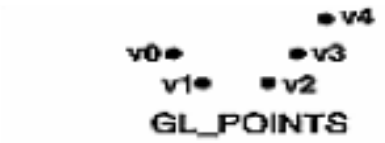
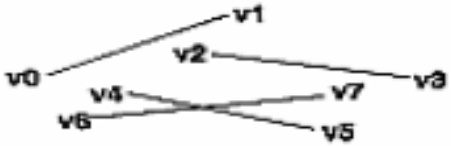
بسم الله الرحمن الرحيم

ادامه جزوه
open GL

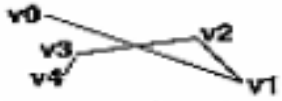
GL_POINTS	با هر راس بصورت یک نقطه برخورد می شود . (حداقل یک راس نیاز است .)
GL_LINES	با هر جفت راس بصورت یک خط مستقل برخورد می شود . (حداقل دو راس نیاز است.)
GL_LINE_STRIP	یک گروه به هم متصل از خطوط را از راس اول تابه آخر ترسیم می کند.
GL_LINE_LOOP	یک گروه به هم متصل از خطوط را از راس اول تابه آخر ترسیم می کند و سپس به اولین باز می گردد .
GL_TRIANGLES	با هر سه راس به صورت یک مثلث مستقل برخورد می کند . (حداقل سه راس نیاز است.)
GL_TRIANGLE_STRIP	یک گروه متصل به هم از مثلث ها را ترسیم می کند . یک مثلث به ازای هر راس پس از دو راس اول تعریف می شود .
GL_TRIANGLE_FAN	یک گروه متصل به هم از مثلث ها را ترسیم می کند . یک مثلث به ازای هر راس پس از دو راس اول تعریف می شود .
GL_QUADS	با هر گروه چهارتایی از رئوس بصورت یک چهار ضلعی مستقل رفتار می کند . (حداقل چهار راس نیاز است.)
GL_QUAD_STRIP	یک گروه متصل به هم از چهار ضلعی ها را ترسیم می کند . یک چهار ضلعی به ازای هر راس پس از جفت اول تعریف می شود .
GL_POLYGON	یک چند ضلعی محدب ترسیم می کند . رئوس ۱ تا n این چند ضلعی را تعریف می کنند . (حداقل سه راس نیاز است .)



