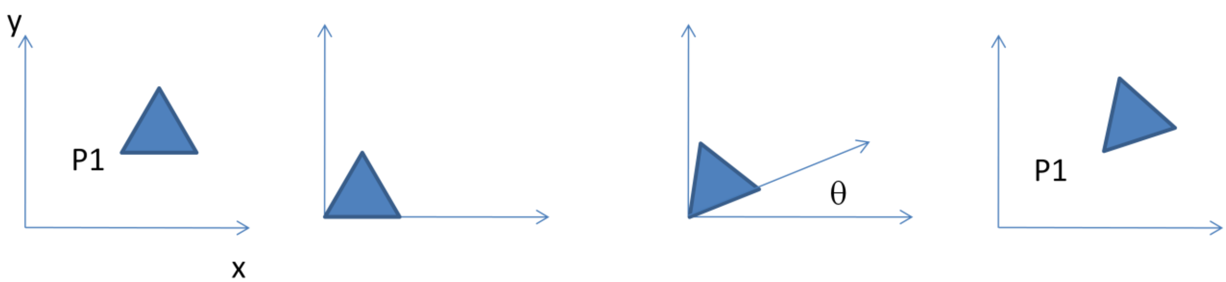
تبدیلهای هندسی - ۲

دوران حول یک نقطهٔ خاص

تا کنون تمامی دورانها حول مبدأ بود. فرض کنید بخواهیم حول نقطهٔ دلخواه P1 دوران دهیم. چون دوران حول مبدأ را می دانیم، مسئلهٔ اوّلیه را به را به چند مسئلهٔ ساده تر تبدیل می کنیم:

- انتقال P1 به مبدأ
- دوران
- انتقال به مکان اوّلیه



- انتقال به مبدأ: T(-x1,-y1)
- دوران: R(θ)
- انتقال به مکان اوّلیه: T(x1,y1)

$$T(x_1,y_1).R(\theta).T(-x_1,-y_1)=\begin{bmatrix}1&0&x_1\\0&1&y_1\\0&0&1\end{bmatrix}.\begin{bmatrix}\cos(\theta)&-\sin(\theta)&0\\\sin(\theta)&\cos(\theta)&0\\0&0&1\end{bmatrix}.\begin{bmatrix}1&0&-x_1\\0&1&-y_1\\0&0&1\end{bmatrix}$$

$$=\begin{bmatrix}\cos(\theta)&-\sin(\theta)&x_1(1-\cos(\theta))+y_1\sin(\theta)\\\sin(\theta)&\cos(\theta)&y_1(1-\cos(\theta))+x_1\sin(\theta)\\0&0&1\end{bmatrix}$$

تغییراندازه حول نقطهٔ دلخواه

بطور مشابه

$$T(x_1,y_1).S(s_x,s_y).T(-x_1,-y_1)=\begin{bmatrix}1&0&x_1\\0&1&y_1\\0&0&1\end{bmatrix}.\begin{bmatrix}s_x&0&0\\0&s_y&0\\0&0&1\end{bmatrix}.\begin{bmatrix}1&0&-x_1\\0&1&-y_1\\0&0&1\end{bmatrix}$$

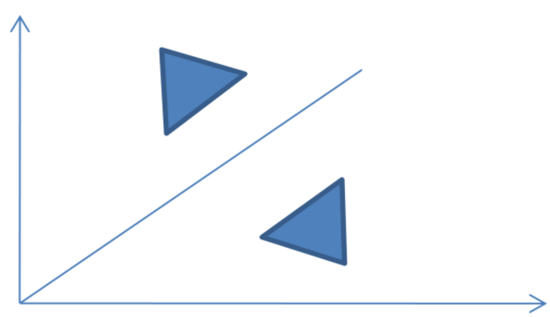
$$=\begin{bmatrix}s_x&0&x_1(1-s_x)\\0&s_y&y_1(1-s_y)\\0&0&1\end{bmatrix}$$

انعکاس حول یک خط دلخواه

فرض کنید خط دلخواه که از مبدأ عبور می کند با محور x زاویهٔ β بسازد. مراحل:

- دوران β- حول مبدأ تا خط بر روی محور x منطبق شود
- انعکاس
- دوران β حول مبدأ

$$R(\beta).S(1,-1).R(-\beta)=\begin{bmatrix}\cos(2\beta)&\sin(2\beta)&0\\\sin(2\beta)&-\cos(2\beta)&0\\0&0&1\end{bmatrix}$$



مثال

فرض کنید می خواهیم خانه را حول P_1 تغییر اندازهٔ و دوران داده و سپس به نقطهٔ P_2 منتقل کنیم.

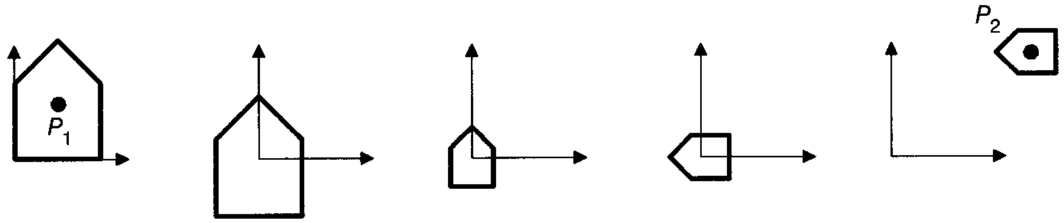


Fig. 5.9 Rotation of a house about the point P_1, and placement such that what was at P_1 is at P_2.

