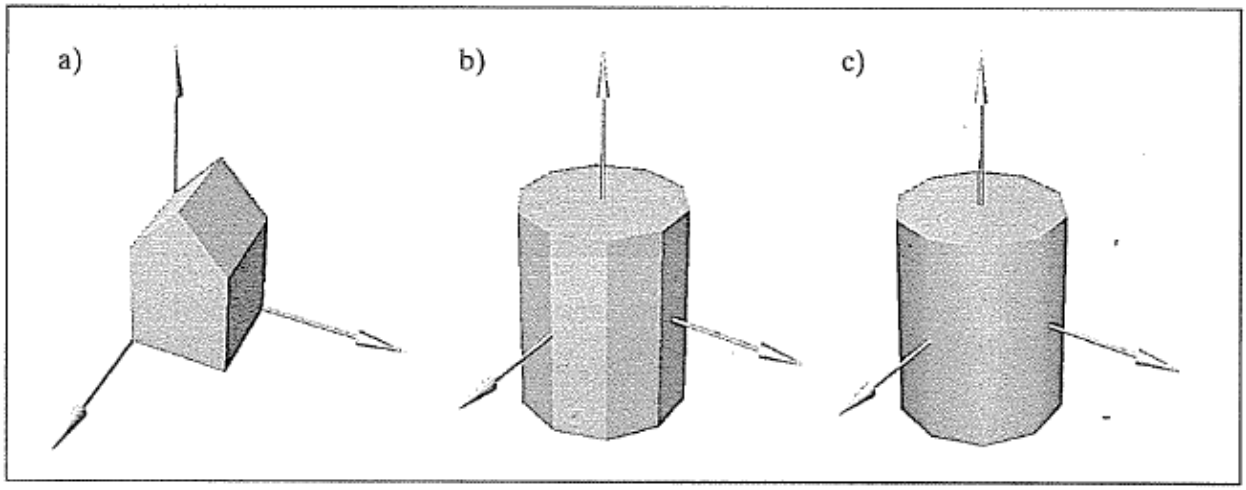
بسمه تعالی

# شبکه های چندضلعی-۱

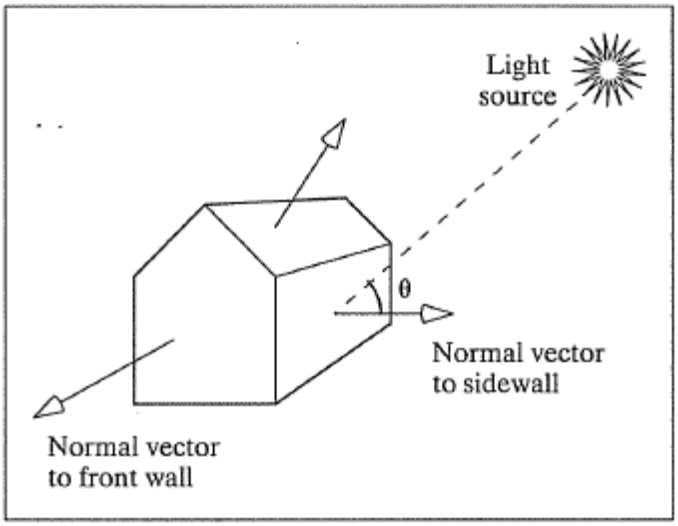
## مقدمه

بسیاری از اشیاء را می توان با شبکه های چندضلعی نمایش داد. یک شبکهٔ چندضلعی مجموعه ای از چندضلعیهای به هم متصل می باشد. این چندضلعیها را می توان بصورت تک تک با استفاده از دستورهائی که یک چندضلعی را نمایش می دهند ترسیم نمود ولی با توجه به تعداد زیاد چندضلعیهائی که باید ترسیم شوند این روش کارآئی لازم را ندارد.



