OpenGL programming in Visual Basic 6 (Part 2)

By Pooya Shahinfar

PS-Ware team

www.ps-ware.net

Email: info@ps-ware.net

PS-WARE
Persian SoftWARE

فهرست

مقدمه

کتاب الکترونیکی که در پیش رو دارید به برنامه‌نویسی گرافیکی سه بعدی در ویژوال بیسیک به کمک رابط نرم‌افزاری OpenGL اختصاص دارد. در قسمت اول این کتاب الکترونیکی با اصول اولیه کار با OpenGL و همچنین گرافیکهای ساده دو بعدی آشنا شدید، حال نوبت به آن رسیده که شما با شیوه ایجاد اشیاء سه بعدی و تبدیلات در OpenGL آشنا شوید. با مطالعه این قسمت شما خواهید توانست گرافیک‌های پویا و سه بعدی را در OpenGL ایجاد کنید. و در قسمت‌ها و سری‌های آینده سری آموزشی OpenGL با نور پردازی، ایجاد مه و نگاشت بافت در OpenGL آشنا خواهید شد.

در همینجا قبل از شروع آموزش این کتاب الکترونیکی لازم است به تمامی مخاطبین این قسمت متذکر شوم که بهتر است قبل از شروع مطالعه مروری کوتاه بر روی ماتریس‌ها در ریاضیات دبیرستانی و پشته‌ها در درس ساختمان داده داشته باشد. اما در صورتی که شما دانشجو و یا فارغ تحصیل رشته کامپیوتر نیستید و با ساختمان داده‌ها آشنایی ندارید می‌توانید از یادآوری مربوط به پشته‌ها در فصل پایانی کتاب استفاده کنید.

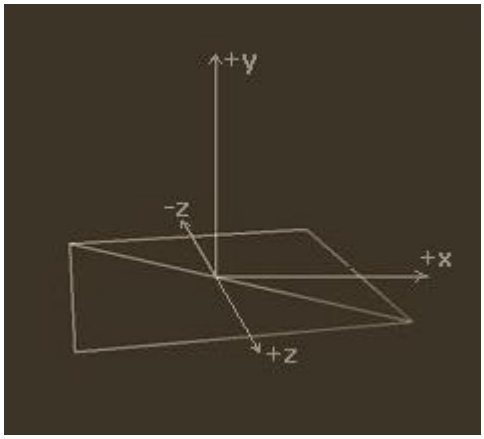
پویا شاهین‌فر
بهار 1386

اصول اولیه گرافیک 3 بعدی

تاکنون اشیاء و اشکالی که می‌ساختید تنها دارای دو بعد بودند. اما همانطور که می‌دانیم هدف از کار با OpenGL طراحی اشیاء و اشکال سه بعدی است. در OpenGL ساخت اشیاء سه بعدی به همان سادگی ساخت اشیاء و اشکال دو بعدی است و از همان قوانین پیروی می‌کند. با این تفاوت که شما باید علاوه بر دادن موقعیت x و y یک راس مکان آن را نیز بر روی محور z ها مشخص کنید. بنابراین ما در ابتدا باید به معرفی محور z ها در OpenGL بپردازیم.

آشنایی با محور Zها

محور zها در OpenGL بر محور xها و yها عمود است و همانند دو محور دیگر از منفی یک تا یک شماره گذاری شده است. در شکل زیر شما می‌توانید موقعیت محور zها را نسبت به دو محور دیگر ببینید.



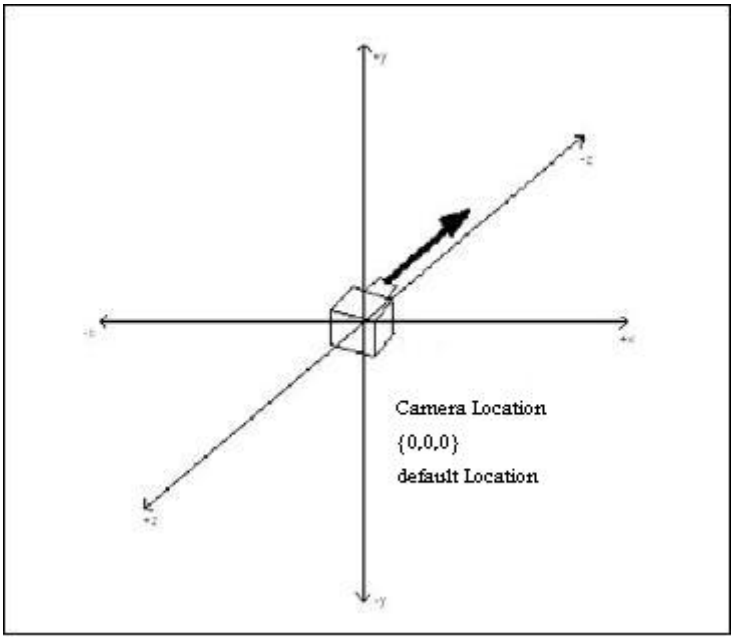
شکل 1- موقعیت محور z نسبت به دو محور دیگر

تمام اجسامی که شما تا کنون آن را به کمک OpenGL رسم می‌کردید و اجسامی که اندازه z را برای آنها تعریف نمی‌کنید OpenGL مقدار صفر را به صورت پیش فرض به آنها می‌دهد.بنابراین در صورتی که می‌خواهید موقعیت z را برای یک راس مشخص کنید به عنوان مثال نقطه را در موقعیت (0,0,.5) قرار دهید باید از تابع glVertex3f() به جای تابع glVertex2f() استفاده کنید. تفاوت این دو تابع تنها دراین است که تابع glVertex3f() دارای 3 پارامتر به جای دو پارامتر است و می‌تواند موقعیت z راس را بگیرد. بنابراین اگر به خواهیم نقطه (.5,.5,.2) را در OpenGL رسم کنیم. باید به شکل زیر عمل کنیم.

```
glBegin bmPoints
        glVertex3f 0.5, 0.5, 0.2
glEnd
```

در صورتی که مقدار z را مثبت بدهید شکل از روی صفحه حذف خواهد شد. دلیل این امر این است که موقعیت پیش‌فرض دوربین در نقطه (0،0،0) است و به سمت منفی محور zها نگاه می‌کند بنابراین

شما نمی توانید اجسامی را که در پشت دوربین قرار دارند را ببینید.شکل زیر بیانگر موقعیت دوربین و سمتی است که به آن نگاه می‌کند.