مراحل:

- انتقال به مبدأ
- تغییر اندازه حول مبدأ
- دوران حول مبدأ
- انتقال به P2

$$T(x_2, y_2).R(\theta).S(s_x, s_y).T(-x_1, -y_1)$$

بطور کلی ترتیب انجام تبدیلها مهم است. اگر M1 و M2 دو ماتریس تبدیل بنیادی باشند، در حالات زیر M1.M2=M2.M1

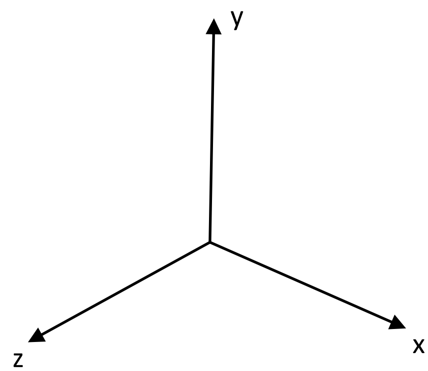
M1	M2
انتقال	انتقال
تغییر اندازه	تغییراندازه
دوران	دوران
تغییر اندازهٔ یکنواخت	دوران

تبدیلهای سه بعدی

تبدیلهای ۲ بعدی توسط ماتریسهای ۳×۳ انجام در مختصات همگن انجام شدند. تبدیلهای ۳ بعدی نیز توسط ماتریسهای ۴×۴ با استفاده از نمایش مختصات همگن نقاط انجام می گیرند. نقطهٔ (x,y,z) توسط (x,y,z,W) در دستگاه مختصات همگن نمایش داده می شود. دو چهارتائی نقطهٔ مشابهی را نمایش می دهند اگر هر یک مضرب غیرصفر دیگری باشد. نقطهٔ (0,0,0,0) مجاز نیست.

مشابه حالت دوبعدی، نمایش استاندارد یک نقطهٔ (x,y,z,W) که در آن $W\neq0$ بوسیلهٔ (x/W,y/W,z/W,1) داده می شود و همگن کردن نقطه نامیده می شود. همچنین نقاطی که مختصهٔ W آنها صفر است نقاط در بی نهایت نامیده می شوند. هر نقطه در فضای سه بعدی توسط یک خط که از مبدأ عبور می کند در فضای چهاربعدی نمایش داده می شود و نمایش همگن شدهٔ این نقاط یک زیر فضای سه بعدی از آن فضا را تشکیل می دهد که توسط معادلهٔ W=1 تعریف می شود.

دستگاه مختصات سه بعدی را راستگرد در نظر می گیریم: اگر در راستای z بایستیم، محور x در خلاف جهت عقربه های ساعت باید ۹۰ درجه بچرخد تا بر روی محورy منطبق شود.



در یک دستگاه چپگرد زوایای مثبت در جهت حرکت عقربه های ساعت در نظر گرفته می شوند.

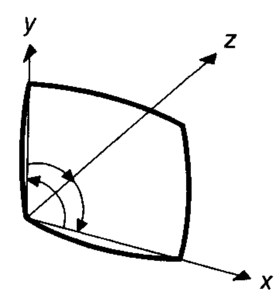


Fig. 5.15 The left-handed coordinate system, with a superimposed display screen.

انتقال سه بعدی

$$T(d_x, d_y, d_z) = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

تغییر اندازهٔ سه بعدی

$$S(s_x, s_y, s_z) = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

دوران سه بعدی دارای تنوع بیشتری است.

دوران حول محور z

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

دوران حول محور x

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

دوران حول محور y

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

سطرها (و ستونهای) ماتریسهای دارای طول واحد بوده و برهم عمود هستند. ماتریس عمود خاص هستند.

سه نوع ماتریس کشش سه بعدی بنیادی نیز وجود دارند.

کشش به موازات صفحهٔ xy

$$SH_{xy} = \begin{bmatrix} 1 & 0 & sh_x & 0 \\ 0 & 1 & sh_y & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

انتقال نقطهٔ (x,y,z) به $(x+sh_x.z,\ y+sh_y.z,\ z)$. کشش به موازات صفحهٔ yz و zx بطور مشابه تعریف می شوند.

مثال:

انتقال خطوط P_1P_2 و P_1P_3 از محل اوّلیهٔ خود به محل نهائی خود. به دو روش آن را انجام می دهیم.

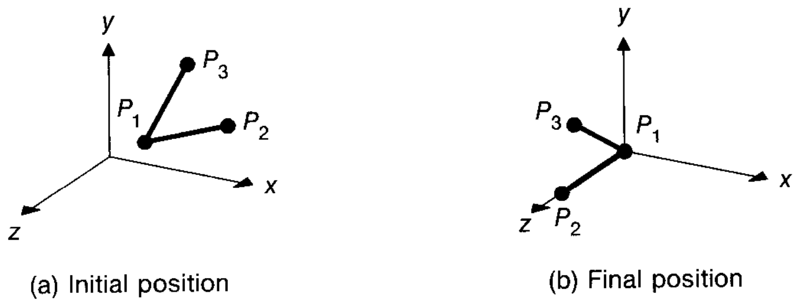


Fig. 5.16 Transforming P_1, P_2, and P_3 from their initial (a) to their final (b) position.

روش اوّل: تجزیهٔ مسئله به مسائل ساده تر

۱. انتقال P1 به مبدأ

۲. دوران حول محور y بطوریکه P_1P_2 در صفحهٔ yz قرار گیرد.

۳. دوران حول محور x بطوریک P_1P_2 روی محور z قرار گیرد.

