, 74}, {129, 83}, {194, 101},

239, 67}, {320, 128}, {289, 190}};

glClear(GL\_COLOR\_BUFFER\_BIT);

glBegin(GL\_LINE\_STRIP);

glVertex2iv(vertices[0]);

glVertex2iv(vertices[1]);

glVertex2iv(vertices[2]);

glVertex2iv(vertices[3]);

glEnd();

glBegin(GL\_LINE\_LOOP);

glVertex2iv(vertices[3]);

glVertex2iv(vertices[4]);

glVertex2iv(vertices[5]);

glVertex2iv(vertices[6]);

glVertex2iv(vertices[3]);

glEnd();

glFlush();

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(640, 480);

glutCreateWindow("The Big Dipper");

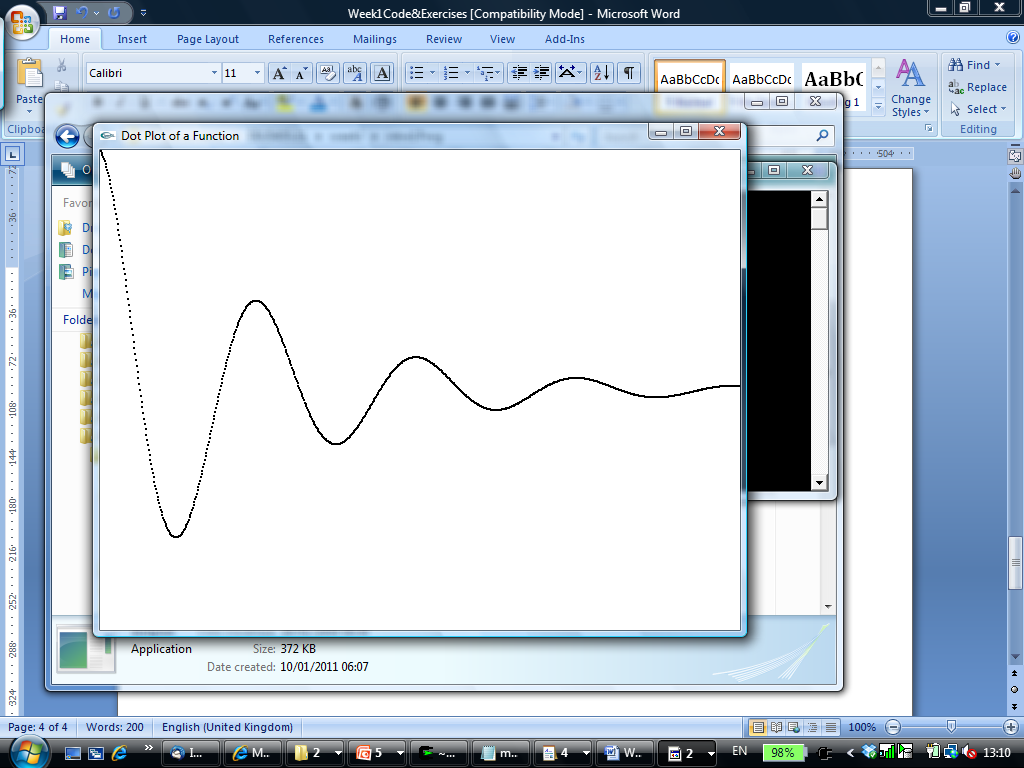
glutDisplayFunc(mydisplay);

myInit();

glutMainLoop();

}

1. **dotplot.c**



#include <math.h>

#include <gl/Glut.h>

const int screenWidth = 640; // width of the screen window in pixels

const int screenHeight = 480; // height of the screen window in pixels

GLdouble A, B, C, D; // values used for scaling and shifting

//<<<<<<<<<<<<<<<<<<<<<<< myInit >>>>>>>>>>>>>>>>>>>>

void myInit(void)

{

glClearColor(1.0,1.0,1.0,0.0); // the background color is white

glColor3f(0.0f, 0.0f, 0.0f); // the drawing color is black

glPointSize(2.0); // a 'dot' is 2 by 2 pixels

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0.0, (GLdouble)screenWidth, 0.0, (GLdouble)screenHeight);

A = screenWidth / 4.0; // sets the values used for scaling and shifting

B = 0.0;

C = D = screenHeight / 2.0;

}

//<<<<<<<<<<<<<<<<<<<<<<<< myDisplay >>>>>>>>>>>>>>>>>

void myDisplay(void)

{

GLdouble x;

glClear(GL\_COLOR\_BUFFER\_BIT); // clear the screen

glBegin(GL\_POINTS);

for(x = 0; x < 4.0 ; x += 0.005)

{

GLdouble func = exp(-x) \* cos(2 \* 3.14159265 \* x);

glVertex2d(A \* x + B, C \* func + D);

}

glEnd();

glFlush(); // send all output to display

}

//<<<<<<<<<<<<<<<<<<<<<<<< main >>>>>>>>>>>>>>>>>>>>>>

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

{

glutInit(&argc, argv); // initialize the toolkit

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB); // set the display mode

glutInitWindowSize(screenWidth, screenHeight); // set the window size

glutInitWindowPosition(100, 150); // set the window position

glutCreateWindow("Dot Plot of a Function"); // open the screen window

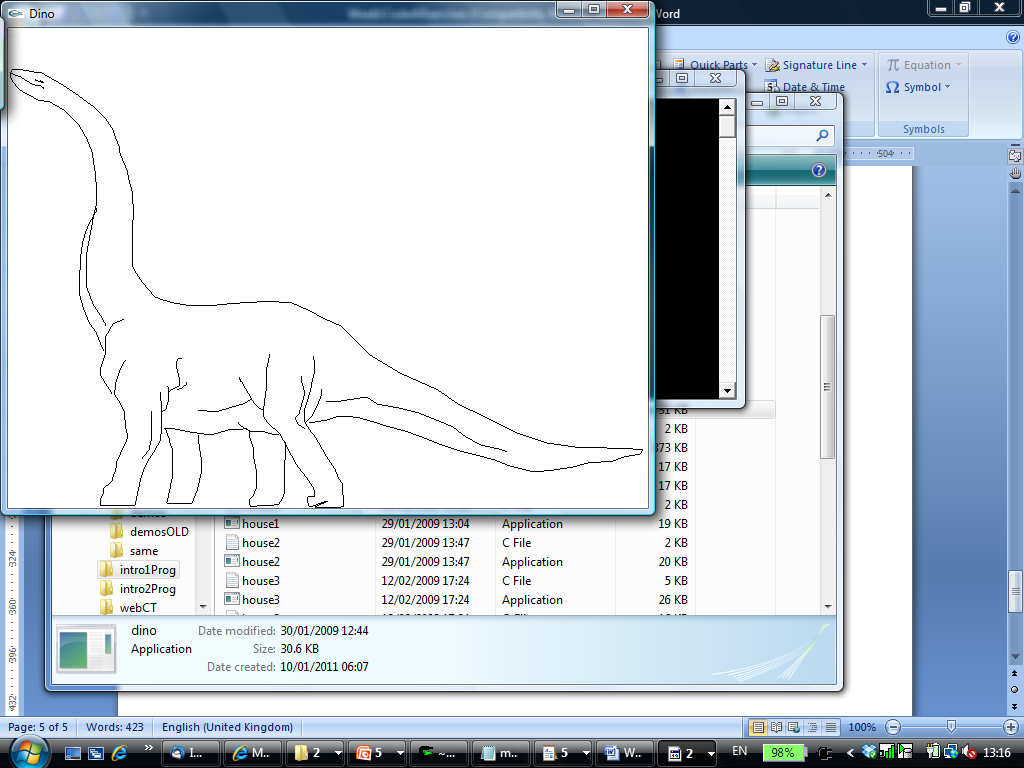
glutDisplayFunc(myDisplay); // register the redraw function

myInit();

glutMainLoop(); // go into a perpetual loop

}

1. **dino.c**



#include <stdio.h>

#include <GL/glut.h>

void myInit(void) {

glClearColor(1.0, 1.0, 1.0, 0.0);

glColor3f(0.0, 0.0, 0.0);

glPointSize(3.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0.0, 640.0, 0.0, 480.0);