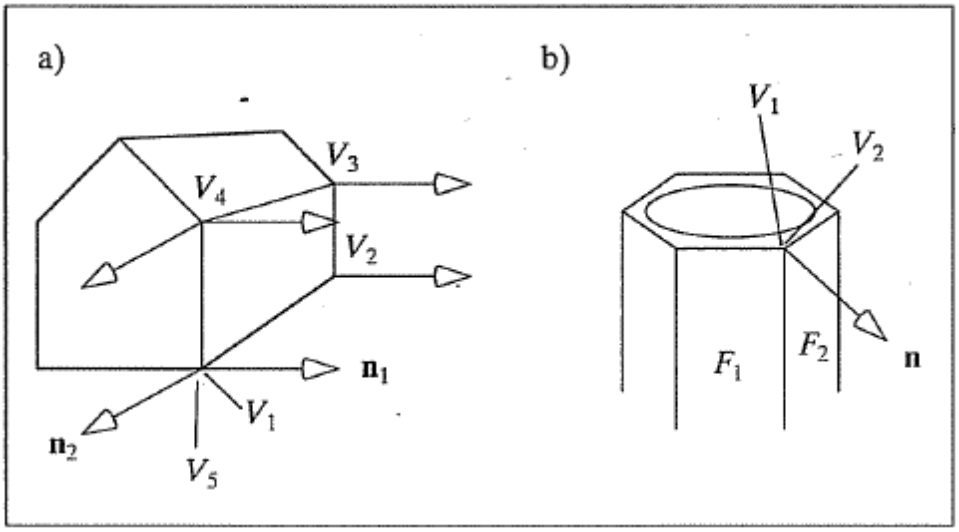

شکل ۱

برای مشخص کردن شبکهٔ چندضلعی به لیستی از چندضلعیهای موجود در شبکه به همراه جهتی که هر چندضلعی رو کرده است نیاز داریم.




شکل ۲

خواهیم دید که بهتر است به هر رأس یک عمود وابسته کنیم. به عنوان مثال برای رأس V4 در شکل خانه در شکل ۳ سه بردار عمود برای هر یک از وجوهی که در آن قرار دارد انتساب داده می شود. برای برخی از اشیاء نرم که با شبکه چندضلعی مدل می شوند برای رنگ آمیزی و سایه زنی بهتر به رأسی که بین دو وجه مشترک است ممکن است یک بردار نسبت داده شود.



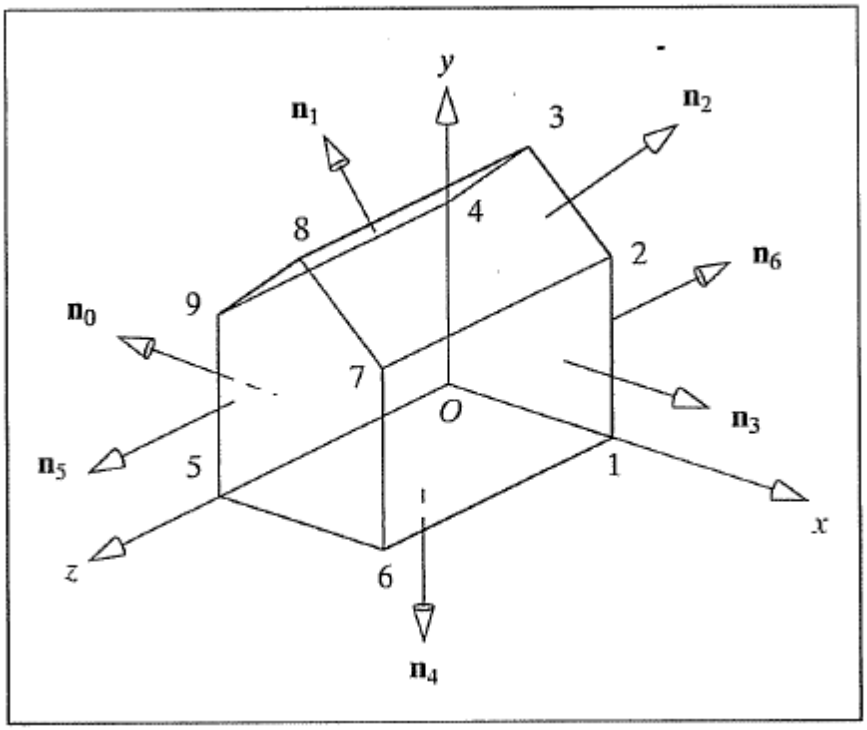

شکل ۳

یک روش برای نمایش یک شبکهٔ چندضلعی آن است که برای هر چندضلعی مختصات همه نقاط آن را لیست کنیم. چون چندضلعیها در محل تماس نقاط مشترک دارند این کار باعث ذخیره مقدار زیادی اطلاعات زائد خواهد بود. روش دیگر استفاده از سه لیست می باشد:

لیست رأس: مختصات رئوس درون شبکه

لیست عمود: جهت عمودهای موجود در شکل

لیست وجه: دارای اندیسهائی به دو لیست دیگر




| Vertex | x | y | z |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 |
| 2 | 1 | 1 | 0 |
| 3 | 0.5 | 1.5 | 0 |
| 4 | 0 | 1 | 0 |
| 5 | 0 | 0 | 1 |
| 6 | 1 | 0 | 1 |
| 7 | 1 | 1 | 1 |
| 8 | 0.5 | 1.5 | 1 |
| 9 | 0 | 1 | 1 |

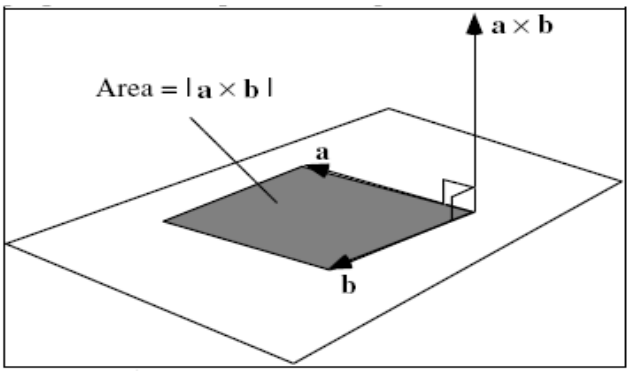
| Normal | $n_x$ | $n_y$ | $n_z$ |
|---|---|---|---|
| 0 | −1 | 0 | 0 |
| 1 | −0.707107 | 0.707107 | 0 |
| 2 | 0.707107 | 0.707107 | 0 |
| 3 | 1 | 0 | 0 |
| 4 | 0 | −1 | 0 |
| 5 | 0 | 0 | 1 |
| 6 | 0 | 0 | −1 |

| Face | Vertices | Associated Normal |
|---|---|---|
| 0 (left) | 0, 5 ,9 ,4 | 0, 0, 0, 0 |
| 1 (roof left) | 3, 4, 9, 8 | 1, 1, 1, 1 |
| 2 (roof right) | 2, 3, 8, 7 | 2, 2, 2, 2 |
| 3 (right) | 1, 2, 7, 6 | 3, 3, 3, 3 |
| 4 (bottom) | 0, 1, 6, 5 | 4, 4, 4, 4 |
| 5 (front) | 5, 6, 7, 8, 9 | 5, 5, 5, 5, 5 |
| 6 (back) | 0, 4, 3, 2, 1 | 6, 6, 6, 6, 6 |

رئوس یک چندضلعی در خلاف جهت عقربه های ساعت چنانکه از بیرون شئ دیده می شود لیست می شوند. در صورتی که روی رئوس به ترتیب گفته شده قدم بزنیم داخل جسم همواره سمت چپ ما خواهد بود.

## یافتن عمود رئوس

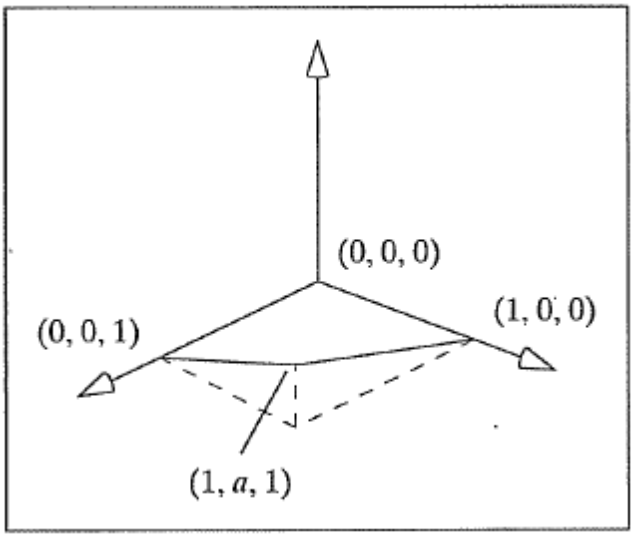
می توان عمود هر رأس را با دست محاسبه کرد. باید در نظر گرفت که هر وجه سه یا بیشتر رأس دارد و محاسب دستی خسته کننده و گاه دشوار است. راه بهتر آن است که بخواهیم برنامه عمود را محاسبه کند. می دانیم:




با داشتن هر سه رأس متوالی V1، V2، V3 (در خلاف جهت عقربه های ساعت) محاسبهٔ m=(V3-V2)x(V1-V2)

آنرا عادی ساخته تا بردار یکه بدست آید.