٤. دوران حول محور z بطوریکه P_1P_3 در صفحهٔ yz قرار گیرد.

مرحلهٔ اوّل: انتقال P1 به مبدأ

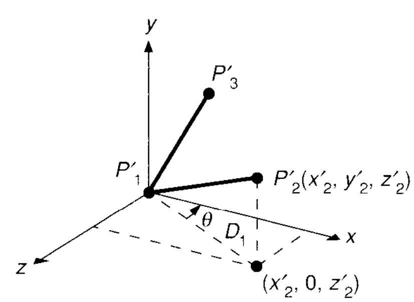


Fig. 5.17 Rotation about the y axis: The projection of $P_1'P_2'$, which has length D_1, is rotated into the z axis. The angle θ shows the positive direction of rotation about the y axis: The actual angle used is $-(90 - \theta)$.

$$T(-x_1,-y_1,-z_1)=\begin{bmatrix}1&0&0&-x_1\\0&1&0&-y_1\\0&0&1&-z_1\\0&0&0&1\end{bmatrix}$$

اعمال T به نقاط P1، P2، و P3

$$P_1^{'}=T(-x_1,-y_1,-z_1).P_1=\begin{bmatrix}0\\0\\0\\1\end{bmatrix}$$

$$P_2^{'}=T(-x_1,-y_1,-z_1).P_2=\begin{bmatrix}x_2-x_1\\y_2-y_1\\z_2-z_1\\1\end{bmatrix}$$

$$P_3^{'}=T(-x_1,-y_1,-z_1).P_3=\begin{bmatrix}x_3-x_1\\y_3-y_1\\z_3-z_1\\1\end{bmatrix}$$

مرحلهٔ دوّم: دوران حول محور y

زاویهٔ دروان 90-θ=(θ-90)–

$$\cos(\theta-90)=\sin\theta=\frac{z_2^{'}}{D_1}=\frac{z_2-z_1}{D_1}$$
$$\sin(\theta-90)=-\cos\theta=\frac{x_2^{'}}{D_1}=-\frac{x_2-x_1}{D_1}$$

$$D_1=\sqrt{(z_2^{'})^2+(x_2^{'})^2}$$

$$P_2^{''}=R_y(\theta-90).P_2^{'}=\begin{bmatrix}0 & y_2-y_1 & D_1 & 1\end{bmatrix}^T$$

مرحلهٔ سوّم: دوران حول محور x

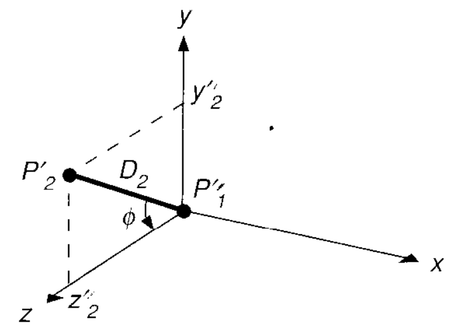


Fig. 5.18 Rotation about the x axis: $P_1'P_2'$ is rotated into the z axis by the positive angle ϕ. D_2 is the length of the line segment. The line segment P_1P_3 is not shown, because it is not used to determine the angles of rotation. Both lines are rotated by $R_x(\phi)$.

$$\cos\phi = \frac{z_2''}{D_2}, \quad \sin\phi = \frac{y_2''}{D_2}$$

$$D_2 = \left|P_1''P_2''\right| = \left|P_1P_2\right|$$

$$P_2''' = R_x(\phi).P_2'' = \begin{bmatrix}0 & 0 & \left|P_1P_2\right| & 1\end{bmatrix}^T$$

مرحلهٔ چهارم: دوران حول محور z

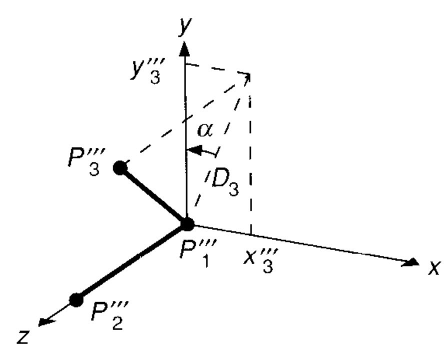


Fig. 5.19 Rotation about the z axis: The projection of $P_1'P_3'$, whose length is D_3, is rotated by the positive angle α into the y axis, bringing the line itself into the (y, z) plane. D_3 is the length of the projection.

$$\cos\alpha = \frac{y_3'''}{D_3}, \quad \sin\alpha = \frac{x_3'''}{D_3}$$

$$D_3 = \sqrt{\left(x_3'''^2 + y_3'''^2\right)}$$

ماتریس تبدیل نهائی

$$R_z(\alpha).R_x(\phi).R_y(\theta - 90).T(-x_1, -y_1, -z_1)$$

روش دوّم: با استفاده از خواص ماتریسهای عمود خاص

زیر ماتریس ۳×۳ زیر را در نظر بگیرید:

$$R = \begin{bmatrix} r_{1x} & r_{2x} & r_{3x} \\ r_{1y} & r_{2y} & r_{3y} \\ r_{1z} & r_{2z} & r_{3z} \end{bmatrix}$$

می دانیم سطرهای ماتریس دوران یکه بوده و اگر در ماتریس دوران ضرب شوند بر روی بردارهای یکۀ محورها قرار می گیرند. بردار یکۀ Rz در راستای P1P2 برداری است که بر روی بردار یکۀ محور z قرار می گیرد. بنابر این:

$$R_z = \begin{bmatrix} r_{1z} & r_{2z} & r_{3z} \end{bmatrix}^T = \frac{P_1P_2}{|P_1P_2|}$$

