/**********Picking program for Linux**********/


```
#include<GL/glut.h>
#define N 5
GLdouble ww=500, wh=500;
void drawobject(GLenum mode)
{
	if(mode==GL_RENDER);
	glLoadName(1);
	glColor3f(1.0,0.0,0.0);
	glRectf(0.5,0.5,1.0,1.0);
	if(mode==GL_RENDER);
	glLoadName(2);
	glColor3f(0.0,0.0,1.0);
	glRectf(-1.0,-1.0,-0.5,-0.5);
}
void myMouse(int button, int state, int x, int y)
{
	GLuint namebuffer[10];
	GLint viewport[4];
	if(button==GLUT_LEFT_BUTTON && state==GLUT_DOWN)
	{
		glRenderMode(GL_RENDER);
		glGetIntegerv(GL_VIEWPORT,viewport);
		glMatrixMode(GL_PROJECTION);
		glPushMatrix();
		glLoadIdentity();
		gluPickMatrix((GLdouble)x,(GLdouble)(viewport[3]-y),N,N,viewport);
		gluOrtho2D(-5,5,-5,5);
		drawobject(GL_RENDER);
		glMatrixMode(GL_PROJECTION);
		glPopMatrix();
		glFlush();
		glRenderMode(GL_SELECT);
		glutPostRedisplay();
	}
}
void display()
{
	glClearColor(1.0,1.0,1.0,1.0);
	glClear(GL_COLOR_BUFFER_BIT);
	drawobject(GL_SELECT);
	glFlush();
}
void reshape(GLint w, GLint h)
{
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluOrtho2D(-5,5,-5,5);
	glMatrixMode(GL_MODELVIEW);
	glViewport(0,0,w,h);
	ww=w;
	wh=h;
}
int main(int argc, char **argv)
{
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_RGB|GLUT_SINGLE);
	glutInitWindowPosition(50,50);
	glutInitWindowSize(ww,wh);
	glutCreateWindow("picking example");
	glutMouseFunc(myMouse);
	glutDisplayFunc(display);
	glutReshapeFunc(reshape);
	glutMainLoop();
	return 1; }
```

```
#include<windows.h>
#include<iostream>
#include<GL/glu.h>
#include<GL/glut.h>
#define N 2
using namespace std;
void drawObjects()
{
	glLoadName(1);
	glColor3f(1.0,0.0,0.0);
	glRectf(0.5,0.5,1.0,1.0);
	glLoadName(2);
	glColor3f(0.0,0.0,1.0);
	glRectf(-1.0,-1.0,-0.5,-0.5);
}
void mouse(int button, int state, int x, int y)
{
	GLuint namebuffer[10];
	GLint viewport[4];
	if(button==GLUT_LEFT_BUTTON&&state==GLUT_DOWN)
	{
		glRenderMode(GL_SELECT);
		glInitNames();
		glPushName(-1);
		glSelectBuffer(10,namebuffer);
		glGetIntegerv(GL_VIEWPORT,viewport);
		glMatrixMode(GL_PROJECTION);
		glPushMatrix();
		glLoadIdentity();
		gluPickMatrix((GLdouble) x, (GLdouble) (viewport[3]-y),N,N,viewport);
		gluOrtho2D(-5,5,-5,5);
		drawObjects();
		glMatrixMode(GL_PROJECTION);
		glPopMatrix();
		glFlush();
		glutPostRedisplay();
	}
}
void display()
{
	glClear(GL_COLOR_BUFFER_BIT);
	drawObjects();
	glutSwapBuffers();
}
void reshape(GLint w, GLint h)
{
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluOrtho2D(-5,5,-5,5);
	glMatrixMode(GL_MODELVIEW);
}
int main(int argc, char **argv)
{
	glutInit(&argc,argv);
	glutInitWindowSize(500,500);
	glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB);
	glutCreateWindow("Picking");
	glutMouseFunc(mouse);
	glutDisplayFunc(display);
	glutReshapeFunc(reshape);
	glutMainLoop();
}
```