شکل 2-1 – موقعیت پیش فرض دوربین در OpenGL

بهتر است در هنگام کار سه بعدی در OpenGL به نکات زیر توجه کنید:

- ما به کمک توابع در مثالهای این کتاب الکترونیکی موقیت دوربین را به نقطه (0,0,-1) تغییر دادیم.(این توابع را درفصل آینده بطور کامل توضیح خواهیم داد) بنابراین اگر شما مقدار 1 را به z بدهید متوجه خواهید شد که شکل از روی صفحه محو نخواهد شد.
- با اینکه در دنیای واقعی محور z بیانگر دوری و یا نزدیکی اشکال به شما است به طوری که با دادن مقدار منفی به z باید اشکال کوچکتر شود. اما در OpenGL در حالت پیش فرض این چنین نیست و OpenGL فرض کرده که شما در حال طراحی یک نرم افزار CAD هستید نه یک گیم سه بعدی اکشن بنابراین تمامی اشکال را با مقدارهای متفاوت z به یک اندازه نشان می‌دهد. در فصل بعد به شما شیوه پیاده سازی یک دنیای سه بعدی واقعی بشکلی که اجسام با zهای متفاوت به اندازه‌های متفاوت دیده شوند آشنا خواهیم کرد
- ویژوال بیسیک 6 یا در اصطلاح کتابخانه VBOGL تاحدودی با بعضی از توابع OpenGL مشکل دارد و نمی‌تواند به طور کامل از آنها پشتیبانی کند. اگر می‌خواهید از قدرت واقعی OpenGL بهره ببرید بهتر است از زبان برنامه نویسی C و یا جاوا استفاده کنید.

اجسام سه بعدی آماده در کتابخانه GLUT

همانگونه که پیش از این اشاره کردیم در OpenGL اشکال آماده سه بعدی و یا حتی دو بعدی نظیر کره،دایره و یا مکعب وجود ندارد. ولی شما می‌توانید این اشکال را به کمک توابع موجود در کتابخانه‌های همراه OpenGL نظیر GLU و GLUT ایجاد کنید. (ما در اینجا تنها به شرح کتابخانه GLUT می‌پردازیم)

امتحان کتابخانه GLUT

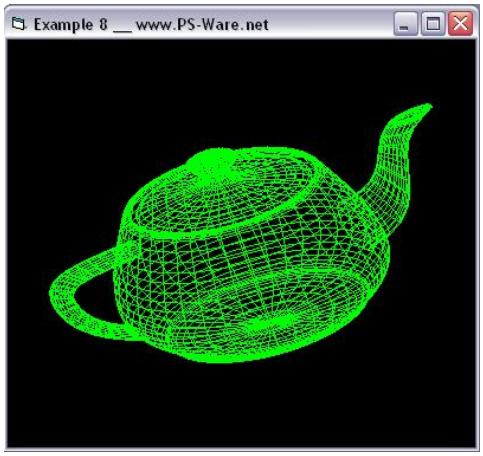
برای اینکه مطمئن شوید کتابخانه GLUT به درستی کار می‌کند، باید از مثال معروف غوری استفاده کنیم . در تمام نرم‌افزارها و رابطهای سه بعدی شکل از پیش تعریف شده‌ای به نام Teapot و یا همان غوری چایی موجود است. اگر کتابخانه و یا نرم افزار موفق شد این غوری را بدرستی رسم کند این

بدان معنا است که آن کتابخانه و یا نرم‌افزار بدرستی کار می‌کند. (البته در رسم اشکال – برای نورپردازی و یا سایه زنی از خرگوش استفاده می‌شود.)

برای رسم یک غوری چایی در GLUT شما باید از دستور glutWireTeapot استفاده کنید.این دستور یک غوری در مرکز صفحه رسم می‌کند و تنها دارای یک پارامتر است که آن اندازه غوری را مشخص می‌کند. برای اینکه سه بعدی بودن غوری را بهتر تشخیص دهید ما آن را به دور محور (1,1,1) به انداازه 1 درجه می‌چرخانیم.پس بنابراین کد برنامه ما به شکل زیر خواهد بود:

```
glColor4f 0, 1, 0, 1
glRotatef 1, 1, 1, 1
glutWireTeapot (0.6)
```

شما می‌تونید مثال مربوط به این کد را از روی مثال شماره 8 این کتاب بر دارید و از آن استفاده کنید. و در شکل زیر نمایی از مثال شماره 8 این کتاب را می‌بینید.



شکل شماره 3-1 – مثال شماره 8 کتاب

اشکال سه بعدی آماده در OpenGL

کتابخانه GLUT شامل یکسری از اشکال دیگر از جمله کره، مکعب و . . . است که آنها بشرح زیر هستند.

کره

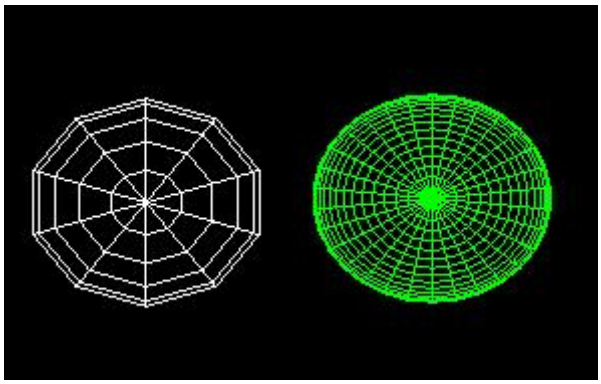
برای رسم یک کره توخالی با طرح سیمی شما باید از دستور glutWireSphere و برای یک کره تو پر glutSolidSphere استفاده کنید. هر دو این دستورها دارای 3 پارامتر هستند که آنها به شرح زیرند:

Radius: بیانگر انداز شعاع کره است و نوع آن GLdouble است.
Slices: تعداد بخش‌های کره بدور محور zها (اگر زمین را یک کره فرض کنیم، این پارامتر برای آن همانند تعداد طولهای جغرافیایی می‌ماند.)
Stacks: تعداد بخش‌های کره در اتداد محور zها (اگر زمین را یک کره فرض کنیم این پارامتر همانند تعداد عرض‌های جغرافیایی برای آن می‌ماند)

در اینجا لازم است به این نکته اشاره کنیم که GLUT برای رسم اشکال سه بعدی مانند کره از چهار ضلعی‌ها استفاده می‌کند و با چسباندن آنها به هم شکل نهایی را می‌سازد شما در حقیقت با دادن مقدار به پارامترهای Slices و Stacks تعداد چهار ضلعی‌ها را مشخص می‌کنید در نتیجه هرچه مقدار

این پارامترها بیشتر باشد شکل نهایی شما به کره واقعی نزدیکتر خواهد بود از طرفی دیگر برنامه شما کندتر خواهد شد.