Rx برداری است که در نهایت پس از دوران بر روی بردار یکۀ محور x قرار می گیرد. این بردار عمود به صفحۀ P1، P2، و P3 است

$$R_x = \begin{bmatrix} r_{1x} & r_{2x} & r_{3x} \end{bmatrix}^T = \frac{P_1P_3 \times P_1P_2}{|P_1P_3 \times P_1P_2|}$$

و نهایتاً

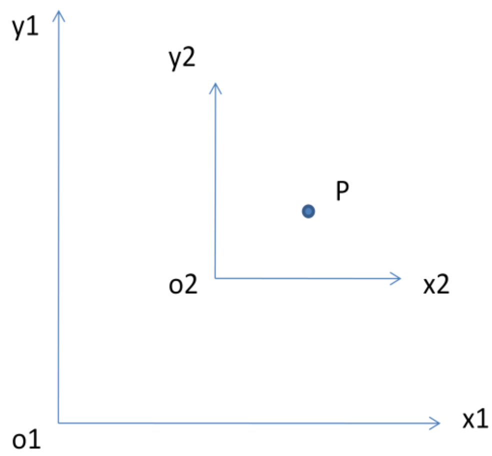
$$R_y = \begin{bmatrix} r_{1y} & r_{2y} & r_{3y} \end{bmatrix}^T = R_z \times R_x$$

ماتریس تبدیل نهائی:

$$\begin{bmatrix} r_{1x} & r_{2x} & r_{3x} & 0 \\ r_{1y} & r_{2y} & r_{3y} & 0 \\ r_{1z} & r_{2z} & r_{3z} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.T(-x_1, -y_1, -z_1)$$

تغییر دستگاه مختصات

تبدیل نقاط در یک دستگاه مختصات را می توان به عنوان تبدیل یک دستگاه مختصات به دیگری نگریست.

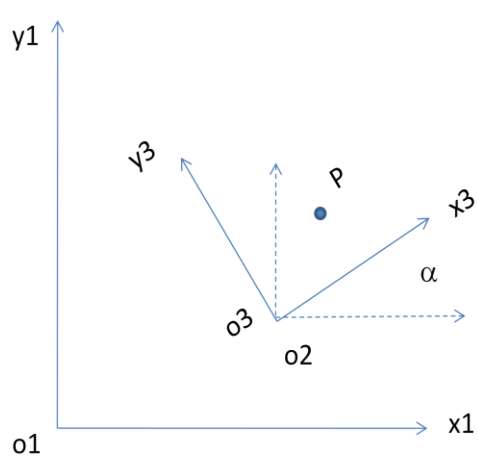


فرض کنید مختصات مبدأ دستگاه دوّم O2 را در دستگاه اوّل می دانیم.

$$^{1}o_{2} = \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

اگر مختصات P را در دستگاه دوّم داشته باشیم، برای بدست آوردن مختصات آن در دستگاه اوّل خواهیم داشت:

$$^{1}P = T(x_1, y_1).^{2}P$$



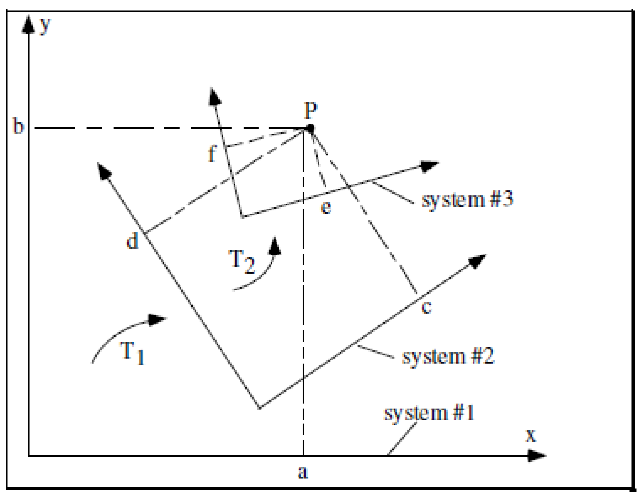
حال اگر دستگاه دوّم را به اندازۀ α حول مبدأ دستگاه دوّم بچرخانیم تا دستگاه سوّم بدست آید و مختصات P را در دستگاه سوّم داشته باشیم. برای بدست آوردن مختصات P در دستگاه دوّم

$$^{2}P = R(\alpha).^{3}P$$

و برای بدست آوردن مختصات P در دستگاه اوّل:

$$^{1}P = T(x_1, y_1).R(\alpha).^{3}P$$

بطور کلی اگر دستگاه دوّم با تبدیل T_1 از دستگاه اوّل با ماتریس تبدیل M_1 بدست آمده باشد و دستگاه سوّم با تبدیل T_2 از دستگاه دوّم با اعمال ماتریس تبدیل M_2 بدست آمده باشد و مختصات P را در دستگاه سوّم داشته باشیم. برای بدست آوردن مختصات P در دستگاه اول خواهیم داشت:

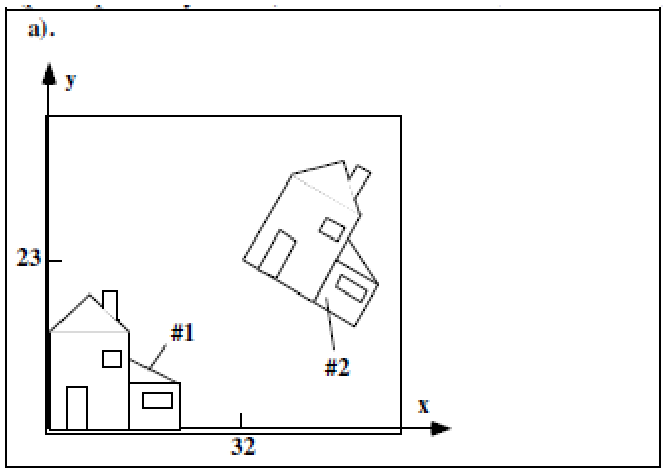


شکل ۱

$$\begin{pmatrix} a \\ b \\ 1 \end{pmatrix} = M_1 \begin{pmatrix} c \\ d \\ 1 \end{pmatrix} = M_1 M_2 \begin{pmatrix} e \\ f \\ 1 \end{pmatrix}$$

دقت به این نکته لازم است که اگر مختصات P را در دستگاه اوّل داشیم و تبدیل M1 و سپس M2 بر روی آن اعمال شده باشد، برای بدست آوردن مختصات جدید نقطه در همان دستگاه اول ضرب در M1 و بعد ضرب در M2 را انجام می دهیم.

تبدیلها در OpenGL



شکل ۲

رسم خانهٔ اوّل

```
glBegin(GL_LINES);
      glVertex2d(V[0].x, V[0].y);
      glVertex2d(V[1].x, V[1].y);
      glVertex2d(V[2].x, V[2].y);
      .... // the remaining points
glEnd();
```

روش اوّل

ساختن ماتریس تبدیل لازم M. نوشتن روال transform2D() برای تبدیل یک نقطهٔ دوبعدی به نقطه ای دیگر

سپس انتقال تک تک نقاط و مجدداً رسم خانه

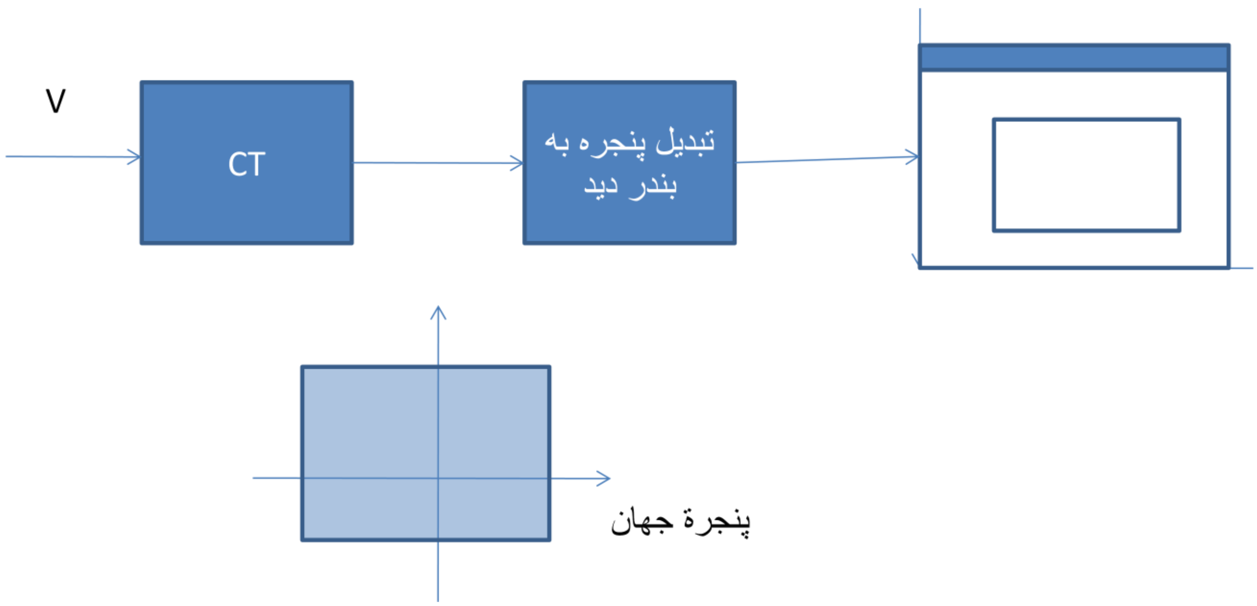
```
glBegin(GL_LINES);
   glVertex2f(transform2D(M, V[0]));
   glVertex2f(transform2D(M, V[1]));
   ..........
glEnd();
```

البته دقت کنید که glVertex2f دو آرگومان می گیرد.

روش دوّم

تا کنون آشنا شدیم که قبل از نمایش هر نقطه بصورت خودکار تبدیل پنجره به بندردید بر روی آن اعمال می شود. می توان تبدیل دیگری به نام تبدیل فعلی (CT) نیز داشت که بصورت خودکار به هر نقطه اعمال شود.

شکل زیر خط لولهٔ ساده شدهٔ OpenGL را نمایش می دهد. هر دستور نمایشی همانند glVertex2d() با آرگومان V که صدا زده شود ابتدا تبدیل CT بر روی آن اعمال شده، سپس تبدیل پنجره به بندر دید انجام شده و نقطه نمایش داده می شود.



شکل ۳

OpenGL یک ماتریس modelview نگاهداری می کند. هر رأس ابتدا در این ماتریس ضرب می شود. کافی است آنرا به مقدار مناسب بنشانیم. OpenGL همواره تبدیلها به صورت سه بعدی انجام می گیرد. در حالت استفاده از تبدیل دو بعدی، مختصهٔ سوّم را صفر استفاده می کنیم. دوران ۲ بعدی، مشابه دوران حول z. در تغییر اندازه sz را همیشه برابر ۱ می گیریم (در حالت ۲ بعدی).