/*sample program on Menu using GLUT*/


```
#include<windows.h>
#include<GL/glut.h>
GLint size=2;
void display()
{
	glClearColor(0.0,0.0,0.0,1.0);
	glClear(GL_COLOR_BUFFER_BIT);
 }
void reshape(GLint w, GLint h)
{
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluOrtho2D(-10,10,-10,10);
	glMatrixMode(GL_MODELVIEW);
	glViewport(0,0,w,h);
}
void draw(int size)
{
	 glBegin(GL_POLYGON);
			glVertex2f(-size, -size);
			glVertex2f(size, -size);
			glVertex2f(size, size);
			glVertex2f(-size, size);
		glEnd();
	glFlush();
}
void demo_menu(int id)
{
	switch(id)
	{
		case 1: exit(0);
				break;
		case 3: size=2*size;
				glColor3f(0.0,0.0,1.0);
				draw(size);
				break;
		case 4: if(size>3)
				size=size/2;
				glColor3f(0.0,1.0,0.0);
				draw(size);
				break;
		case 2: glColor3f(1.0,0.0,1.0);
				draw(size);
				break;
	}
	glutPostRedisplay();
}

int main(int argc, char **argv)
{
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_DEPTH|GLUT_RGB|GLUT_SINGLE);
	glutInitWindowPosition(50,50);
	glutInitWindowSize(500,500);
	glutCreateWindow("sample menu");
	glutCreateMenu(demo_menu);
	glutAddMenuEntry("Quit", 1);
	glutAddMenuEntry("original",2);
	glutAddMenuEntry("Increase",3);
	glutAddMenuEntry("Decrease",4);
	glutAttachMenu(GLUT_RIGHT_BUTTON);
	glutDisplayFunc(display);
	glutReshapeFunc(reshape);
	glutMainLoop();
```

/* Program on sub menu*/

```
#include<windows.h>
#include<GL/glut.h>
GLint size=2;
int sub_menu;
void display()
{
	glClearColor(0.0,0.0,0.0,1.0);
	glClear(GL_COLOR_BUFFER_BIT);
 }
void reshape(GLint w, GLint h)
{
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluOrtho2D(-10,10,-10,10);
	glMatrixMode(GL_MODELVIEW);
	glViewport(0,0,w,h);
}
void draw(int size)
{
	 glBegin(GL_POLYGON);
			glVertex2f(-size, -size);
			glVertex2f(size, -size);
			glVertex2f(size, size);
			glVertex2f(-size, size);
		glEnd();
	glFlush();
}
void demo_menu(int id)
{
	switch(id)
	{
		case 1: exit(0);
				break;
		case 2: glColor3f(1.0,0.0,1.0);
				draw(size);
				break;
		case 3: size=2*size;
				glColor3f(0.0,0.0,1.0);
				draw(size);
				break;
		case 4: if(size>3)
				size=size/2;
				glColor3f(0.0,1.0,0.0);
				draw(size);
				break;
	}
	glutPostRedisplay();
}

int main(int argc, char **argv)
{
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_DEPTH|GLUT_RGB|GLUT_SINGLE);
	glutInitWindowPosition(50,50);
	glutInitWindowSize(500,500);
	glutCreateWindow("sample menu");
	sub_menu=glutCreateMenu(demo_menu);
	glutAddMenuEntry("Increase",3);
	glutAddMenuEntry("Decrease",4);
	glutCreateMenu(demo_menu);
```

```
	AddMenuEntry("Quit", 1);
	glutAddMenuEntry("original",2);
	glutAddSubMenu("Resize", sub_menu);
	glutAttachMenu(GLUT_RIGHT_BUTTON);
	glutDisplayFunc(display);
	glutReshapeFunc(reshape);
	glutMainLoop();
}
```

/* Rotating square using double buffer */

```
#include<windows.h>
#include<GL/glut.h>
#include<math.h>
GLfloat theta, thetar;
void display()
{
	glClear(GL_COLOR_BUFFER_BIT);
	glBegin(GL_POLYGON);
	glDrawBuffer(GL_FRONT_AND_BACK);
	glColor3f(0.0,0.0,1.0);
	thetar = theta/(3.141/180.0);
	glVertex2f(cos(thetar),sin(thetar));
	glVertex2f(-sin(thetar),cos(thetar));
	glVertex2f(-cos(thetar),-sin(thetar));
	glVertex2f(sin(thetar),-cos(thetar));
	glEnd();
	//for Outer circle
	glBegin(GL_POINTS);
	GLfloat r=1.0;
	for(int i=0;i<=360;i++)
		glVertex2f(cos(i*(3.141/180))*r, sin(i*(3.141/180))*r);
	glEnd();
	glutSwapBuffers();
}
//perform background processing tasks or continuous animation when window system events are not being received.
void idle()
{
	theta+=2;
	if(theta>=360.0)
		theta-=360.0;
	glutPostRedisplay();
}
void myMouse(int btn, int state, int x, int y)
{
	if(btn==GLUT_LEFT_BUTTON && state==GLUT_DOWN)
		glutIdleFunc(idle);
	if(btn==GLUT_RIGHT_BUTTON && state==GLUT_DOWN)
		glutIdleFunc(NULL);
}
void reshape(int w, int h)
{
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	glOrtho(-2,2,-2,2,2,-2);
	glMatrixMode(GL_MODELVIEW);
	glViewport(0,0,w,h);
}
int main(int argc, char **argv)
{
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_RGB|GLUT_DOUBLE);
	glutCreateWindow("rotate square");
	glutDisplayFunc(display);
	glutReshapeFunc(reshape);
```

```
    glutMouseFunc(myMouse);
    glutMainLoop();
}
```