شما در شکل زیر دو کره را می‌بینید که در کره سفید رنگ مقدار پارامترهای Slices و Stacks برابر با 10 و در کره سبز رنگ برابر با 30 قرار داده شده.



شکل 4-1 – افزایش مقدار پارامترهای Slices و Stacks باعث بهبود نتیجه حاصل شده می‌شود.

مکعب

برای رسم یک مکعب سیمی شما باید از دستور glutWireCube و برای رسم یک مکعب توپر باید از دستور glutSolidCube استفاده کنید که به صورت زیر تعریف شده است.

```
Sub glutSolidCube (size As GLdouble)
```

همانگونه که می‌بینید این دستور تنها یک پارامتر دارد و آن اندازه هر لبه مکعب به صورت اعشاری است.به عنوان مثال دستور زیر یک مکعب در مرکز صفحه می‌کشد که اندازه هر ضلع آن 1 است یا به عبارت دیگر این مکعب نصف صفحه را پر می‌کند.

```
glutWireCube 1
```

در GLUT دستوری برای رسم مکعب مستطیل وجود ندارد اما شما می‌توانید مکعب‌ها را به کمک دستورات مقیاس (در فصل بعد شرح داده خواهد شد) به مکعب مستطیل تبدیل کنید.

مخروط

شما می‌توانید به کمک تابع glutWireCone یک مخروط سیمی و به کمک تابع glutSolidCone یک مخروط توپر رسم کنید. شکل کلی این دو تابع به صورت زیر است.

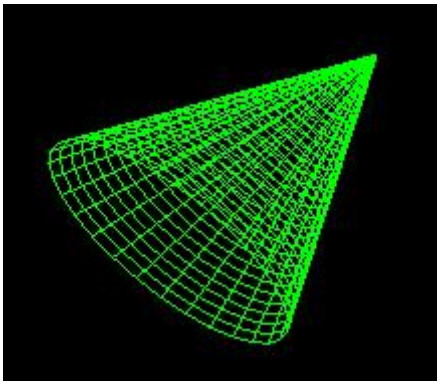
```
Sub glutSolidCone (base As GLdouble, height As GLdouble, slices As GLint, stacks As Glint)
Sub glutWireCone (base As GLdouble, height As GLdouble, slices As GLint, stacks As Glint)
```

همانگونه که می‌بینید هر دوی این دو تابع دارای 4 پارامتر هستند که آها به ترتیب برابر هستند با:

base: شعاع پایین مخروط
Height: ارتفاع مخروط
Slices: تعداد بخش‌های مخروط دور محور zها
Stacks: تعداد بخش‌های مخروط در امتداد محور zها

پارامترهای Slices و Stacks همانند پارمترهای همنام خود در تابع glutWireSphere هستند و این دستور همانند دو دستور دیگر یک مخروط را در مرکز صفحه رسم می‌کند. در زیر شما یک نمونه کد مربوط به رسم مخروط را می‌بینید. که با تایپ کردن آن در تابع DrawGLScene می‌توانید نتیجه‌ای مشابه شکل زیر را بگیرید.(ما در این مثال برای اینکه شما راحتر سه بعدی بودن مخروط را احساس کنید آن را به دور محور (1,1,1) به کمک تابع glRotatef می‌چرخانیم)

```
glColor4f 0, 1, 0, 1
glRotatef 1, 1, 1, 1
glutWireCone 0.4, 0.8, 30, 30
```



شکل 5-1 – نمایی از نتیجه حاصل شده از تکه کد مربوط به رسم مخروط

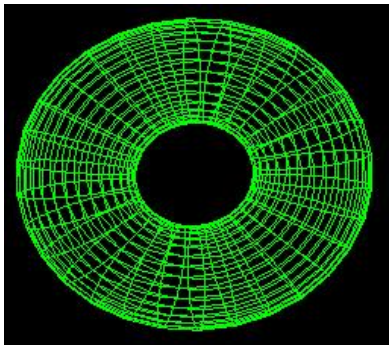
تویوپ و حلقه (torus)

یکی از اشکال آماده دیگر در کتابخانه GLUT حلقه است شما برای رسم یک حلقه سیمی می‌توانید از تابع glutWireTorus و برای رسم یک حلقه توپر می‌توانید از دستور glutSolidTorus استفاده کنید. این دستور دارای چهار پارامتر است که آنها عبارتند از:

innerRadius: شعاع درونی حلقه
outerRadius: شعاع بیرونی حلقه
nsides: تعداد بخش‌ها در هر حلقه
Rings: تعداد حلقه‌های Torus

شما می‌توانید نمایی از تکه کد زیر را که یک حلق را رسم می‌کند در شکل زیر ببینید:

```
glColor4f 0, 1, 0, 1
glRotatef 1, 1, 1, 1
glutWireTorus 0.2, 0.4, 30, 30
```