دستورهای تغییر ماتریس modelview عبارتند از glRotate*()، glScale*() و glTranslate*(). به جای * در دستورهای فوق می تواند f یا d قرار گیرد. اگر f قرار گیرد آرگومانها از نوع float و اگر d قرار گیرد آرگومانها از نوع double خواهند بود. این دستورها مستقیم CT را نمی نشانند، بلکه هر یک بوسیلهٔ CT پس ضرب می شود. CT=CT*M

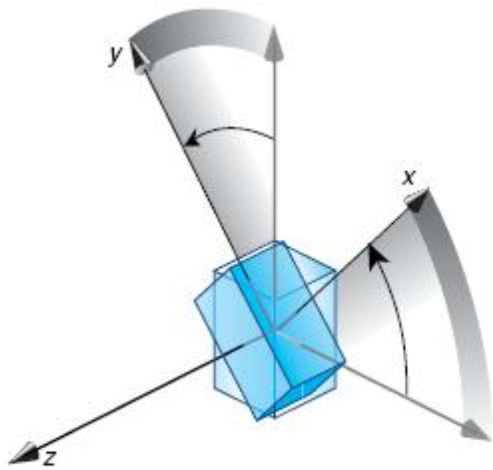
آرگومانهای این دستورها عبارتند از:

glScale*(sx, sy, sz)

glTranslate*(dx, dy, dz)

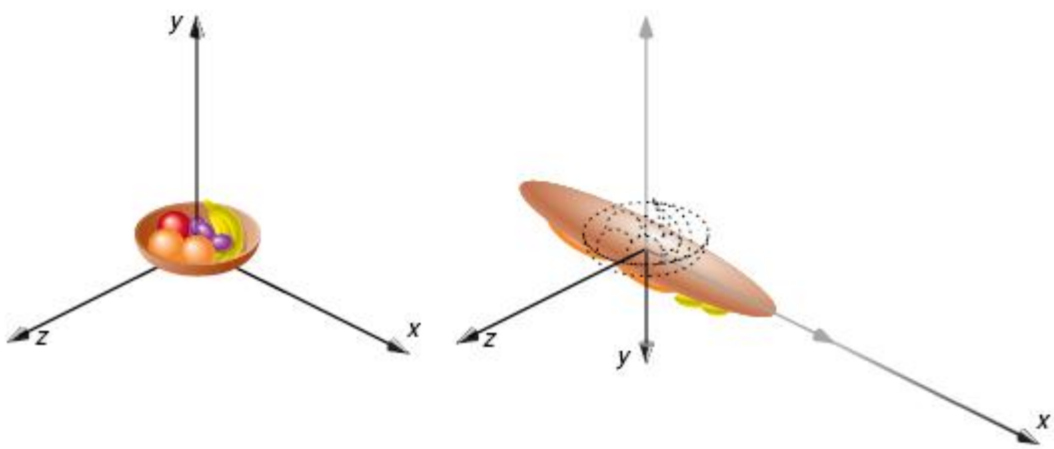
glRotate*(angle,x,y,z) چرخش به اندازهٔ angle درجه حول محور مشخص شده بوسیلهٔ بردار (x,y,z).

شکل زیر اثر دوران یک شیء به اندازهٔ ۴۵ درجه حول محور z را با اجرای دستور glRotatef(45.0, 0.0, 0.0, 1.0) نشان می دهد.



شکل ٤ [Shriener 2010]

شکل زیر نیز اثر اعمال تغییر اندازهٔ و انعکاس را نشان می دهد.



شکل ٥ تغییر اندازه و انعکاس [Shriener 2010]

برای مقدار دهی اوّلیه به CT از دستور ()glLoadIdentity استفاده می شود. این دستور باعث می شود که CT به یک ماتریس همانی (قطر اصلی ۱ و سایر عناصر صفر) مقدار دهی اوّلیه شود. چون این دستور می تواند به هر یک از ماتریسهائی که OpenGL پشتیبانی می کند اعمال شود باید به OpenGL اطلاع دهیم که چه ماتریسی را تغییر می دهیم. این کار توسط دستور glMatrixMode(GL_MODELVIEW) انجام می گیرد.

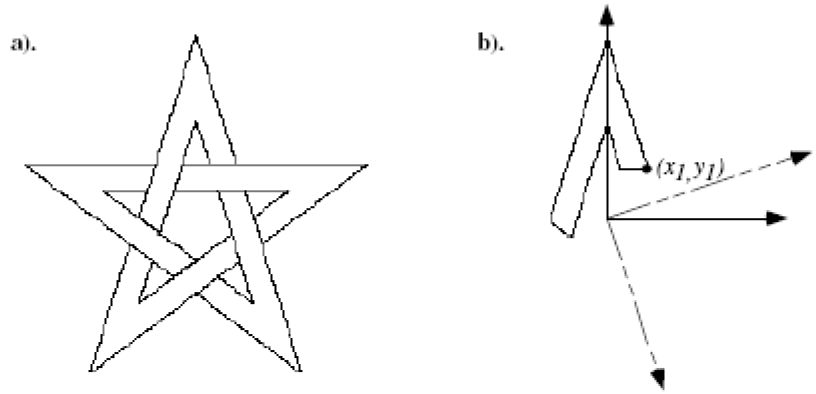
.

```
gluOrtho2D              //نشاندن پنجره
glViewport(....)         // نشاندن بندر دید
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
house();
glTranslated(20, 30,0);
glRotated(-30.0, 0.0, 0.0, 1.0);
house();
```

در برنامهٔ فوق ابتدا پنجرهٔ جهان و بندردید را معین کرده و CT را مقدار دهی اوّلیه می کنیم. سپس خانه را رسم می کنیم. خانهٔ دوّم نسبت به دستگاه مختصات خودش رسم می شود. این دستگاه نسبت به دستگاه مختصات جهان انتقال یافته و سپس ۳۰- درجه دوران یافته است. بنابر این برای رسم هر نقطهٔ آن خانه لازم است ابتدا یک دوران و سپس یک انتقال به آن انجام شود تا مختصات آن نقطه در پنجرهٔ جهان مشخص شود.

