HshThe reshape logic to avoid distortion is not applied to any of the question. Other than that all the provided solutions work fine.

1).

#include <GL/glut.h>

/\* initial triangle \*/

GLfloat v[3][2] = { { -1.0, -0.58 },

{ 1.0, -0.58 },{ 0.0, 1.15 } };

int n;

void triangle(GLfloat \*a, GLfloat \*b, GLfloat \*c)

{

glVertex2fv(a);

glVertex2fv(b);

glVertex2fv(c);

}

void divide\_triangle(GLfloat \*a, GLfloat \*b, GLfloat \*c, int m)

{

/\* triangle subdivision using vertex numbers \*/

typedef GLfloat point2[2];

point2 v0, v1, v2;

int j;

if (m>0)

{

for (j = 0; j<2; j++) v0[j] = (a[j] + b[j]) / 2;

for (j = 0; j<2; j++) v1[j] = (a[j] + c[j]) / 2;

for (j = 0; j<2; j++) v2[j] = (b[j] + c[j]) / 2;

divide\_triangle(a, v0, v1, m - 1);

divide\_triangle(c, v1, v2, m - 1);

divide\_triangle(b, v2, v0, m - 1);

}

else(triangle(a, b, c));

/\* draw triangle at end of recursion \*/

}

void display()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glBegin(GL\_TRIANGLES);

glColor3f(0.0, 0.0, 1.0);

divide\_triangle(v[0], v[1], v[2], n);

glEnd();

glFlush();

}

void myinit()

{

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(-2.0, 2.0, -2.0, 2.0);

glMatrixMode(GL\_MODELVIEW);

glClearColor(1.0, 1.0, 1.0, 1.0);

}

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

{

n = 4;

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(500, 500);

glutCreateWindow("2D Gasket");

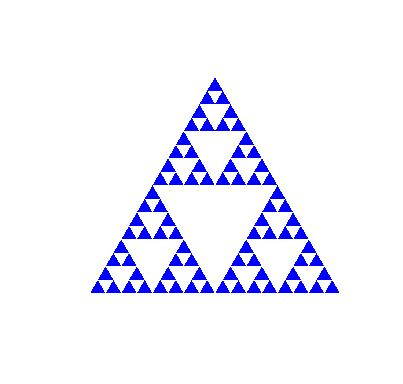
glutDisplayFunc(display);

myinit();

glutMainLoop();

return 0;

}



--------------------------------------------------------------------------------------------------------------------------

2).

#include <GL/glut.h>

/\* initial triangle \*/

GLfloat v[4][3] = { { -1.0, -1.0, 1.0 },{ 1.0, -1.0, 1.0 },{ 0.0, 0.732, 0.0 },{ 0.0, -0.4226,0.0 } };

int n; /\* number of recursive steps \*/

void triangle(GLfloat \*a, GLfloat \*b, GLfloat \*c)

/\* display one triangle \*/

{

glVertex3fv(a);

glVertex3fv(b);

glVertex3fv(c);

}

void divide\_triangle(GLfloat \*a, GLfloat \*b, GLfloat \*c, int m)

{

/\* triangle subdivision using vertex numbers \*/

GLfloat v0[3], v1[3], v2[3];

int j;

if (m>0)

{

for (j = 0; j<3; j++) v0[j] = (a[j] + b[j]) / 2;

for (j = 0; j<3; j++) v1[j] = (a[j] + c[j]) / 2;

for (j = 0; j<3; j++) v2[j] = (b[j] + c[j]) / 2;

divide\_triangle(a, v0, v1, m - 1);

divide\_triangle(c, v1, v2, m - 1);

divide\_triangle(b, v2, v0, m - 1);

}

else(triangle(a, b, c));

/\* draw triangle at end of recursion \*/

}

void tetrahedron(int m)

{

glColor3f(0.0, 1.0, 0.0);

divide\_triangle(v[0], v[1], v[2], m);

glColor3f(1.0, 0.0, 0.0);

divide\_triangle(v[3], v[2], v[1], m);

glColor3f(0.0, 0.0, 0.0);

divide\_triangle(v[0], v[3], v[1], m);

glColor3f(0.0, 0.0, 1.0);

divide\_triangle(v[0], v[2], v[3], m);

}

void display()

{

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glBegin(GL\_TRIANGLES);

tetrahedron(n);

//divide\_triangle(v[0], v[1], v[2], n);

glEnd();

glFlush();

}

void myinit()

{

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(-2.0, 2.0, -2.0, 2.0);

glMatrixMode(GL\_MODELVIEW);

glClearColor(1.0, 1.0, 1.0, 1.0);

glColor3f(0.0, 0.0, 0.0);

glEnable(GL\_DEPTH\_TEST);

glRotated(180, 0, 1, 0);

}

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

{

n = 4;

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB | GLUT\_DEPTH);

glutInitWindowSize(500, 500);

glutCreateWindow("3D Gasket");

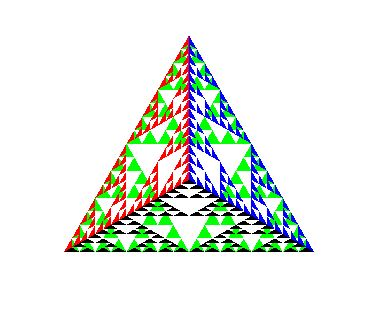
glutDisplayFunc(display);

myinit();

glutMainLoop();

return 0;

}



-------------------------------------------------------------------------------------------------------------------------------

3).