شکل 6-1 - نمایی از حلقه رسم شده توسط تکه کد مربوط به رسم حلقه

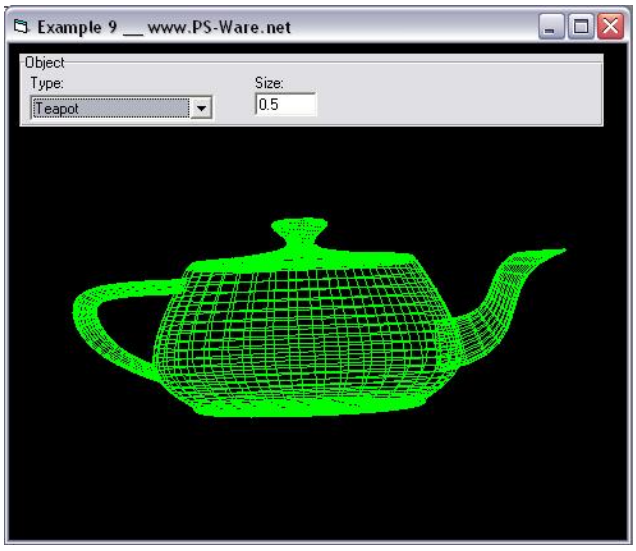
سایر توابع رسم اشکال سه بعدی در GLUT

کتابخانه GLUT شامل یکسری از توابع دیگر برای رسم اشکال سه بعدی است که کاربرد چندانی ندارند و هیچ پارامتری قبول نمی‌کنند. و هنگام فراخوانی آنها اشکال رسم شده توسط آنها تمامی صفحه را پر می‌کند و شما تنها با توابع مربوط به بزرگنمایی می‌توانید آنها را کوچک کنید. شما می‌توانید لیستی از توابع را در جدول زیر به همراه توضیح آنها مشاهده کنید.

نام تابع	توضیح
glutWireDodecahedron	این تابع یک دوازده سطحی سیمی رسم می‌کند.
glutSolidDodecahedron	این تابع یک دوازده سطحی توپر رسم می‌کند.
glutWireOctahedron	این تابع یک هشت سطحی سیمی رسم می‌کند.
glutSolidOctahedron	این تابع یک هشت سطحی توپر رسم می‌کند.
glutWireTetrahedron	این تابع یک چهار ضلعی سیمی را رسم می‌کند.
glutSolidTetrahedron	این تابع یک چهار ضلعی توپر را رسم می‌کند.
glutWireIcosahedron	این تابع یک بیست رویی سیمی را رسم می‌کند.
glutSolidIcosahedron	این تابع بک بیست رویی توپر را رسم می‌کند.

در صورتی شکل سه بعدی را نیاز دارید که آن در کتابخانه GLUT موجود نیست مانند هرم می‌توانید با تغییر دادن پارامترها و یا ترکیب اشکال آن را ایجاد کنید. برای مثال برای ساخت هرم کافیست پارامتر Slices تابع glutWireCone را برابر با چهار قرار دهید.

برای پایان دادن به این فصل از کتاب یک مثال جامع را می‌آوریم که در آن شما می‌توانید در هنگام اجرای برنامه نوع شکلی را که می‌خواهید انتخاب کنید و پارامترهای مربوط به آن را تغییر دهید. سورس کد مربوط به این مثال همراه این کتاب الکترونیکی در داخل پوشه EX9 قرار دارد.



شکل 7-1 – نمایی از مثال شماره 9 کتاب

هم اکنون به پایان فصل اول کتاب رسیدیم . مطالب این فصل بسیار ساده بودند. اما در بعد فصل مطالب بسیار زیاد، پر کاربرد و نسبتا پیچیده را فرا خواهید گرفت. بنابراین به شما پیشنهاد می‌کنم قبل از مطالعه فصل بعد مطالب فصول گذشته را به خوبی فرا بگیرید و آنها را مرور کنید.

تبدیلات و ماتریس‌ها

هم اکنون زمان آن فرا رسیده است که یک وقفه کوتاه در یادگیری شیوه ایجاد اشیاء در جهان ایجاد کنیم و بر روی آموزش چگونگی حرکت اشیاء در پیرامون جهان بپردازیم. این یک جزء بسیار مهم و حیاتی در ساخت بازی‌ها و نرم‌افزارهای گرافیکی کامپیوتری به شمار می‌آید. تا کنون صحنه‌های سه بعدی که شما ایجاد می‌کردید کاملا ایستا،غیر پویا، کسل کننده و غیر قابل محاوره‌ای بودند اما از این به بعد شما با شیوه‌های حرکت دادن،چرخاندن و بزرگ و کوچک کردن اشیاء آشنا خواهید شد. این اعمال هر چند به نظر پیچیده می‌رسند اما OpenGL با استفاده از تبدیلات مختصاتی این کار را برای شما بسیار آسان کرده است.

در طول این فصل شما مطالب زیر را فرا خواهید گرفت:

- تبدیلات ابتدایی مختصات
- تبدیلات دوربین و نمایش
- ماتریس‌ها و پشته ماتریس‌ها در OpenGL
- پردازش تصویر

مفهوم تبدیلات مختصات

برای یک لحظه به پیرامون خود نگاه کنید و یک دوربین عکاسی را بر دارید. هم اکنون با استفاده از دوربینی که در دستان دارید از محیط اطراف خود عکسهای متفاوتی را بگیرید. به عنوان مثال ممکن است شما در محیط اداری کاریتان باشید که در آن دیوارها، کامپیوتر، میز کاری و اشیاء دیگر وجود دارد هر کدام از این اشیاء در دفتر کاری شما شکل و هندسه خاص خود را دارند و از مرکز آنها می‌توان یک محور مختصات رسم کرد که ما در اصطلاح به آن سیستم مختصات محلی می‌گوییم . هر کدام از این سیستم‌های مختصاتی منحصر به فرد هستند و دارای مکان متفاوتی از مرکز جهانی(مرکز اتاق) که در آن واقع هستید دارند. مکان و جهت هرکدام از این اشیاء به این مطلب بستگی دارد که این اشیاء به چه میزان در نزدیکی و یا دورس شما قرار دارند. هنگامی که شما از این اشیاء در حال تهیه عکس هستید، لنز دوربین نیز بر روی تصویر نهایی نیز تاثیر خواهد داشت. دورنمایی (zoom) دوربین می‌تواند اشیاء را از آنچه که هستند بزرگتر یا کوچکتر نشان دهد. هدف کلی شما تصویر رسم شده بر روی فیلم (و یا اگر دوربین شما دیجیتالی باشد بر روی حافظه دوربین) است و می‌دانیم که دوربین و فیلم خود نیز هرکدام دارای خواص خاصی هستند؛ همانند اندازه فیلم و دقت آن که چگونگی و اندازه تصویر نهایی را تعیین می‌کند. بنابراین در حالت کلی می‌توان گفت تصویر نهایی حاصل شده بستگی به مکان اشیاء ، مکان شما ، لنز دورین ، فیلم و تنظیمات دوربین شما دارد.