مثال: رسم یک ستاره

چنانچه به شکل ستارهٔ نشان داد ه شده توجه شود مشخص خواهد شد که این ستاره از پنج بار تکرار یک شاخهٔ آن بوجود آمده است. بنابر این می توان روالی به نام ()starMotif داشت که یک شاخه را رسم می نماید. سپس هر بار آن را ۷۲ درجه (یک پنچم دایره) دوران داده و دوباره آن را رسم می کنیم.

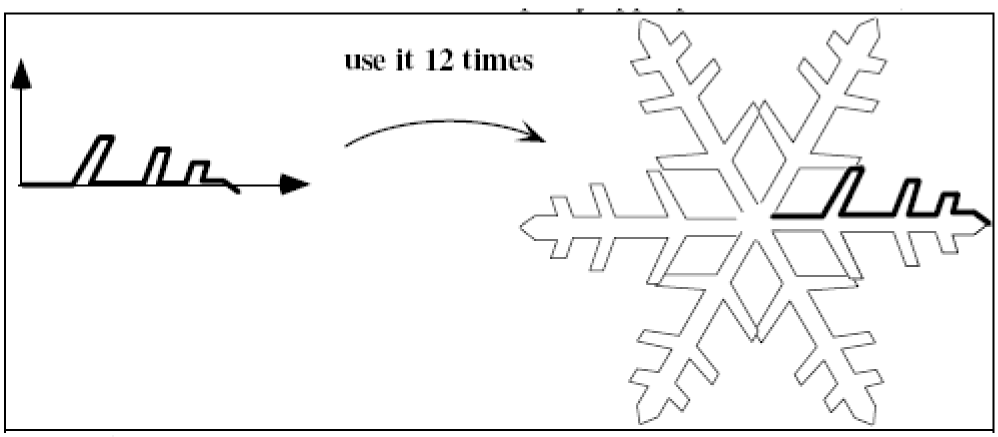


شکل ٦

```
for (int i=0; i<5; i++){
        starMotif();
       glRotated(72.0, 0.0, 0.0, 1.0);
}
```

با هر بار فراخوانی دستور glRotated() زاویهٔ فعلی بر روی زوایای قبلی انباشته می شود.

مثال: رسم یک دانه برف



شکل ٧

یا دانه برف نیز از دوازده بار تکرار نیمی از یک شاخهٔ برف تشکیل شده است. هر شاخه خود از نیمهٔ بالائی و انعکاس آن بوجود آمده است.

رسم یک شاخه برف

```
flakeMotif();
glScaled(1.0, -1.0, 1.0);
flakeMotif();
glScaled(1.0, -1.0, 1.0);
glRotated(60.0);
```

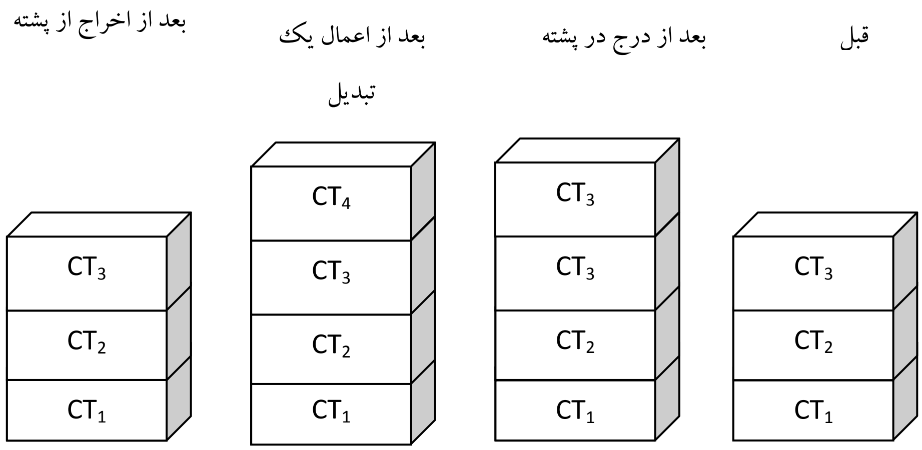
رسم یک دانه برف

```
void drawFlake(){
    for (int i=0; i<6; i++){
        flakeMotif();
        glScaled(1.0, -1.0, 1.0);
        flakeMotif();
        glScaled(1.0, -1.0, 1.0);
        glRotated(60.0);
    }
}
```

ذخیرهٔ CT

یک برنامه ممکن است دارای دبناله های متنوعی از تبدیلها باشد. گاهی لازم می شود که این دنباله ها را کنار گذارده و با دنبالهٔ دیگری به کار ادامه دهیم و پس از اتمام آن عمل مجدداً به همان دنباله ها بازگردیم. برای این کار لازم است که دنبالهٔ اوّلیه را در مکان مناسبی ذخیره و در موقع مناسب آن را بازیابی نمائیم. ممکن است حتی مجموعه ای از دنباله های قبلی را ذخیره نموده و با دنبالهٔ خاصی از آنها به کار ادامه دهیم.