}

void drawPolyLineFile(char \*filename) {

FILE \*ifp;

int i, j;

GLint numpolys, numLines, x, y;

ifp = fopen (filename, "r");

if(ifp==NULL) {

printf("Error: can't open file for reading.\n");

return;

}

glClear(GL\_COLOR\_BUFFER\_BIT);

fscanf (ifp, "%d", &numpolys);

for (j=0; j < numpolys; j++) {

fscanf (ifp, "%d", &numLines);

glBegin (GL\_LINE\_STRIP);

for (i = 0; i < numLines; i++) {

fscanf (ifp, "%d %d", &x, &y);

glVertex2i (x, y);

}

glEnd();

}

glFlush();

fclose (ifp);

}

void mydisplay(){

drawPolyLineFile ("dino.dat");

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(640, 480);

glutCreateWindow("Dino");

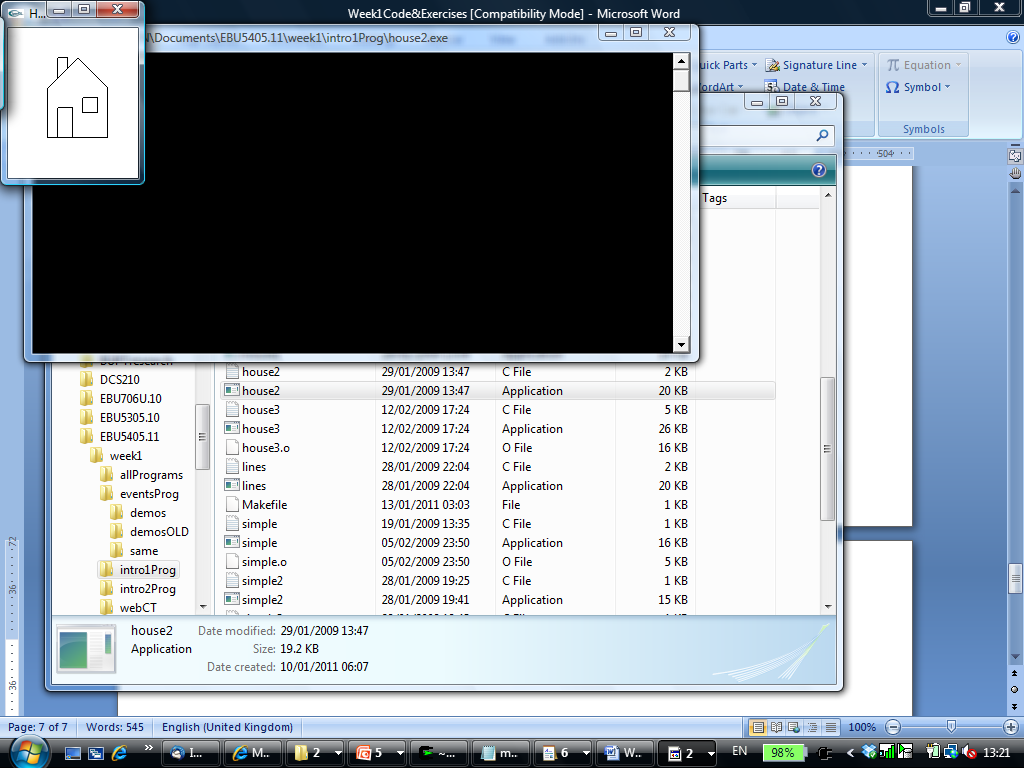
glutDisplayFunc(mydisplay);

myInit();

glutMainLoop();

}

1. **parameterisedHouse.c**



#include <GL/glut.h>

void myInit(void) {

glClearColor(1.0, 1.0, 1.0, 0.0);

glColor3f(0.0, 0.0, 0.0);

glPointSize(5.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0.0, 130.0, 0.0, 150.0);

}

void mydisplay(){

glClear(GL\_COLOR\_BUFFER\_BIT);

int peak[2] = {70, 120};

int width = 60;

int height = 80;

glBegin(GL\_LINE\_LOOP); // house

glVertex2i(peak[0], peak[1]);

glVertex2i(peak[0]+width/2, peak[1]-(3\*height/8));

glVertex2i(peak[0]+width/2, peak[1]-height);

glVertex2i(peak[0]-width/2, peak[1]-height);

glVertex2i(peak[0]-width/2, peak[1]-(3\*height/8));

glEnd();

glBegin (GL\_LINE\_STRIP); // chimney

glVertex2i(peak[0]-(2\*width/6), peak[1]-(2\*height/8));

glVertex2i(peak[0]-(2\*width/6), peak[1]);

glVertex2i(peak[0]-(width/6), peak[1]);

glVertex2i(peak[0]-(width/6), peak[1]-(height/8));

glEnd();

glBegin (GL\_LINE\_STRIP); // door

glVertex2i(peak[0]-(2\*width/6), peak[1]-(height));

glVertex2i(peak[0]-(2\*width/6), peak[1]-(5\*height/8));

glVertex2i(peak[0]-(5\*width/60), peak[1]-(5\*height/8));

glVertex2i(peak[0]-(5\*width/60), peak[1]-(height));

glEnd();

glBegin(GL\_LINE\_LOOP); // window

glVertex2i(peak[0]+(5\*width/60), peak[1]-(55\*height/80));

glVertex2i(peak[0]+(5\*width/60), peak[1]-(4\*height/8));

glVertex2i(peak[0]+(2\*width/6), peak[1]-(4\*height/8));

glVertex2i(peak[0]+(2\*width/6), peak[1]-(55\*height/80));

glEnd();

glFlush();

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(130, 150);

glutCreateWindow("House");

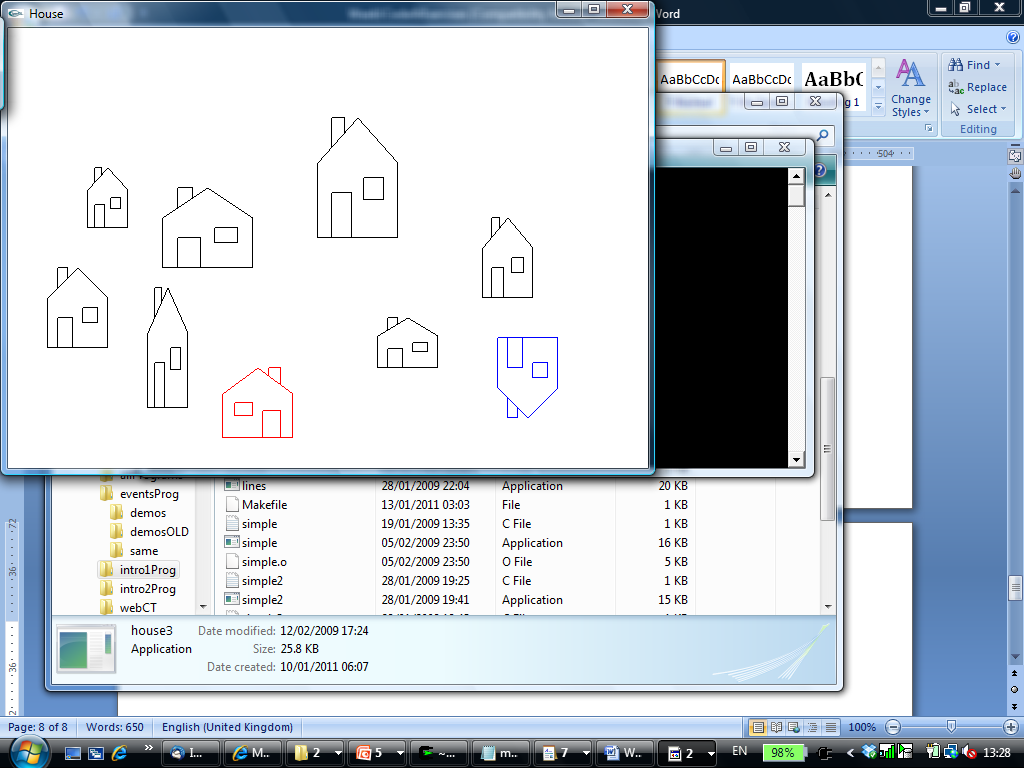
glutDisplayFunc(mydisplay);

myInit();

glutMainLoop();