تبدیلات در OpenGL نیز به همین شکل عمل می‌کنند. آنها به شما اجازه می‌دهند تا اشیاء را در محیط سه بعدی حرکت دهید،بچرخانید و دستکاری کنید. همچنین به شما اجازه می‌دهند تا مختصات اشیاء سه بعدی را برای نمایش دادن آنها بر روی صفحه نمایش به مختصات دو بعدی تبدیل کنید. اگر چه به نظر می‌رسد تبدیلات در OpenGL به صورت مستقیم اشیاء را تغییر می‌دهند اما در حقیقیت آنها مختصات محلی هریک از رئوس را به یک سیستم مختصاتی دیگر تغییر می‌دهند. هنگامی که شما یک راس را بر روی صفحه نمایش می‌خواهید نشان دهید. آن راس بعد از چهار نوع متفاوت از تبدیلات، آماده نمایش بر روی صفحه نمایش می‌شود سپس موتور گرافیکی OpenGL آن را بر روی صفحه ترسیم می‌کند. این تبدیلات عبارتند از:

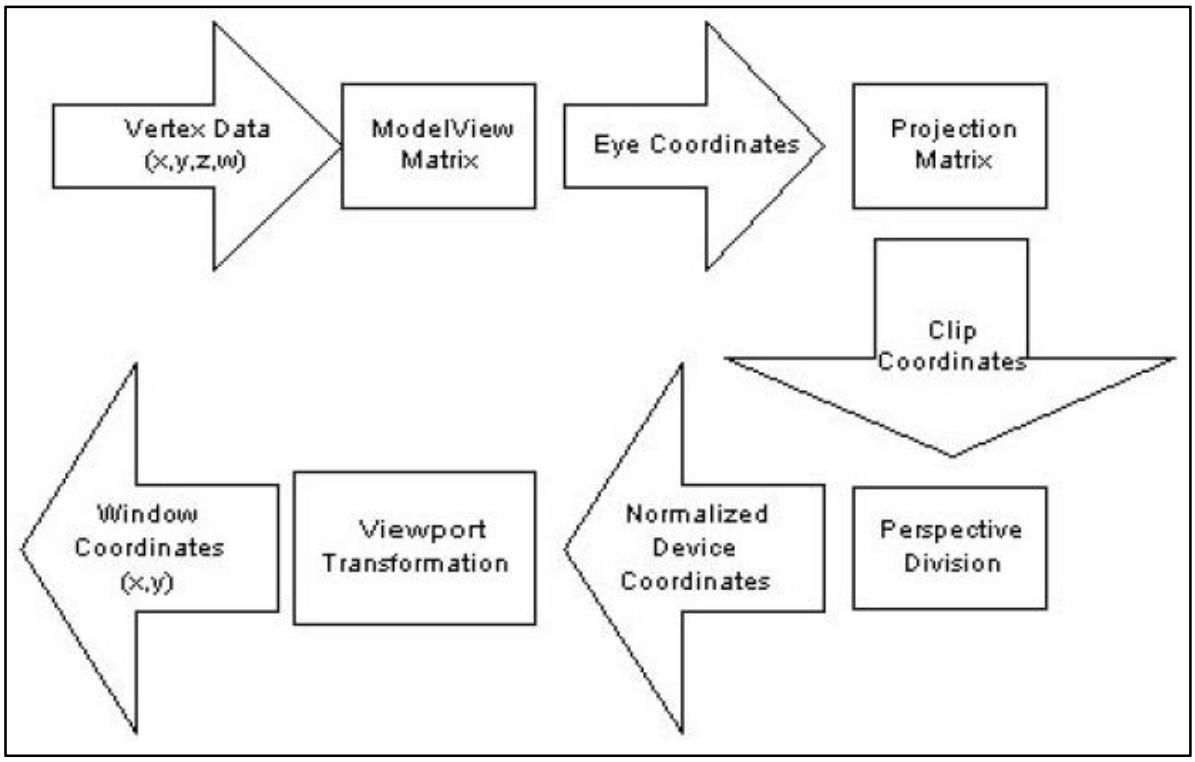
- تبدیلات مدلینگ (Modeling): تبدیلات مدلینگ اشیاء را در پیرامون صحنه حرکت می‌دهند همچنین مختصات محلی را به مختصات پیرامون انتقال می‌دهند.
- تبدیلات نمایشی (Viewing): تبدیلات نمایشی مختصات پیرامون را به مختصات تعیین شده توسط دوربین تبدیل می‌کند.
- تبدیلات تصویری (Projection):تبدیلات تصویری اندازه نمایش و کوتاه‌سازی پنجره را بر روی صفحه نمایش تعیین می‌کنند.
- تبدیلات دریچه دید (Viewport):تبدیلات Viewport مختصات سه‌بعدی را به منظور نمایش بر روی صفحه به مختصات دو بعدی تبدیل می‌کنند.

با اینکه این چهار نوع از تبدیلات در گرافیک سه بعدی هستند اما OpenGL تبدیلات مدلینگ و نمایشی را در یک تبدیلات واحد به نام تبدیلات نمامدل (Modelview) جای داده است. تبدیلات نمامدل را در قسمت ماتریس نمامدل در این فصل شرح خواهیم داد.

جدول زیر این تبدیلات را بطور خلاصه نشان می‌دهد:

نام تبدیلات	توضیحات
Viewing	در گرافیک سه بعدی بیانگر مختصات دوربین است. (جزء تبدیلات استاندارد OpenGL نیست)
Modeling	در گرافیک سه بعدی بیانگر حرکت دستی اشیاء پیرامون صحنه نمایش است.(جزء تبدیلات استاندارد OpenGL نیست)
Projection	نوع نمایش و میزان کوتاه‌سازی اشیاء را تعیین می‌کند.
Viewport	مختصات سه بعدی را به مختصات دو بعدی برای نمایش تبدیل می‌کند.
Modelview	تلفیقی از تبدیلات viewing و modeling است.