برای این کار می توان از پشته ای از تبدیلها استفاده نمود. ماتریسی که در بالای پشته قرار دارد CT عملی است و دستورهای تبدیل صادر شده بر روی آن اعمال می شود.



شکل ۸ پشته ای از CTها [Hill and Kelley 2007].

glPushMatrix() جهت درج در پشته

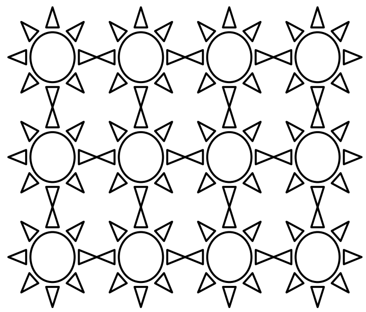
glPopMatrix() جهت اخراج از پشته

```
void    pushCT(void){
        glMatrixMode(GL_MODELVIEW);
        glPushMatrix();
}
```

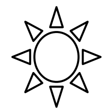
```
void    popCT(void){
        glMatrixMode(GL_MODELVIEW);
        glPopMatrix();
}
```

در OpenGL خارج کردن از پشته ای که شامل فقط یک ماتریس تک باشد خطاست، بنابر این آزمودن تعداد ماتریسهائی که در پشته هستند قبل از اخراج مهم است. دستور glGet(GL_MODELVIEW_STACK_DEPTH) تعداد ماتریسهای باقیمانده در پشته را بازمی گرداند.

مثال: روش دیگری برای کاشیکاری



ب



الف

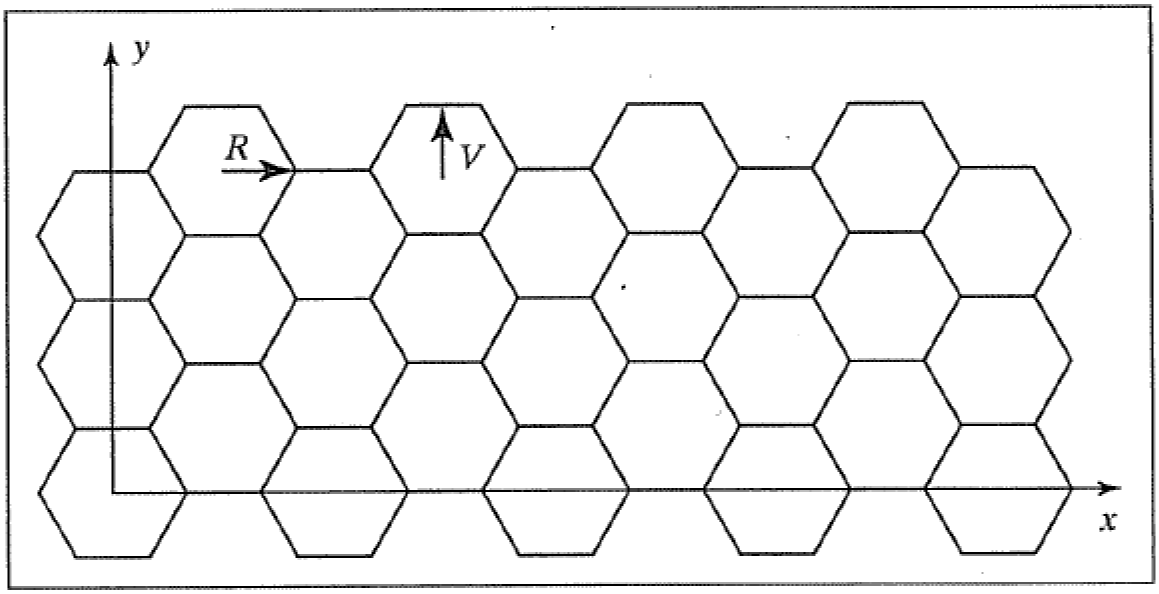
شکل ۹

```
pushCT();
glTranslated(W, H, 0.0);
for (int row=0; row < 3; row++){
   pushCT();
   for (col=0; col<4; col++){
      drawMotif();
      glTranslated(L, 0.0, 0.0);
     }
     popCT();
     gltranslated(0.0, D, 0.0);
}
popCT();
```

تمرین

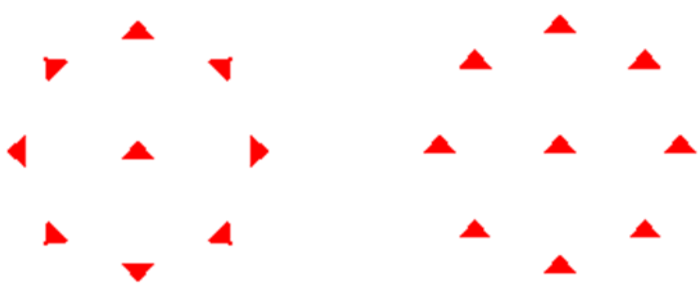
۱- ماتریسهای تبدیل کشش به موازات صفحهٔ yz و zx را بدست آورید.

۲- برنامه ای بنویسید که شکل ۱۰ را ایجاد کند.



شکل ۱۰ یک کاشیکاري شش ضلعي ساده.

۳- برنامه ای بنویسید که اشکال نشان داده شده در شکل ۱۱ را رسم نماید.



شکل ۱۱