}

**Exercise 1 (code to be completed):**



#include <GL/glut.h>

GLint peak[2], width, height;

void myInit(void) {

glClearColor(1.0, 1.0, 1.0, 0.0);

glColor3f(0.0, 0.0, 0.0);

glPointSize(5.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0.0, 640.0, 0.0, 440.0);

}

void drawhouse() { ***// same as in the mydisplay function of the parameterisedHouse.c program***

glBegin(GL\_LINE\_LOOP);

glVertex2i(peak[0], peak[1]); // house

glVertex2i(peak[0]+width/2, peak[1]-(3\*height/8));

glVertex2i(peak[0]+width/2, peak[1]-height);

glVertex2i(peak[0]-width/2, peak[1]-height);

glVertex2i(peak[0]-width/2, peak[1]-(3\*height/8));

glEnd();

glBegin (GL\_LINE\_STRIP); // chimney

glVertex2i(peak[0]-(2\*width/6), peak[1]-(2\*height/8));

glVertex2i(peak[0]-(2\*width/6), peak[1]);

glVertex2i(peak[0]-(width/6), peak[1]);

glVertex2i(peak[0]-(width/6), peak[1]-(height/8));

glEnd();

glBegin (GL\_LINE\_STRIP); // door

glVertex2i(peak[0]-(2\*width/6), peak[1]-(height));

glVertex2i(peak[0]-(2\*width/6), peak[1]-(5\*height/8));

glVertex2i(peak[0]-(5\*width/60), peak[1]-(5\*height/8));

glVertex2i(peak[0]-(5\*width/60), peak[1]-(height));

glEnd();

glBegin(GL\_LINE\_LOOP); // window

glVertex2i(peak[0]+(5\*width/60), peak[1]-(55\*height/80));

glVertex2i(peak[0]+(5\*width/60), peak[1]-(4\*height/8));

glVertex2i(peak[0]+(2\*width/6), peak[1]-(4\*height/8));

glVertex2i(peak[0]+(2\*width/6), peak[1]-(55\*height/80));

glEnd();

glFlush();

}

void drawhouse1() { ***// flipped house (the door and the chimney on the right side of the house)***

glColor3f(1.0, 0.0, 0.0);

glBegin(GL\_LINE\_LOOP);

glVertex2i(peak[0], peak[1]);

glVertex2i(peak[0]+width/2, peak[1]-(3\*height/8));

glVertex2i(peak[0]+width/2, peak[1]-height);

glVertex2i(peak[0]-width/2, peak[1]-height);

glVertex2i(peak[0]-width/2, peak[1]-(3\*height/8));

glEnd();

glBegin (GL\_LINE\_STRIP); // chimney (4 lines to be completed)

?

?

?

?

glEnd();

glBegin (GL\_LINE\_STRIP); // door (4 lines to be completed)

?

?

?

?

glEnd();

glBegin(GL\_LINE\_LOOP); // window (4 lines to be completed)

?

?

?

?

glEnd();

glFlush();

}

void drawhouse2() { ***// upside down house***

glColor3f(0.0, 0.0, 1.0);

glBegin(GL\_LINE\_LOOP); // house (5 lines to be completed)

?

?

?

?

?

glEnd();

glBegin (GL\_LINE\_STRIP); // chimney (4 lines to be completed)

?

?

?

?

glEnd();

glBegin (GL\_LINE\_STRIP); // door (4 lines to be completed)

?

?

?

?

glEnd();

glBegin(GL\_LINE\_LOOP); // window (4 lines to be completed)

?

?

?

?

glEnd();

glFlush();

}

void mydisplay(){

glClear(GL\_COLOR\_BUFFER\_BIT);

peak[0] = 70;

peak[1] = 200;

width = 60;

height = 80;

drawhouse ();

peak[0] = 100;

peak[1] = 300;

width = 40;

height = 60;

drawhouse ();

peak[0] = 200;

peak[1] = 280;

width = 90;

height = 80;

drawhouse ();

peak[0] = 160;

peak[1] = 180;

width = 40;

height = 120;

drawhouse ();

peak[0] = 350;

peak[1] = 350;

width = 80;

height = 120;

drawhouse ();

peak[0] = 400;

peak[1] = 150;

width = 60;

height = 50;

drawhouse ();

peak[0] = 500;

peak[1] = 250;

width = 50;

height = 80;

drawhouse ();

peak[0] = 250;

peak[1] = 100;

width = 70;

height = 70;

drawhouse1 ();

peak[0] = 520;

peak[1] = 50;

width = 60;

height = 80;

drawhouse2 ();

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(640, 440);

glutCreateWindow("House");

glutDisplayFunc(mydisplay);

myInit();

glutMainLoop();

}

1. **simpleRevisited.c**

#include <GL/glut.h>

void myinit()

{

glClearColor (0.0, 0.0, 0.0, 1.0);

glColor3f(1.0, 1.0, 1.0);

glMatrixMode (GL\_PROJECTION);

glLoadIdentity ();

glOrtho(-1.0, 1.0, -1.0, 1.0, -1.0, 1.0);

}

void mydisplay(){

glClear(GL\_COLOR\_BUFFER\_BIT);

glBegin(GL\_POLYGON);

glVertex2f(-0.5, -0.5);

glVertex2f(-0.5, 0.5);

glVertex2f(0.5, 0.5);

glVertex2f(0.5, -0.5);

glEnd();

glFlush();

}

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

{

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

glutInitWindowSize(500,500);

glutInitWindowPosition(0,0);

glutCreateWindow("simple");

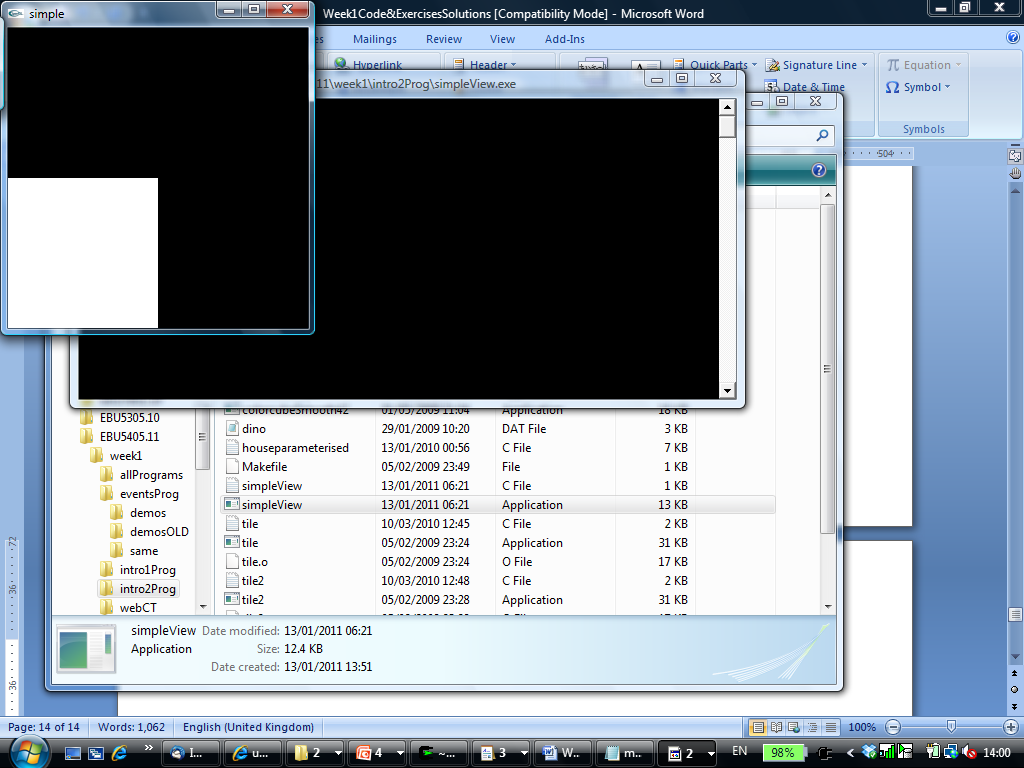
glutDisplayFunc(mydisplay);