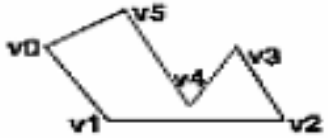
GL_POINTS



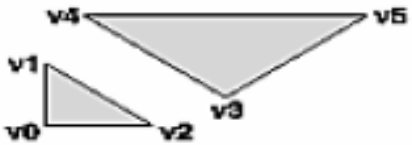
GL_LINES



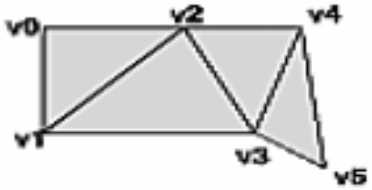
GL_LINE_STRIP



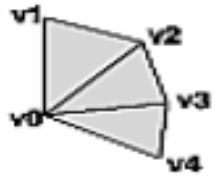
GL_LINE_LOOP



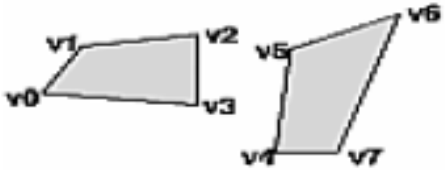
GL_TRIANGLES



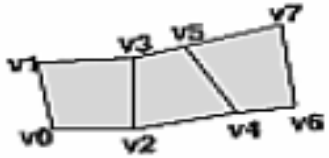
GL_TRIANGLE_STRIP



GL_TRIANGLE_FAN



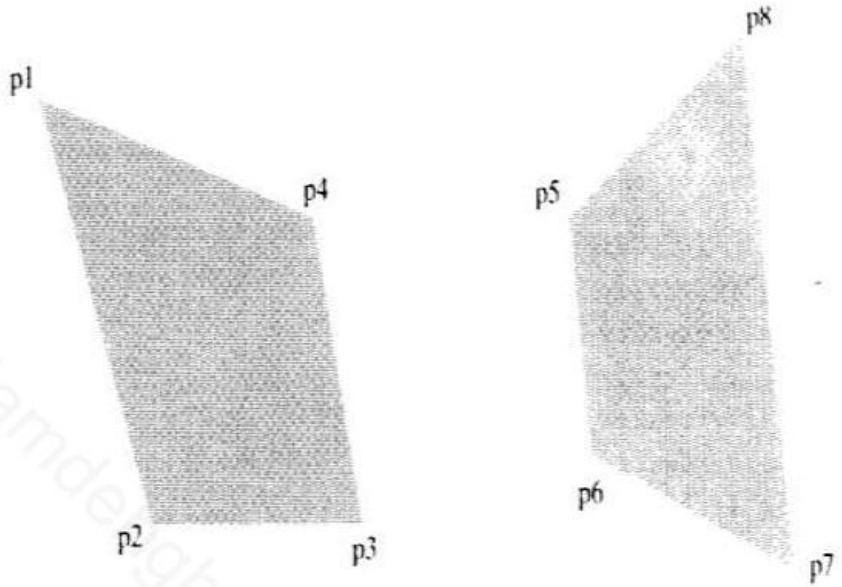
GL_QUADS



GL_QUAD_STRIP

GL_POLYGON

```
glBegin (GL_QUADS);
    glVertex2iv (p1);
    glVertex2iv (p2);
    glVertex2iv (p3);
    glVertex2iv (p4);
    glVertex2iv (p5);
    glVertex2iv (p6);
    glVertex2iv (p7);
    glVertex2iv (p8);
glEnd ( );
```



چهار نقطهٔ اول، رئوس یک چهارضلعی، چهار نقطهٔ بعدی، رئوس چهارضلعی بعدی، و تا آخر را تعریف می‌کنند. برای هر سطح مجموعه‌ای از سطوح پر از چهارضلعیهای متصل به هم، نمایش داده می‌شود.

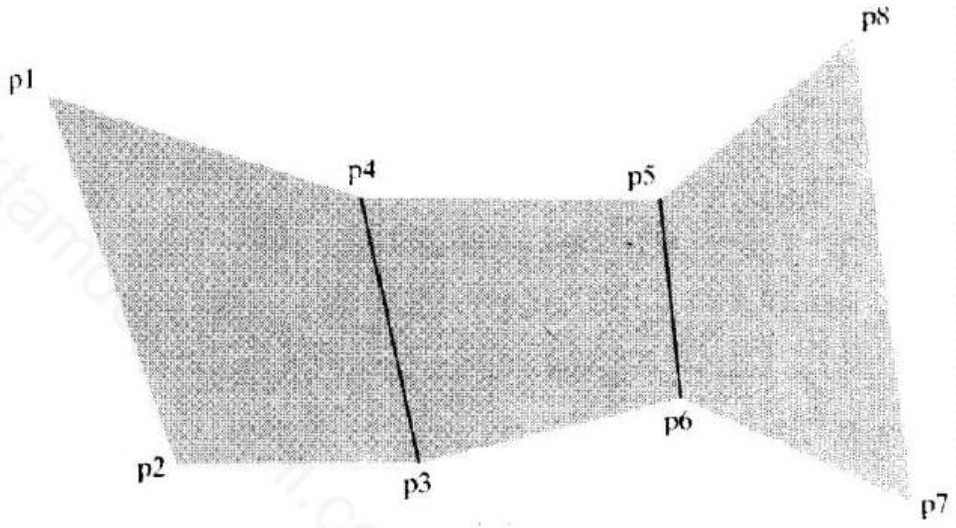
اگر N مضربی از ۴ نباشد، رئوس مازاد مورد استفاده قرار نخواهند گرفت.