چند مشکل در این روش می تواند وجود داشته باشد. اگر سه رأس تقریباً در یک راستا باشند، مقدار ضرب خارجی کوچک بوده و ممکن است مقداری عدم دقت عددی بدست آید. ممکن است همهٔ رئوس وجه در یک صفحه قرار نگیرند (بخاطر برخی روشهای مدلسازی)




روش نیول:

$$m_x = \sum_{i=0}^{N-1} (y_i - y_{next(i)})(z_i + z_{next(i)})$$
$$m_y = \sum_{i=0}^{N-1} (z_i - z_{next(i)})(x_i + x_{next(i)})$$
$$m_z = \sum_{i=0}^{N-1} (x_i - x_{next(i)})(y_i + y_{next(i)})$$

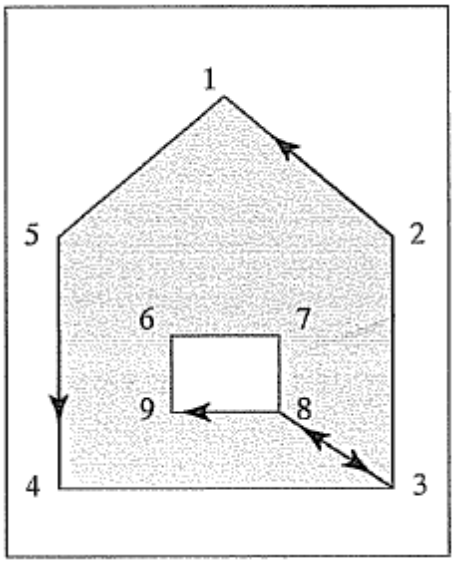
N تعداد رئوس در وجه

next(j)=(j+1)mod N

**وجوه سوراخدار**

ترتیب رئوس:5,4,3,8,9,6,7,8,3,2,1

اضلاع چندضلعی درونی در جهت حرکت عقربه های ساعت طی می شوند. در این وضعیت، مجدداً داخل شئ سمت چپ قرار می گیرد.