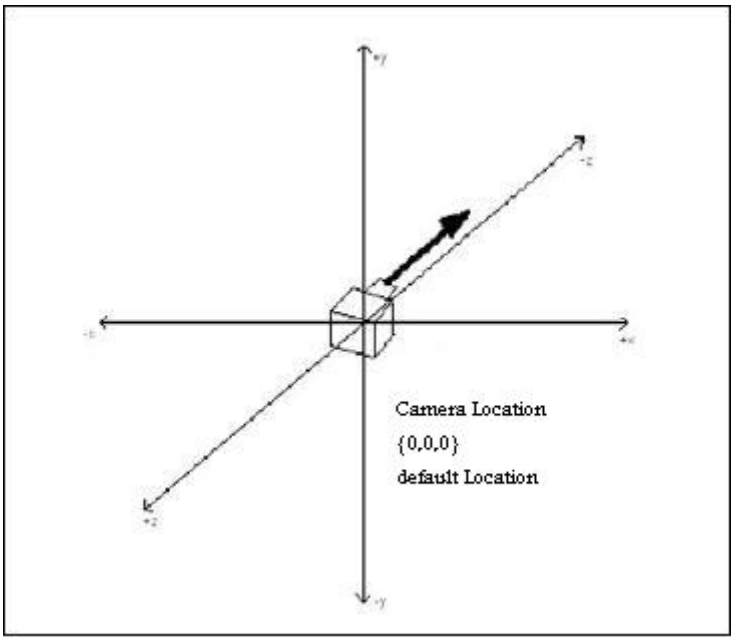
هنگامی که شما یک برنامه سه بعدی را طراحی می‌کنید باید بخاطر داشته باشید که تبدیلات با ترتیب خاصی اعمال می‌شود. و باید این ترتیب را در دستورات خود در هنگام طراحی رعایت کنید. شکل زیر بیانگر ترتیب اعمال تبدیلات در OpenGL است.



شکل شماره 1-2 – ترتیب اعمال تبدیلات در OpenGL

مختصات چشم

یکی مهمترین مفاهیم در تبدیلات و نمایش مفهوم دوربین یا همان مختصات چشم است. در گرافیک سه بعدی، ماتریس که مختصات جهان پیرامون را به مختصات چشم ما تبدیل می‌کند ماتریس نمایش نامیده می‌شود. این ماتریس در حقیقت یا به زبان ساده‌تر موقعیت دوربین و جهتی را که آن نگاه می‌کند تعیین می‌کند. هنگامی که یک شی را با مختصات چشم بیان می‌کنیم در حقیقت رابطه هندسی مابین شی و مختصات دوربین را تعریف می‌کنیم و مختصات شی نسبت به مختصات و فاصله آن با دوربین بیان می‌شود.شما با کمک تبدیلات نمایشی می‌توانید دوربین را در پیرامون جهت سه‌بعدی حرکت دهید. یا به عبارت دیگر همانند این می‌ماند که خود در اتاق در حال حرکت هستید. در OpenGL موقعیت پیش فرض دوربین نقطه (0,0,0) است و دوربین به سمت منفی محور zها نگاه می‌کند. در شکل زیر موقعیت پیش فرض دوربین نشان داده شده است.



شکل 2-2 – موقعیت پیش فرض دوربین

برای پیدا کردن یک دیدگاه بهتر نسبت به جهت گیری دوربین، تصور کنید که در مبدا گفته شده قرار دارید حال 90 درجه به سمت دست چپ خود (دراتداد محور yها) بچرخید در این صورت شما به محور x–ها نگاه خواهید کرد به همین شکل اگر 180 درجه بچرخید شما به محور z+ها نگاه خواهید کرد.

تبدیلات نمایشی

تبدیلات نمایشی برای تعیین موقعیت دوربین و جهت گیری آن استفاده می‌شوند.همانگونه که پیش از این توضیح داده شده موقعیت پیش فرض دوربین در OpenGL نقطه (0,0,0) است و جهت گیری آن به سمت منفی محور zها است.

در هنگام اعمال تبدیلات نمایشی به این مطلب باید توجه داشته باشید که هرگونه تبدیلان نمایشی قبل از تبدیلات مدلینگ باید اعمال شود. این بدین دلیل است که تبدیلات در OpenGL به صورت معکوس اعمال می‌شوند بنابراین با این عمل شما مطمئن خواهید شد که تبدیلات نمایشی بدرستی اعمال خواهد شد.

چگونه باید تبدیلات نمایشی را اعمال کرد؟ در ابتدا شما نیاز دارید تا ماتریس فعلی را پاک کنید (دوربین را به موقعیت پیش فرض ببرید) این عمل را می‌توانید به کمک دستور glLoadIdentity در

OpenGL انجام دهید. این دستور بدون پارامتر است و با فراخوانی آن در OpenGL ماتریش کنونی برابر با مانریس همانی خواهد شد بنابراین موقعیت دوربین به نقطه (0,0,0) خواهد رفت.

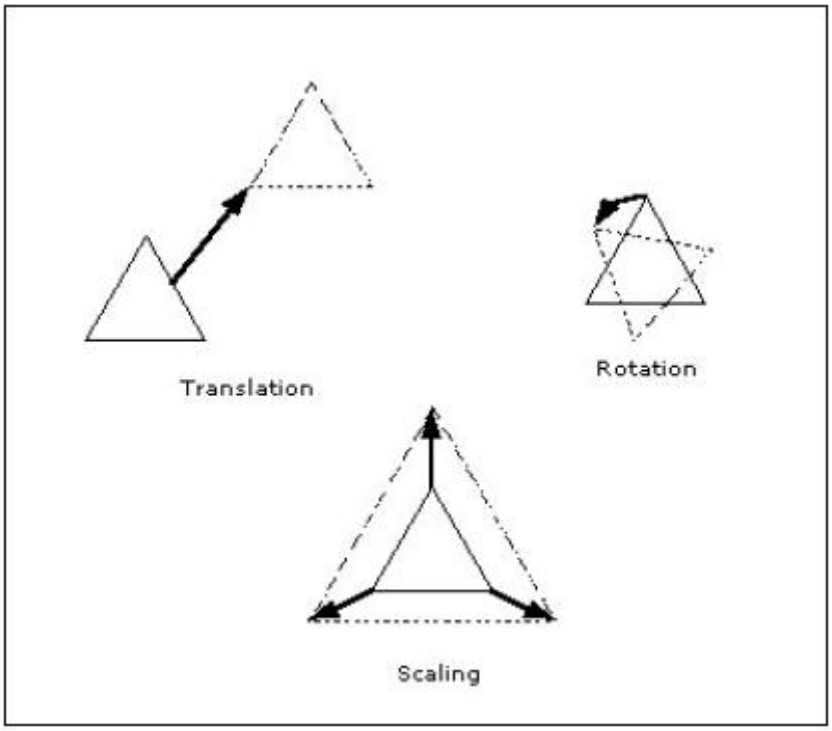
یادآوری: ماتریس همانی ماتریسی است که تمامی عناصر قطر اصلی آن و مابقی عناصر آن صفر است.