لازم است رئوس به گونه ای فهرست شوند که ترتیب رئوس پاد ساعتگرد باشد

For i=1 … n/2-2 //number of quads

(2i-1,2i,2i+2,2i+1) makes ith guad

```
glBegin (GL_QUAD_STRIP);
    glVertex2iv (p1);
    glVertex2iv (p2);
    glVertex2iv (p4);
    glVertex2iv (p3);
    glVertex2iv (p5);
    glVertex2iv (p6);
    glVertex2iv (p8);
    glVertex2iv (p7);
glEnd ( );
```

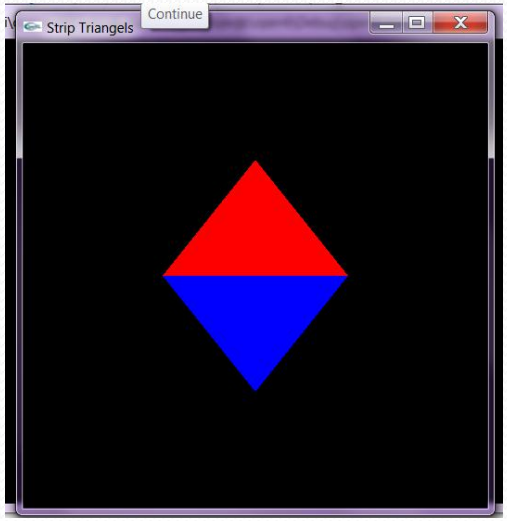


بعد از دو رأس اول، برای هر زوج از رئوس مشخص شده در فهرست، یک چهارضلعی تولید می‌شود و لازم است که رئوس به‌گونه‌ای فهرست شوند که ترتیبی صحیح در خلاف جهت چرخش عقربه‌های ساعت برای رئوس هر چهارضلعی تولید شود.

برای فهرستی از N رأس با شرط $N \geq ۴$، $N/۲ - ۱$ چهارضلعی حاصل می‌شود.

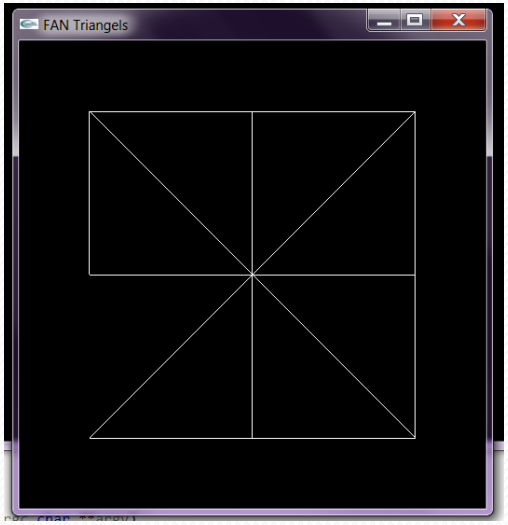
```
#include<gl/glut.h>
void init()
{
glClearColor(0,0,0,0);
glMatrixMode(GL_PROJECTION);
gluOrtho2D(100,100,0,200);
}
void display()
{
glClear(GL_COLOR_BUFFER_BIT);
glBegin(GL_TRIANGLE_STRIP);
glColor3f(0,0,1.0);
glVertex2f(0.4, 0.0);
glVertex2f(0.0, -0.5);
glVertex2f(-0.4, 0.0);
glColor3f(1,0,0);
glVertex2f(-0.4, 0.0);
glVertex2f(0.0, 0.5);
glVertex2f(0.4, 0.0);
glEnd();
glFlush();
}
```

```
int main(int argc,char **argv)
{
glutInit(&argc,argv);
glutInitDisplayMode(GLUT_SINGLE|
GLUT_RGB);
glutInitWindowSize(500,500);
glutInitWindowPosition(200,200);
glutCreateWindow("Strip Triangels");
init();
glutDisplayFunc(display);
glutMainLoop();
return 0;
```