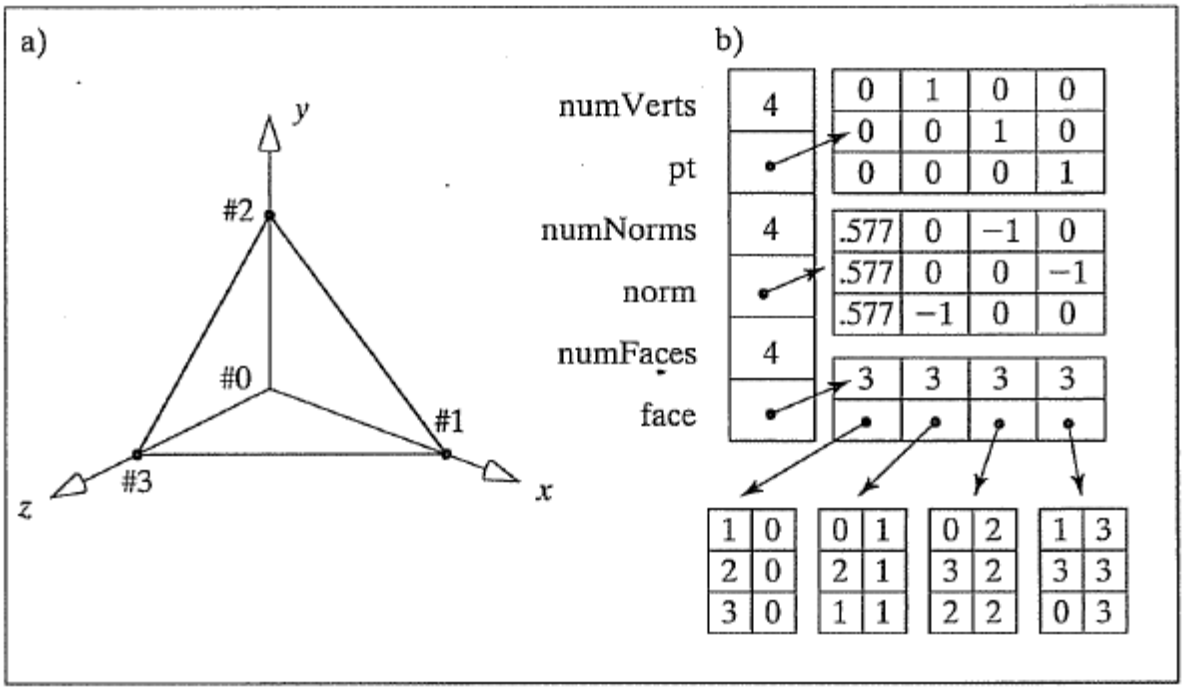

**کار با شبکهٔ چندضلعی در برنامه**

یک شبکهٔ چندضلعی دارای یک لیست رأس، یک لیست عمود، و یک لیست وجه است.

```
//################## VertexID ###################
class VertexID{
   public:
      int vertIndex;    // index of this vertex in the vertex list
      int normIndex;    // index of this vertex's normal
};
//#################### Face #################
class Face{
   public:
      int nVerts;      // number of vertices in this face
      VertexID * vert;      // the list of vertex and normal indices
      // constructor
      // destructor
};
//###################### Mesh ######################
class Mesh{
   private:
      int numVerts;     // number of vertices in the mesh
      Point3* pt;       // array of 3D vertices
      int numNormals;   // number of normal vectors for the mesh
      Vector3 *norm;    // array of normals
      int numFaces;     // number of faces in the mesh
      Face* face;       // array of face data
      // ... others to be added later
   public:
      Mesh();           // constructor
      ~Mesh();          // destructor
      int readFile(char * fileName);    // to read in a filed mesh
      .. others ..
};
```

برای یافتن عمود رأس vام در وجه fام norm[face[f].vert[v].normIndex]