برای مقدار دهی و ایجاد تغییر در در ماتریس نمایشی فعلی راههای گوناگونی وجود دارند. یکی از راهها برابر سازی ماتریس نمایش با ماتریس همانی به کمک دستور glLoadIdentity است که در نتیجه موقعیت وجهت گیری دوربین به حالت پیش‌فرض بر می‌گردد و سایر روش‌ها عبارتند از:

- استفاده از دستور gluLookAt برای حرکت دادن و چرخش دوربین در محیط پیرامون. این دستور در ادامه فصل در قسمت شیوه استفاده از دستور gluLookAt شرح داده خواهد شد.
- استفاده از دستورهای چرخش و حرکت انتقالی مدلینگ: دو دستور glRotatef و glTranslate دستوراتی هستند که بدین منظور به کار می‌روند. هرکدام از این دو دستور به شکل کامل در ادامه این فصل در قسمت چرخش و انتقال بطور کامل شرح داده خواهند شد.
- ایجاد شیوه‌ها و روتین‌های خودتان که مجموعه‌ای از دستورهای چرخش و حرکت انتقالی استفاده می‌کند.متاسفانه در ویژوال بیسیک 5 قابلیت استفاده از این روشها را ندارید.

تبدیلات مدلینگ (Modeling)

تبدیلات مدلینگ همانگونه که پیش از این نیز گفته شده به شما اجازه می‌دهند تا یک شی را به کمک حرکت دادن،چرخاندن و بزرگنمایی آن مکان‌دهی و یا جهت دهی کنید. شما می‌توانید هرکدام از این عملیات را به صورت جداگانه و یا به صورت ترکیبی از هم بر روی یک شکل اعمال کنید. شکل زیر نشان دهنده انواع تبدیلات و اثرات آنها بر روی اشیاء است.



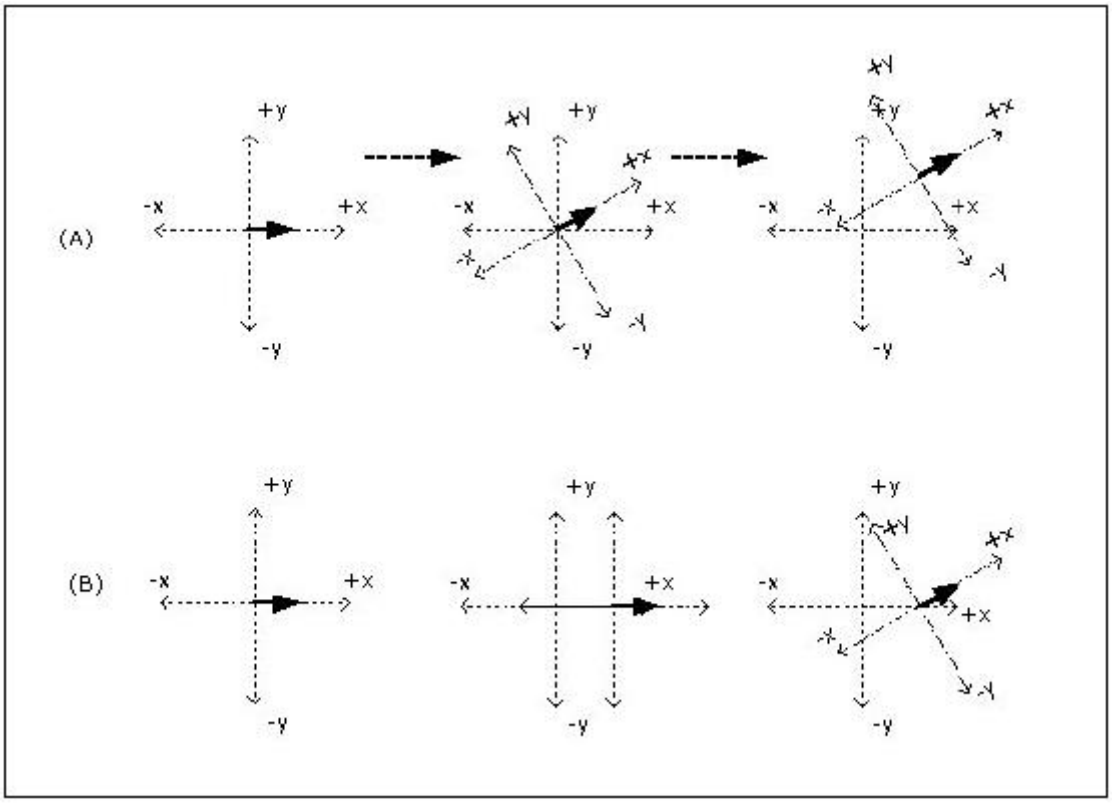
شکل 3-2 – انواع تبدیلات مدلینگ در OpenGL و اثرات آنها بر روی اشکال

Translation (حرکت انتقالی) : این عملیات شامل حرکت دادن یک شی در امتداد یک بردار خاص است.

Rotation (چرخش): این عملیات شامل چرخاندن یک شی به دور یک محور خاص است.

Scaling (مقایس گذاری): به کوچک کردن و یا بزرگ کردن یک شکل می‌گویند و OpenGL با کمک مقیاس دهی هر محور بطور جداگانه به شما این قابلیت را داده که یک شی را بطور غیر یک نواخت منقبض و یا منبسط کنید.

در تبدیلات مدلینگ ترتیب از اهمیت خاصی بر خوردار است به شکل که رعایت نکردن آنها ممکن است منجر به گرفتن نتیجه دور از انتظار شما شود. به مثال زیر دقت کنید.



در حالت Aابتدا شکل راحول محور zها چرخانده‌ایم سپس آن را انتقال داده ایم اما در حالت B ابتدا انتقال داده‌ایم سپس شکل را چرخانده‌ایم.