myinit();

glutMainLoop();

}

1. **simpleView.c**



#include <GL/glut.h>

void mydisplay(){

glClear(GL\_COLOR\_BUFFER\_BIT);

glMatrixMode (GL\_PROJECTION);

glLoadIdentity ();

glOrtho(-0.5, 1.5, -0.5, 1.5, -1.0, 1.0);

glBegin(GL\_POLYGON);

glVertex2f(-0.5, -0.5);

glVertex2f(-0.5, 0.5);

glVertex2f(0.5, 0.5);

glVertex2f(0.5, -0.5);

glEnd();

glFlush();

}

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

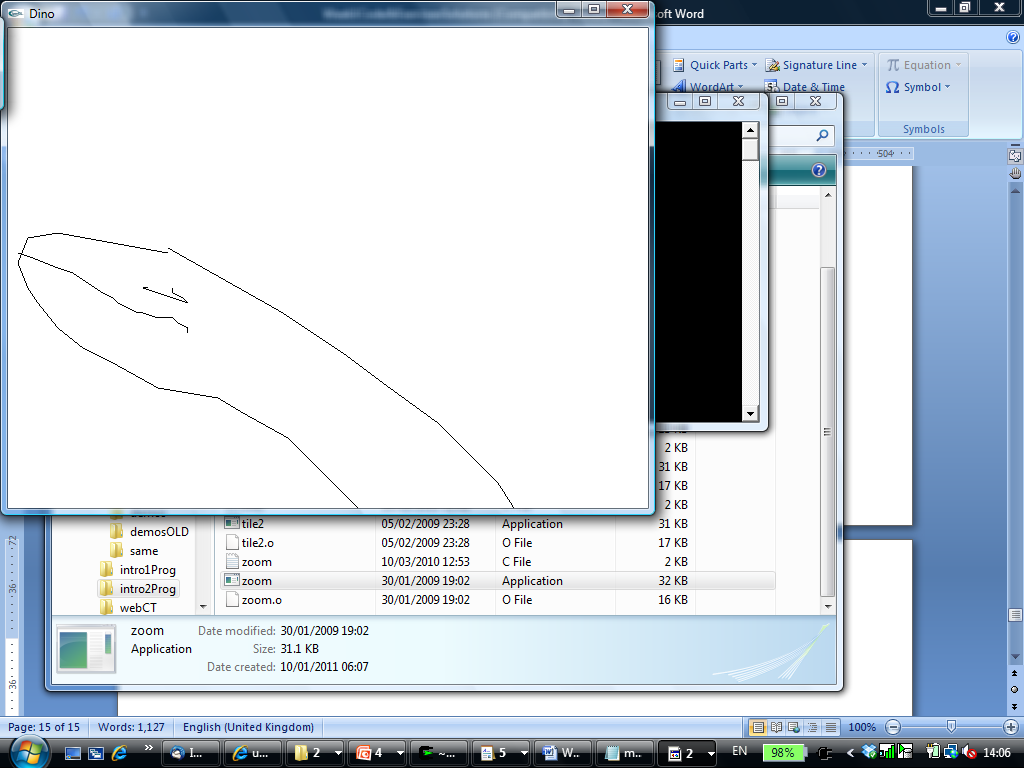
glutCreateWindow("simple");

glutDisplayFunc(mydisplay);

glutMainLoop();

}

1. **dinoZoomed.c**



#include <stdio.h>

#include <GL/glut.h>

GLdouble W = 640.0;

GLdouble H = 480.0;

void myInit(void) {

glClearColor(1.0, 1.0, 1.0, 0.0);

glColor3f(0.0, 0.0, 0.0);

glPointSize(3.0);

}

void setWindow (GLdouble left, GLdouble right, GLdouble bottom, GLdouble top) {

glMatrixMode (GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D (left, right, bottom, top);

}

void drawPolyLineFile(char \*filename) {

………… // same as for dino.c

}

void mydisplay(){

setWindow (0, W/5, 4\*H/5, H);

drawPolyLineFile ("dino.dat");

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(W, H);

glutCreateWindow("Dino");

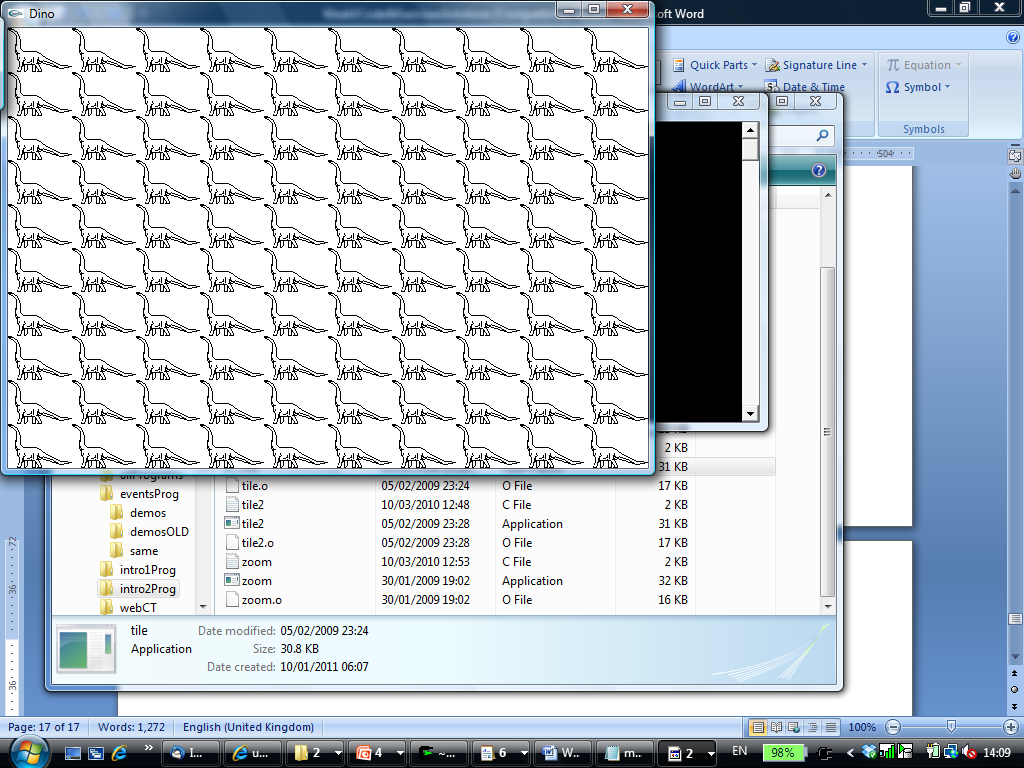
glutDisplayFunc(mydisplay);

myInit();

glutMainLoop();