مثال

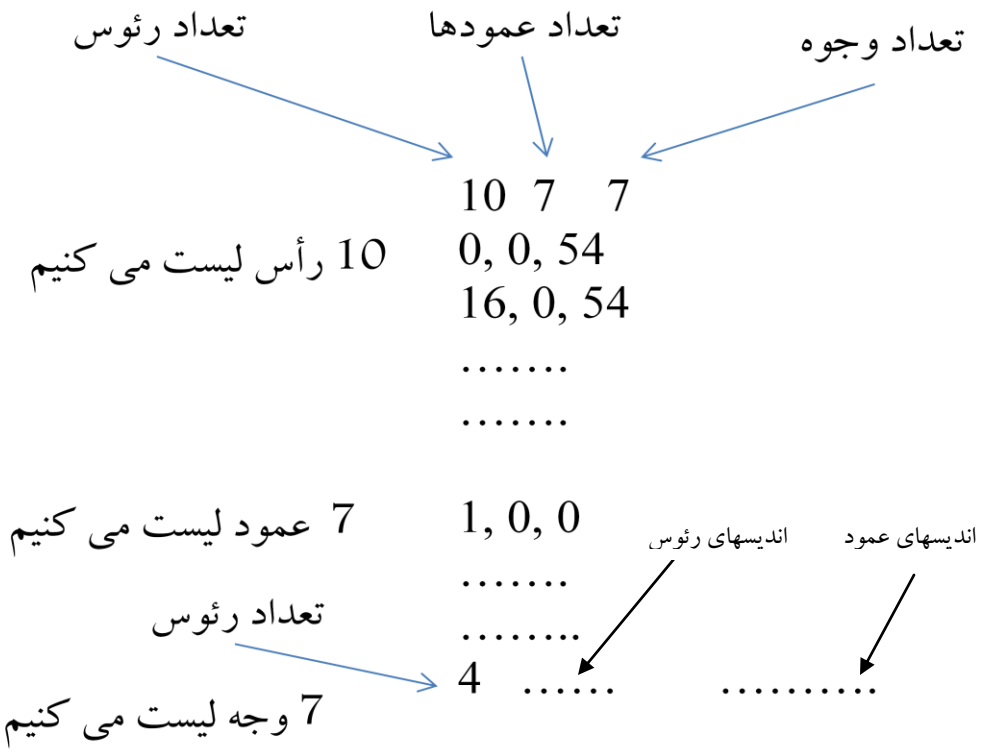



برای رسم روال draw در Mesh را می نویسیم و به ترتیب هر وجه را رسم می کنیم.

```
void Mesh:: draw()      // use OpenGL to draw this mesh
{
   for (int f = 0; f < numFaces; f++)      // draw each face
   {
      glBegin (GL_POLYGON);
         for (int v = 0; v < face[f].nVerts; v++)     // for each one..
         {
            int in = face[f].vert[v].normIndex ;    // index of this normal
            int iv = face[f].vert[v].vertIndex ;    // index of this vertex
            glNormal3f(norm[in].x, norm[in].y, norm[in].z);
            glVertex3f(pt[iv].x, pt[iv].y, pt[iv].z);
         }
      glEnd();
   }
}
```

### ایجاد یک پرونده برای ورود شبکهٔ چندضلعی

حالت ساده:




**چند وجهیها**

تعداد زیادی از اجسام صلب مورد علاقه چندوجهی هستند.

**تعریف:** یک چند وجهی یک شبکهٔ متصل از چندضلعیهای مسطح است که فضای محدودی را در برمی گیرد.

خواص:

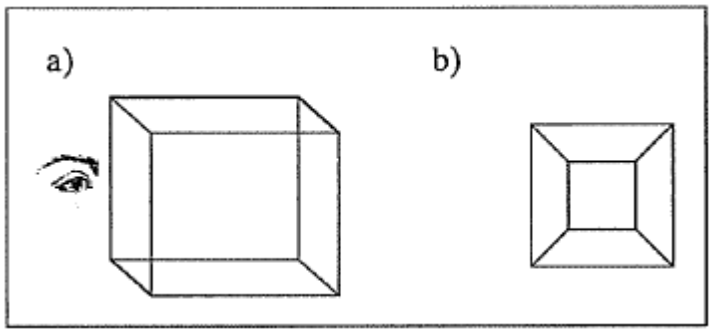
هر لبه دقیقاً بوسیلهٔ ۲ وجه مشترک می شود. حداقل سه لبه در یک رأس ملاقات می کنند. وجوه داخل یکدیگر نمی شوند.

فرمول اولر برای چندوجهی ساده V+F-E=2 که در آن V تعداد رئوس، F تعداد وجوه، و E تعداد لبه ها می باشد.

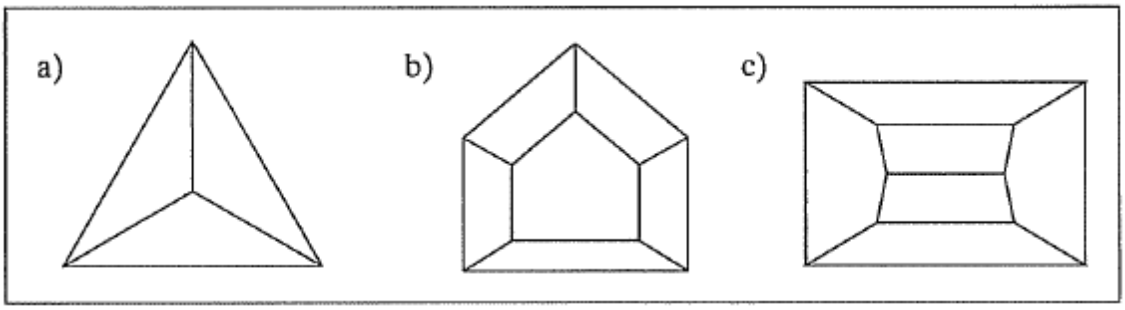
مثال: برای یک مکعب V=8, F=6, E=12

ساختار یک چندوجهی را می توان بوسیلهٔ دیاگرام اشلگل[1] نمایش داد. این دیاگرام بر اساس نگرش یک چندوجهی از یک نقطه خارج از مرکز یکی از وجوهش بوجود می آید.

[1] Schlegel




شکل ٤




شکل ٥ دیاگرام اشلگل براي یک جهار وجهي، یک خانه از جلو، و یک خانه از بالا.

شکل ٥ a و b دیاگرام اشلگل برای یک چهاروجهی و یک خانه را چنانکه از مقابل به آنها نگریسته شود نشان می دهد. شکل c دیاگرام اشلگل را در حالی که از بالا به خانه نگریسته شود نشان می دهد.