#include <stdlib.h>

#include <GL/glut.h>

GLsizei wh = 500, ww = 500;

GLfloat size = 3.0;

void drawSquare(int x, int y)

{

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

glBegin(GL\_POLYGON);

glVertex2f(x + size, y + size);

glVertex2f(x - size, y + size);

glVertex2f(x - size, y - size);

glVertex2f(x + size, y - size);

glEnd();

glFlush();

}

void myReshape(GLsizei w, GLsizei h)

{

glClearColor(1.0, 1.0, 1.0, 1.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

ww = w;

wh = h;

}

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

{

y = wh - y;

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

drawSquare(x, y);

}

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

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

glFlush();

}

}

void myinit(void)

{

glViewport(0, 0, ww, wh);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(0.0, (GLdouble)ww, 0.0, (GLdouble)wh, -1.0, 1.0);

glFlush();

}

void display(void)

{

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutCreateWindow("Mouse Events - Random Points");

myinit();

glutReshapeFunc(myReshape);

glutMouseFunc(mymouse);

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

---------------------------------------------------------------------------------------------------------------------------

4).

#include <stdlib.h>

#include <GL/glut.h>

GLsizei wh = 500, ww = 500;

int states = 0;

int a, b;

void drawRect(int x, int y)

{

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

glBegin(GL\_POLYGON);

glVertex2f(a, b);

glVertex2f(a, y);

glVertex2f(x, y);

glVertex2f(x, b);

glEnd();

glFlush();

}

void myReshape(GLsizei w, GLsizei h)

{

glClearColor(1.0, 1.0, 1.0, 1.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

ww = w;

wh = h;

}

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

{

y = wh - y;

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

if (states == 0) {

states = 1;

a = x;

b = y;

}

else {

states = 0;

drawRect(x, y);

}

}

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

states = 0;

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

glFlush();

}

}

void myinit(void)

{

glViewport(0, 0, ww, wh);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(0.0, (GLdouble)ww, 0.0, (GLdouble)wh, -1.0, 1.0);

glFlush();

}

void display(void)

{

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutCreateWindow("Mouse Events - Rectangle");

myinit();

glutReshapeFunc(myReshape);

glutMouseFunc(mymouse);

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

-------------------------------------------------------------------------------------------------------------------------------

5).

#include <stdlib.h>

#include <GL/glut.h>

GLsizei wh = 500, ww = 500;

int states = 0;

int a, b;

void Polyline(int x, int y)

{

glColor3f(0.0, 0.0, 0.0);

glBegin(GL\_LINES);

glVertex2f(a, b);

glVertex2f(x, y);

glEnd();

glFlush();

}

void myReshape(GLsizei w, GLsizei h)

{

glClearColor(1.0, 1.0, 1.0, 1.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

ww = w;

wh = h;

}

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

{

y = wh - y;

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

if (states == 0) {

states = 1;

a = x;

b = y;

}

else if(states == 1){

Polyline(x, y);

a = x;

b = y;

}

}

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

states = 0;

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

glFlush();

}

}

void myinit(void)

{

glViewport(0, 0, ww, wh);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(0.0, (GLdouble)ww, 0.0, (GLdouble)wh, -1.0, 1.0);

glFlush();

}

void display(void)

{

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutCreateWindow("Mouse Events - Polyline");

myinit();

glutReshapeFunc(myReshape);

glutMouseFunc(mymouse);

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

--------------------------------------------------------------------------------------------------------------------------------

6).

#include<stdio.h>

#include <windows.h>

#include<iostream>

#include <GL/glut.h>

using namespace std;

static int window;

static int menu\_id;

static int submenu\_id1, submenu\_id2;

static int value = 0;

void menu(int num) {

if (num == 0) {

glutDestroyWindow(window);

exit(0);

}

else {

value = num;

}

glutPostRedisplay();

}

void createMenu(void)

{

submenu\_id1 = glutCreateMenu(menu);

glutAddMenuEntry("Normal", 1);

glutAddMenuEntry("Print", 2);

glutAddMenuEntry("Web", 3);

submenu\_id2 = glutCreateMenu(menu);

glutAddMenuEntry("Cut", 4);

glutAddMenuEntry("Copy", 5);

glutAddMenuEntry("Paste", 6);

menu\_id = glutCreateMenu(menu);

glutAddSubMenu("View", submenu\_id1);

glutAddSubMenu("Edit", submenu\_id2);

glutAddMenuEntry("Quit", 0);

glutAttachMenu(GLUT\_RIGHT\_BUTTON);