}

1. **dinoTiled.c**



#include <stdio.h>

#include <GL/glut.h>

void myInit(void) {

glClearColor(1.0, 1.0, 1.0, 0.0);

glColor3f(0.0, 0.0, 0.0);

glPointSize(3.0);

}

void setWindow (GLdouble left, GLdouble right, GLdouble bottom, GLdouble top) {

glMatrixMode (GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D (left, right, bottom, top);

}

void drawPolyLineFile(char \*filename) {

………… // same as for dino.c

}

void mydisplay(){

int k, l;

setWindow (0, 640.0, 0, 440.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

for (k=0; k < 10; k++) {

for (l=0; l < 10; l++) {

glViewport (k\*64, l\*44, 64, 44);

drawPolyLineFile ("dino.dat");

}

}

glFlush();

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(640, 440);

glutCreateWindow("Dino");

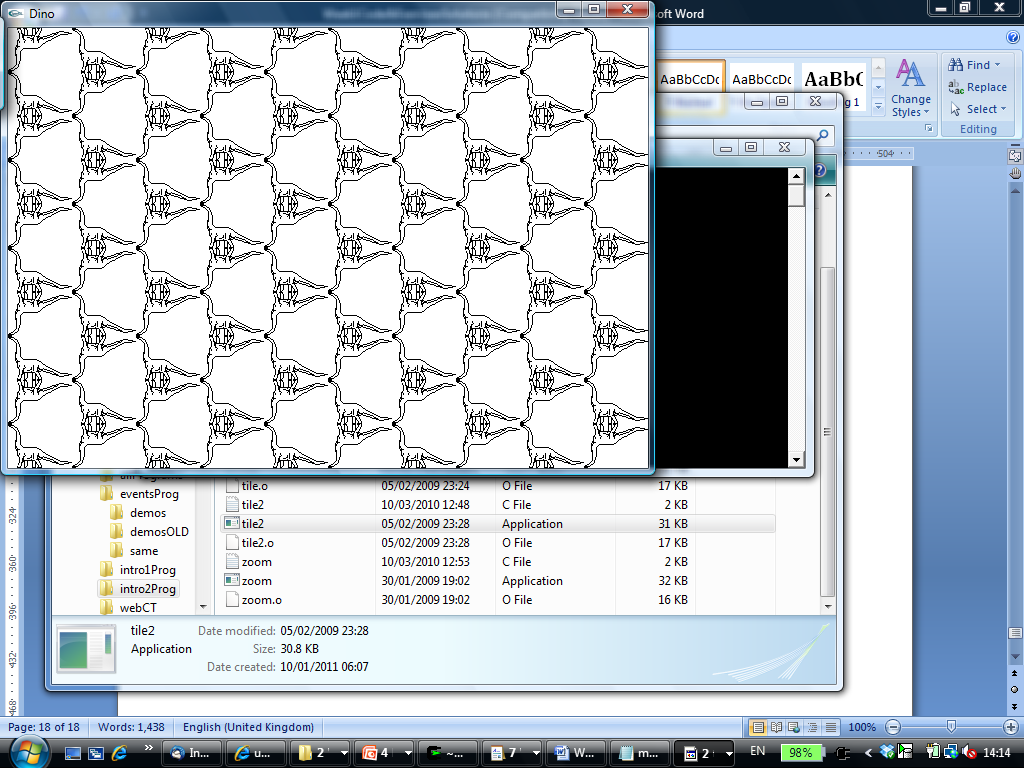
glutDisplayFunc(mydisplay);

myInit();

glutMainLoop();

}

**Exercise 2 (code to be completed):**



#include <stdio.h>

#include <GL/glut.h>

void myInit(void) {

glClearColor(1.0, 1.0, 1.0, 0.0);

glColor3f(0.0, 0.0, 0.0);

glPointSize(3.0);

}

void setWindow (GLdouble left, GLdouble right, GLdouble bottom, GLdouble top) {

glMatrixMode (GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D (left, right, bottom, top);

}

void drawPolyLineFile(char \*filename) {

………… // same as for dino.c

}

void mydisplay(){

int k, l;

glClear(GL\_COLOR\_BUFFER\_BIT);

for (k=0; k < 10; k++) {

for (l=0; l < 10; l++) { // 6 lines to be completed

?

?

?

?

?

?

}

}

glFlush();

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(640, 440);

glutCreateWindow("Dino");

glutDisplayFunc(mydisplay);

myInit();

glutMainLoop();

}

1. **rotatingSquares.c**

#include <stdlib.h>

#include <GL/glut.h>

#include <math.h>

#define DEGREES\_TO\_RADIANS 3.14159/180.0

static GLfloat spin = 0.0;

GLfloat a, b;

void square()

{

glBegin(GL\_QUADS);

glVertex2f(a , b);

glVertex2f(-b , a);

glVertex2f(-a , -b);

glVertex2f(b , -a);

glEnd();

}

void displayd()

{

glClear (GL\_COLOR\_BUFFER\_BIT);

square();

glutSwapBuffers ();

}

void spinDisplay (void)

{

spin = spin + 5.0;

if (spin > 360.0) spin = spin - 360.0;

a = 0.5 \* cos(DEGREES\_TO\_RADIANS \* spin);

b = 0.5 \* sin(DEGREES\_TO\_RADIANS \* spin);

glutPostRedisplay();

}

void myinit ()

{

glClearColor (0.0, 0.0, 0.0, 1.0);

glColor3f (1.0, 1.0, 1.0);

glShadeModel (GL\_FLAT);

}

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

{

glutInit(&argc,argv);

glutInitDisplayMode (GLUT\_DOUBLE | GLUT\_RGB);

glutInitWindowPosition(500,0);

glutCreateWindow("double buffered");

myinit ();

glutDisplayFunc(displayd);

glutIdleFunc (spinDisplay);

glutMainLoop();

}

1. **dinoZoomAnimated.c**

#include <stdio.h>

#include <GL/glut.h>

GLboolean animated = GL\_FALSE;

GLdouble W = 640.0;

GLdouble H = 480.0;

GLdouble zoom = 1.0;

void myInit(void) {

glClearColor(1.0, 1.0, 1.0, 0.0);

glColor3f(0.0, 0.0, 0.0);

glPointSize(3.0);

}

void setWindow (GLdouble left, GLdouble right, GLdouble bottom, GLdouble top) {

glMatrixMode (GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D (left, right, bottom, top);

}

void myidle() {

if (!animated) return;

if (zoom > 0.2) zoom -= 0.0005;

glutPostRedisplay();

}

void drawPolyLineFile(char \*filename) {

………… // same as for dino.c

}

void mydisplay(){

glClear(GL\_COLOR\_BUFFER\_BIT);

setWindow (0, W\*zoom, H\*(1-zoom), H);

drawPolyLineFile ("dino.dat");

glutSwapBuffers();

}

void mymouse(GLint button, GLint state, GLint x, GLint y) {

if (button==GLUT\_LEFT\_BUTTON && state==GLUT\_DOWN)

animated = !animated;

if (button==GLUT\_RIGHT\_BUTTON && state==GLUT\_DOWN)

exit(0);

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);

glutInitWindowSize(W, H);

glutCreateWindow("Dino");

glutDisplayFunc(mydisplay);

glutIdleFunc(myidle);