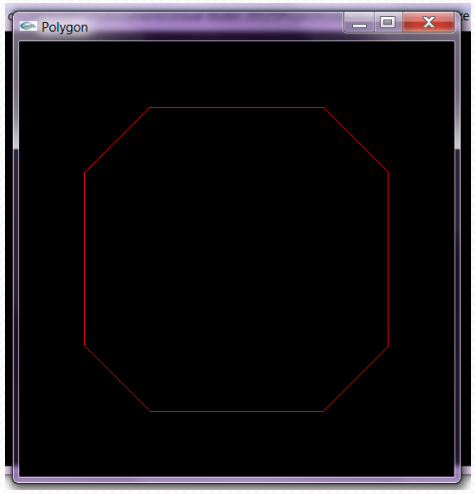
```
#include<gl/glut.h>
void init()
{
glClearColor(0,0,0,0);
glMatrixMode(GL_PROJECTION);
gluOrtho2D(100,100,0,200);
}
void display()
{
glClear(GL_COLOR_BUFFER_BIT);
glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
glBegin(GL_TRIANGLE_FAN);
glVertex2f(-0.0, 0.0);
glVertex2f(-0.7, 0.0);
glVertex2f(-0.7, 0.7);
glVertex2f(0.0, 0.7);
glVertex2f(0.7, 0.7);
glVertex2f(0.7, 0.0);
glVertex2f(0.7, -0.7);
glVertex2f(0.0, -0.7);
glVertex2f(-0.7, -0.7);
glEnd();
glFlush();
}
```

```
int main(int argc,char **argv)
{
glutInit(&argc,argv);
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
glutInitWindowSize(500,500);
glutInitWindowPosition(200,200);
glutCreateWindow("FAN Triangels");
init();
glutDisplayFunc(display);
glutMainLoop();
return 0;
}
```



```
#include<gl/glut.h>
void init()
{
glClearColor(0,0,0,0);
glMatrixMode(GL_PROJECTION);
gluOrtho2D(100,100,0,200);
}
void display()
{
glClear(GL_COLOR_BUFFER_BIT);
glColor3f(1.0,0.0,0.0);
glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
glBegin(GL_POLYGON);
glVertex2f(-0.7, -0.4);
glVertex2f(-0.7, 0.4);
glVertex2f(-0.4, 0.7);
glVertex2f(0.4, 0.7);
glVertex2f(0.7, 0.4);
glVertex2f(0.7, -0.4);
glVertex2f(0.4, -0.7);
glVertex2f(-0.4, -0.7);
glEnd();
glFlush();
}
```

```
int main(int argc,char **argv)
{
glutInit(&argc,argv);
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
glutInitWindowSize(500,500);
glutInitWindowPosition(200,200);
glutCreateWindow("Polygon“);
init();
glutDisplayFunc(display);
glutMainLoop();
return 0;
}
```



```
#include <GL/glut.h>
void Init(void)
{
glClearColor(1.0,1.0,1.0,0.0);
};
void display()
{
glClear(GL_COLOR_BUFFER_BIT);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glPushMatrix();
glRotatef(0,0.0,0.0,1.0);
glBegin(GL_POLYGON);
glColor3f(1.0, 0.0, 0.0);
glVertex2f(-0.5, -0.5);
glColor3f(0.0, 1.0, 0.0);
glVertex2f(0.5, -0.5);
glColor3f(0.0, 0.0, 1.0);
glVertex2f(0.0, 0.5);
glEnd();
glPopMatrix();
glutSwapBuffers();
}
```

```
int main(int argc, char **argv)
{glutInit(&argc,argv);
glutInitDisplayMode(GLUT_DOUBLE |
GLUT_RGB);
glutInitWindowPosition(100,100);
glutInitWindowSize(400,400);
glutCreateWindow("Gradiant Triangle");
Init();
glutDisplayFunc(display);
glutMainLoop();
return 0;
}
```