}

void display(void) {

int i, n;

glClear(GL\_COLOR\_BUFFER\_BIT);

//print necessary stuff on output window

if (value == 1) {

cout << "Normal" << endl;

}

else if (value == 2) {

cout << "Print" << endl;

}

else if (value == 3) {

cout << "Web" << endl;

}

else if (value == 4) {

cout << "Cut" << endl;

}

else if (value == 5) {

cout << "Copy" << endl;

}

else if (value == 6) {

cout << "Paste" << endl;

}

glFlush();

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGBA | GLUT\_SINGLE);

glutInitWindowSize(500, 500);

glutInitWindowPosition(100, 100);

window = glutCreateWindow("Menus - display output");

createMenu();

glClearColor(0.0, 0.0, 0.0, 0.0);

glutDisplayFunc(display);

glutMainLoop();

return EXIT\_SUCCESS;

}

-----------------------------------------------------------------------------------------------------------------------

7).

#include<stdio.h>

#include <windows.h>

#include <GL/glut.h>

static int window;

static int menu\_id;

static int submenu\_id;

static int value = 1;

void menu(int num) {

if (num == 0) {

glutDestroyWindow(window);

exit(0);

}

else if(num == 1) {

value = value \* 2;

}

else {

if(value != 1)

value = value / 2;

}

glutPostRedisplay();

}

void createMenu(void)

{

submenu\_id = glutCreateMenu(menu);

glutAddMenuEntry("Increase size", 1);

glutAddMenuEntry("Decrease size", 2);

menu\_id = glutCreateMenu(menu);

glutAddSubMenu("Resize square", submenu\_id);

glutAddMenuEntry("Quit", 0);

glutAttachMenu(GLUT\_RIGHT\_BUTTON);

}

void display(void) {

int i, n;

glClear(GL\_COLOR\_BUFFER\_BIT);

glClear(GL\_COLOR\_BUFFER\_BIT);

glBegin(GL\_QUADS);

glVertex2f(0.2 \* value, 0.2 \* value);

glVertex2f(-0.2 \* value, 0.2 \* value);

glVertex2f(-0.2 \* value, -0.2 \* value);

glVertex2f(0.2 \* value, -0.2 \* value);

glEnd();

glFlush();

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGBA | GLUT\_SINGLE);

glutInitWindowSize(500, 500);

glutInitWindowPosition(100, 100);

window = glutCreateWindow("Menus - Square");

createMenu();

glClearColor(0.0, 0.0, 0.0, 0.0);

glutDisplayFunc(display);

glutMainLoop();

return EXIT\_SUCCESS;

}

-----------------------------------------------------------------------------------------------------------------------------

8).

#include <stdlib.h>

#include <GL/glut.h>

GLsizei wh = 500, ww = 500;

GLfloat size = 0.0;

void drawSquare(int x, int y)

{

size = rand()%7 + 3;

y = wh - y;

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

glBegin(GL\_POLYGON);

glVertex2f(x + size, y + size);

glVertex2f(x - size, y + size);

glVertex2f(x - size, y - size);

glVertex2f(x + size, y - size);

glEnd();

glFlush();

}

void myReshape(GLsizei w, GLsizei h)

{

glClearColor(1.0, 1.0, 1.0, 1.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

ww = w;

wh = h;

}

void myinit(void)

{

glViewport(0, 0, ww, wh);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(0.0, (GLdouble)ww, 0.0, (GLdouble)wh, -1.0, 1.0);

glFlush();

}

void display(void)

{

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutCreateWindow("Free-Hand");

myinit();

glutReshapeFunc(myReshape);

glutMotionFunc(drawSquare);

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

-------------------------------------------------------------------------------------------------------------------------------

9.

#include <stdlib.h>

#include <GL/glut.h>

GLsizei wh = 500, ww = 500;

int states = 0;

int a, b;

void drawLine(int x, int y)

{

glColor3f(0.0, 0.0, 0.0);

glBegin(GL\_LINES);

glVertex2f(a, b);

glVertex2f(x, y);

glEnd();

glFlush();

}

void myReshape(GLsizei w, GLsizei h)

{

glClearColor(1.0, 1.0, 1.0, 1.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

ww = w;

wh = h;

}

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

{

y = wh - y;

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

if (states == 0) {

states = 1;

a = x;

b = y;

}

else if (states == 1) {

states = 0;

drawLine(x, y);

}

}

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

states = 0;

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

glFlush();

}

}

void myinit(void)

{

glViewport(0, 0, ww, wh);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(0.0, (GLdouble)ww, 0.0, (GLdouble)wh, -1.0, 1.0);

glFlush();

}

void display(void)

{

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutCreateWindow("Mouse Events - lines");

myinit();

glutReshapeFunc(myReshape);

glutMouseFunc(mymouse);

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

-----------------------------------------------------------------------------------------------------------------------

10).