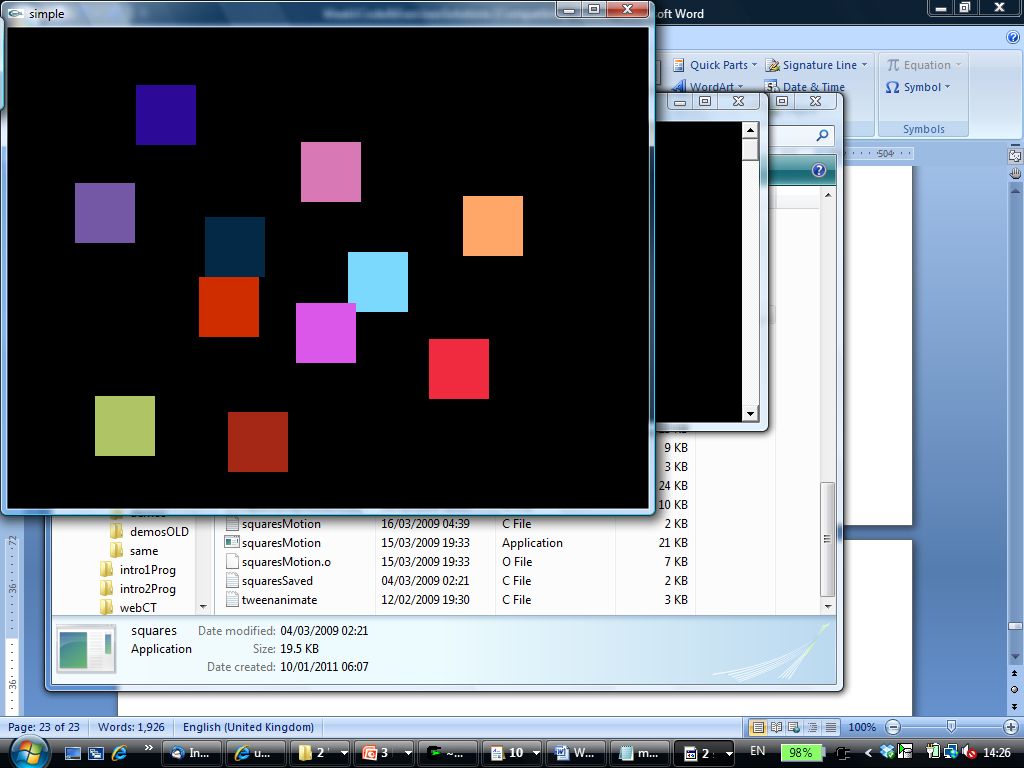
glutMouseFunc(mymouse);

myInit();

glutMainLoop();

}

1. **squaresMouse.c**



#include <GL/glut.h>

GLdouble W = 640.0;

GLdouble H = 480.0;

GLint size = 30;

void myInit(void) {

glMatrixMode (GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D (0, W, 0, H);

}

void mydisplay(){

}

void drawSquare(int x, int y)

{

GLint viewport[4];

glGetIntegerv (GL\_VIEWPORT, viewport);

y = viewport[3] - y; /\* invert y position \*/

glColor3ub((char)rand()%256, (char)rand()%256, (char)rand()%256);

glBegin(GL\_POLYGON);

glVertex2i(x+size, y+size);

glVertex2i(x-size, y+size);

glVertex2i(x-size, y-size);

glVertex2i(x+size, y-size);

glEnd();

glFlush();

}

void mymouse(int btn, int state, int x, int y)

{

if(btn==GLUT\_LEFT\_BUTTON && state==GLUT\_DOWN)

drawSquare(x, y);

}

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

glutInit(&argc, argv);

glutInitWindowSize(W, H);

glutCreateWindow("simple");

glutDisplayFunc(mydisplay);

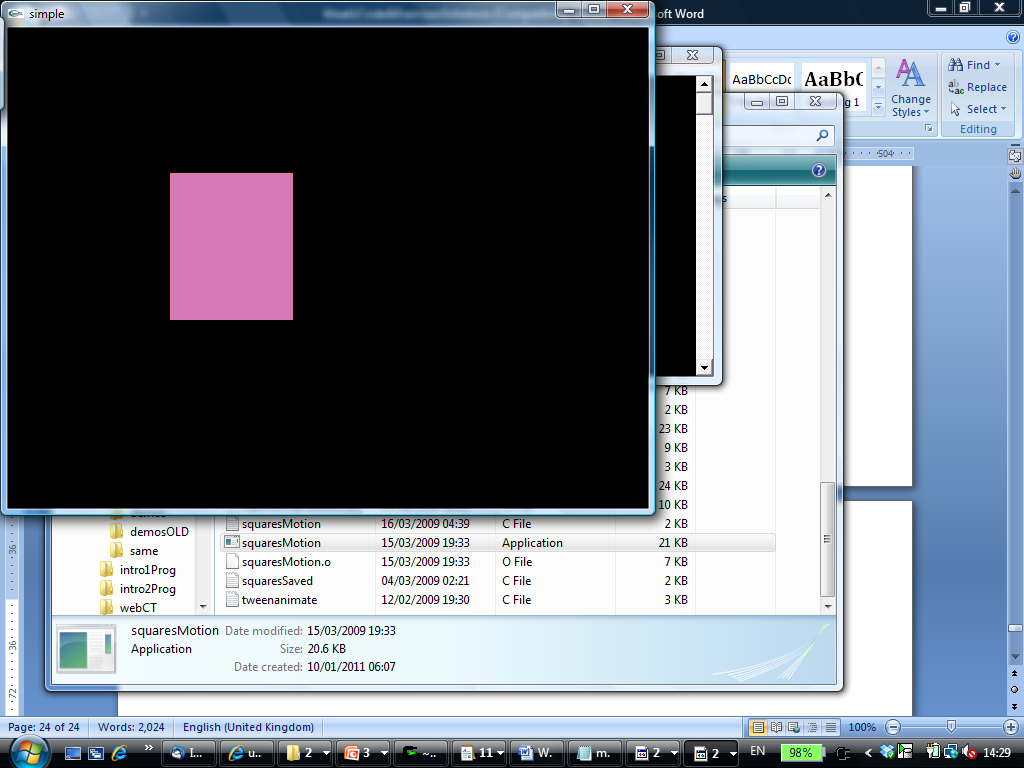
glutMouseFunc(mymouse);

myInit();

glutMainLoop();

}

1. **squareMotion.c**



#include <GL/glut.h>

GLdouble W = 640.0;

GLdouble H = 480.0;

GLint xx = 0;

GLint yy = 0;

void myInit(void) {

glMatrixMode (GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D (0, W, 0, H);

}

void mydisplay(){

}

void mymouse(int btn, int state, int x, int y)

{

GLint viewport[4];

glGetIntegerv (GL\_VIEWPORT, viewport);

if(btn==GLUT\_LEFT\_BUTTON && state==GLUT\_DOWN) {

glColor3ub((char)(rand()\*1000)%256, (char)(rand()\*1000)%256, (char)(rand()\*1000)%256); /\* a random color \*/

xx = x;

yy = viewport[3] - y;

}

if(btn==GLUT\_RIGHT\_BUTTON && state==GLUT\_DOWN)

exit(0);

}

void drawSquare(int x, int y) {

GLint viewport[4];

glGetIntegerv (GL\_VIEWPORT, viewport);

y = viewport[3] - y; /\* invert y position \*/

glClear(GL\_COLOR\_BUFFER\_BIT);

glBegin(GL\_POLYGON);

glVertex2i(xx, yy);

glVertex2i(x, yy);

glVertex2i(x, y);

glVertex2i(xx, y);

glEnd();

glFlush();

}

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

glutInit(&argc, argv);

glutInitWindowSize(W, H);

glutCreateWindow("simple");

glutDisplayFunc(mydisplay);

glutMouseFunc(mymouse);

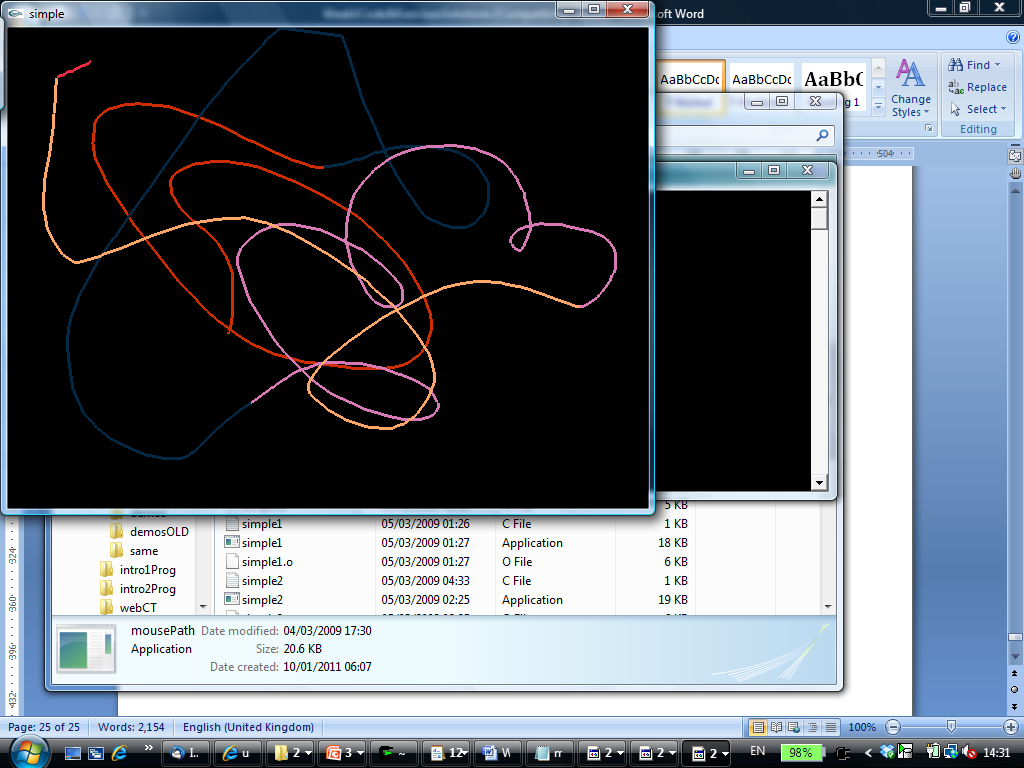
glutMotionFunc(drawSquare);

myInit();

glutMainLoop();

}

**Exercise 3 (code to be completed):**



#include <GL/glut.h>

GLdouble W = 640.0;

GLdouble H = 480.0;

GLint xx = 0;

GLint yy = 0;

GLboolean started = GL\_FALSE;

void myInit(void) {

glMatrixMode (GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D (0, W, 0, H);

glLineWidth (3.0);

}

void mydisplay(){

}

void mymouse(int btn, int state, int x, int y)

{

GLint viewport[4];

glGetIntegerv (GL\_VIEWPORT, viewport);

if(btn==GLUT\_LEFT\_BUTTON && state==GLUT\_DOWN) {

glColor3ub((char)rand()%256, (char)rand()%256, (char)rand()%256);

xx = x;

yy = viewport[3] - y; /\* invert y position \*/

started = GL\_TRUE;

}

if(btn==GLUT\_RIGHT\_BUTTON && state==GLUT\_DOWN)

exit(0);

}

void drawPath(int x, int y) {

if (!started) return;

GLint viewport[4];

glGetIntegerv (GL\_VIEWPORT, viewport); // 8 lines to be completed

?

?

?

?

?

?

?

?

}

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

glutInit(&argc, argv);

glutInitWindowSize(W, H);

glutCreateWindow("simple");

glutDisplayFunc(mydisplay);

glutMouseFunc(mymouse);

glutPassiveMotionFunc(drawPath);

myInit();

glutMainLoop();

}

1. **squareKeyMoving.c**

#include <GL/glut.h>

GLint size = 30;

GLfloat moveX = 0.0;

GLfloat moveY = 0.0;

GLfloat xx = 0;

GLfloat yy = 0;

void myInit(void) {

glMatrixMode (GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D (0, 640, 0, 480);

}

void mydisplay(){

}

void drawSquare(int x, int y)

{

GLint viewport[4];

glGetIntegerv (GL\_VIEWPORT, viewport);

y = viewport[3] - y; /\* invert y position \*/

xx = x;

yy = y;

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3ub((char)rand()%256, (char)rand()%256, (char)rand()%256);

glBegin(GL\_POLYGON);

glVertex2f(xx+size, yy+size);

glVertex2f(xx-size, yy+size);

glVertex2f(xx-size, yy-size);

glVertex2f(xx+size, yy-size);

glEnd();

glFlush();

}

void mymouse(int btn, int state, int x, int y)

{

if(btn==GLUT\_LEFT\_BUTTON && state==GLUT\_DOWN)

drawSquare(x, y);

}

void myspecialkey(int key, int x, int y) {

switch (key) {

case GLUT\_KEY\_LEFT:

moveX = -0.1;

break;

case GLUT\_KEY\_RIGHT:

moveX = 0.1;

break;

case GLUT\_KEY\_DOWN:

moveY = -0.1;

break;

case GLUT\_KEY\_UP:

moveY = 0.1;

break;

}

}

void myspecialkeyup(int key, int x, int y) {

if (key == GLUT\_KEY\_LEFT || key == GLUT\_KEY\_RIGHT)

moveX = 0.0;

if (key == GLUT\_KEY\_UP || key == GLUT\_KEY\_DOWN)

moveY = 0.0;

}

void myidle() {

if (moveX == 0 && moveY == 0) return;

glClear(GL\_COLOR\_BUFFER\_BIT);

xx = xx + moveX;

yy = yy + moveY;

glBegin(GL\_POLYGON);

glVertex2f(xx+size, yy+size);

glVertex2f(xx-size, yy+size);

glVertex2f(xx-size, yy-size);

glVertex2f(xx+size, yy-size);

glEnd();

glFlush();

}

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

glutInit(&argc, argv);

glutInitWindowSize(640, 480);

glutCreateWindow("squaresKeyMotionComplete");

glutDisplayFunc(mydisplay);

glutMouseFunc(mymouse);

glutSpecialFunc(myspecialkey);

glutSpecialUpFunc(myspecialkeyup);

glutIdleFunc(myidle);

myInit();

glutMainLoop();

}

1. **dinoReshape.c**

#include <stdio.h>

#include <GL/glut.h>

void myInit(void) {

glClearColor(1.0, 1.0, 1.0, 0.0);

glColor3f(0.0, 0.0, 0.0);

glPointSize(3.0);

}

void drawPolyLineFile(char \*filename) {

………… // same as for dino.c

}

void mydisplay(){

drawPolyLineFile ("dino.dat");

}

void myreshape(int w, int h)

{

glViewport(0, 0, w, h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

if (w <= h)

gluOrtho2D(0.0, 640.0, 0.0, 480.0 \* (GLfloat) h / (GLfloat) w);

else gluOrtho2D(0.0, 640.0 \*

(GLfloat) w / (GLfloat) h, 0.0, 480.0);

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(640, 480);

glutCreateWindow("dinoReshape");

glutDisplayFunc(mydisplay);

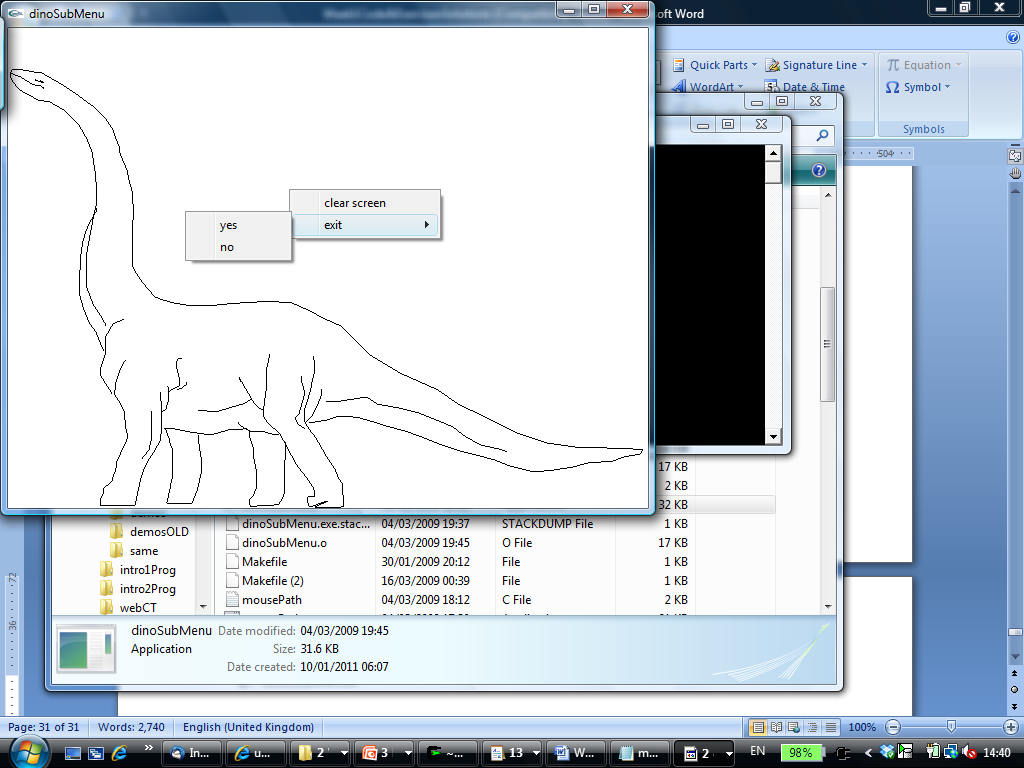
glutReshapeFunc(myreshape);

myInit();

glutMainLoop();

}

1. **dinoSubMenus.c**



#include <stdio.h>

#include <GL/glut.h>

GLint menu\_id = 0;

GLint submenu\_id = 0;

void myInit(void) {

glClearColor(1.0, 1.0, 1.0, 0.0);

glColor3f(0.0, 0.0, 0.0);

glPointSize(3.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0.0, 640.0, 0.0, 480.0);

}

void drawPolyLineFile(char \*filename) {

………… // same as for dino.c

}

void mydisplay(){

drawPolyLineFile ("dino.dat");

}

void mymenu(int id)

{

if(id == 1) {

glClear(GL\_COLOR\_BUFFER\_BIT);

glFlush();

}

}

void mysubmenu(int id)

{

if(id == 1) exit(0);

if(id == 2) return;

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(640, 480);

glutCreateWindow("dinoSubMenu");

glutDisplayFunc(mydisplay);

myInit();

submenu\_id = glutCreateMenu(mysubmenu);

glutAddMenuEntry("yes", 1);

glutAddMenuEntry("no", 2);

glutAttachMenu(GLUT\_RIGHT\_BUTTON);

menu\_id = glutCreateMenu(mymenu);

glutAddMenuEntry("clear screen", 1);

glutAddSubMenu ("exit", submenu\_id);

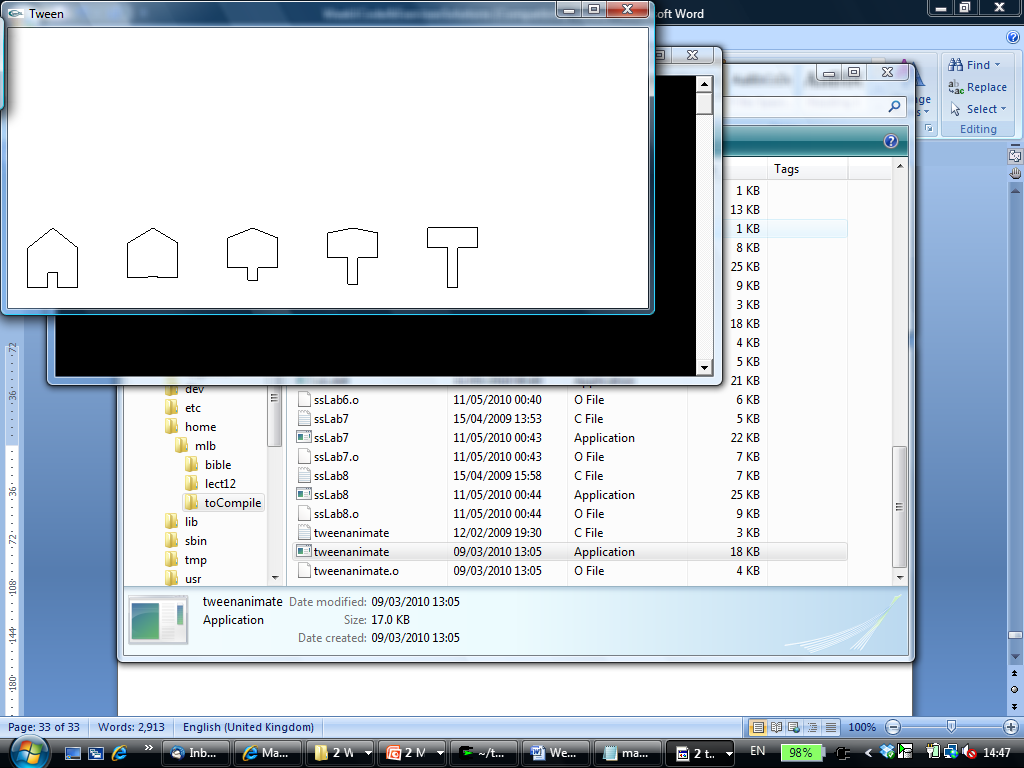
glutAttachMenu(GLUT\_RIGHT\_BUTTON);

glutMainLoop();

}

1. **tweening.c**

**Exercise 4 (code to be completed):**



#include <GL/glut.h>

void myInit(void) {

glClearColor(1.0, 1.0, 1.0, 0.0);

glColor3f(0.0, 0.0, 0.0);

glPointSize(3.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0.0, 640.0, 0.0, 280.0);

}

void mydisplay(){

glClear(GL\_COLOR\_BUFFER\_BIT);

int i,k;

float j;

int offset = 100;

GLfloat A[9][2] = {{20.0, 20.0}, {20.0, 60.0}, {45.0, 80.0},

{70.0, 60.0}, {70.0, 20.0}, {50.0, 20.0},

{50.0, 35.0}, {40.0, 35.0}, {40.0, 20.0}};

GLfloat B[9][2] = {{20.0, 60.0}, {20.0, 80.0}, {45.0, 80.0},

{70.0, 80.0}, {70.0, 60.0}, {50.0, 60.0},

{50.0, 20.0}, {40.0, 20.0}, {40.0, 60.0}};

glBegin(GL\_LINE\_LOOP);

for(k=0;k<9;k++){

glVertex2f(A[k][0], A[k][1]);

}

glEnd();

glFlush();

for (i=0; i<5; i++) {

j = i \* 0.25;

glBegin(GL\_LINE\_LOOP); // 9 lines to be completed

for(k=0;k<9;k++){

glVertex2f(A[k][0]\*(1-j)+B[k][0]\*j+100\*i, A[k][1]\*(1-j)+B[k][1]\*j);

}

glEnd();

}

glFlush();

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(640, 280);

glutCreateWindow("Tween");

glutDisplayFunc(mydisplay);

myInit();

glutMainLoop();

}