تبدیلات تصویر

تبدیلات تصویر اندازه نمایش و کوتاه سازی را تعیین می‌کند و بعد از تبدیلات مدلینگ و نمایشی انجام می‌شود.شما می‌توانید تصور کنید که تبدیلات تصویر تصمیم می‌گیرند که کدامیک از اشیاء در صحنه نمایش داده شوند و آنها باید به چه شکلی باشند. این عمل کاملا شبیه انتخاب یک لنز برای دوربین عکاسیتان است. به عنوان مثال در صورتی که شما از یک لنز Wide-angle (عدسی گسترش) یک منطقه بزرگ را بدون جزئیات دقیق آن می‌بینید اما اگر از یک لنز telephoto استفاده کنید می‌توانید اشیائی را که به شما نزدیکتر هستند بزرگتر و با جزئیات ریز آنها ببینید.

رابط گرافیکی OpenGL دارای دو نوع تبدیلات تصویر است که آنها عبارتند از:

- Perspective Projection: این نوع از نمایش دنیای سه بعدی را دقیقا همانگونه که در دنیای واقعی است نمایش می‌دهد به عبارت دیگر فاصله اشیاء تا بیننده باعث بزرگی و کوچکی آنها خواهد شد.
- Orthographic Projection: این نوع نمایش، اشیاء را با همان اندازه دقیقشان بدون توجه به فاصله آنها از هم بر روی صفحه نمایش می‌دهد. این نمونه برای طراحی نرمافزارهای CAD کاربرد دارد و حالت پیش‌فرض OpenGL است.

تبدیلات دریچه دید

آخرین نوع تبدیلات در OpenGL تبدیلات Viewport است. این تبدیلات تصویر 3D نهایی و کوتاه سازی شده توسط ماتریس تصویر را به یک تصویر 2 بعدی تبدیل می‌کند و همچنین اندازه تصویر نهایی را به اندازه پنجره یا اندازه تعیین شده تغییر می‌دهد.

ماتریس‌ها و OpenGL

تا کنون شما در این فصل از کتاب در مورد انواع تبدیلات در OpenGL مطالبی فرا گرفتید. حال اجازه بدهید تا نگاهی به شیوه استفاده از آنها در OpenGL داشته باشیم. تبدیلات در OpenGL وابسته به ماتریس‌ها است تا به کمک آنها و عملیات ریاضی بتواند تبدیلات خواسته شده توسط کاربر را انجام بدهد. همانگونه که شما به زودی خواهید دید OpenGL دارای یک پشته از ماتریس‌ها به نام Matrix stack است، که برای ایجاد اشکال پیچیده به کمک ترکیبی از اشکال ساده‌تر مفید است.(در مورد پشته ماتریس‌ها در ادامه این فصل بحث خو اهیم کرد)

ماتریس نمامدل (Modelview)

ماتریس نمامدل یک ماتریس 4 * 4 است که به کمک آن می‌توانید در اشکال تغییرات را ایجاد کنید. رئوس یک بردار در ماتریس نمامدل ضرب می‌شودند و در نتیجه تغییر شکل پیدا می‌کنند. خود ماتریس مدل نما با ضرب شدن در یک ماتریس 4 * 4 دیگر تغییر پیدا می‌کند.

قبل از فراخوانی هر نوع تبدیلات، شما باید نوع ماتریس کنونی را که از آن می‌خواهید استفاده کنید را تعیین کنید. برای این عمل می‌توانید از دستور glMatrixMode استفاده کنید. این دستور ماتریس کنونی را که شما می‌خواهید از آن استفاده کنید و یا آن را تغییر بدهید را تعیین می‌کند. به عنوان مثال اگر شما بخواهید از تبدیلات انتقالی استفاده کنید (برای نمونه یک مکعب را حرکت دهید و یا آن را بچرخانید) باید از ماتریس نمامدل استفاده کنید پس در ابتدا باید آن را فعال کنید. پس این دستور را به شکل زیر بکار می‌بریم.

```
glMatrixMode mmModelView
```

آرماگون‌هایی که این دستور قبول می‌کند عبارتند از:

- mmProjection: بیانگر ماتریس تصویر است. . (در ادامه این فصل در مورد ماتریس تصویر بحث خواهیم کرد)
- mmModelView: بیانگر ماتریس نما مدل است
- mmTexture: بیانگر ماتریس بافت است.(در قسمت سوم از این مجموعه آموزشی به شرح بافت‌ها خواهیم پرداخت.)

نوع دیگری از ماتریس به نام ماتریس رنگها وجود دارد که برنامه نویسان زبان سی از ثابت GL_COLOR برای آن استفاده می‌کنند، ولی متاسفانه این نوع ماتریس در کتابخانه OpenGL که برای ویژوال بیسیک نوشته شده،تعریف نشده و قابل استفاده نیست.

همیشه در هنگام رسم اشیاء و تصاویر از ابتدا باید ماتریس نمامدل را بازنشانی[1] کنید (بهتر است این عمل را انجام دهید ولی اجباری نیست.) برای بازنشانی ماتریس‌ها در OpenGL از دستور glLoadIdentity استفاده می‌شود این دستور ماتریس کنونی را که به وسیله دستور glMatrixMode انتخاب شده را بازنشانی می‌کند. برای بازنشانی ماتریس نمامدل باید از کد زیر استفاده کنید:

```
glMatrixMode mmModelView
glLoadIdentity
```

[1] Reset

تبدیلاتی که به شما به کمک ماتریس نمامدل انجام می‌گیرند عبارتند از انتقال (Translation)، چرخش (Rotation) و مقیاس (Scaling) در ادامه به شرح هر کدام از این تبدیلات خواهیم پرداخت اما به یاد داشته باشید قبل از اعمال این تغییرات باید حتما ماتریس نمامدل را فعال کنید.

Translation (انتقال)

به کمک انتقال شما می‌توانید یک نقطه را از یک موقعیت به موقعیت دیگر حرکت دهید. در OpenGL این عمل به وسیله دستور glTranslate انجام می‌گیرد که به صورت زیر تعریف شده است:

```
Sub glTranslatef(x As GLfloat, y As GLfloat, z As GLfloat)
و
Sub glTranslated(x As GLdouble, y As GLdouble, z As GLdouble)
```

همانگونه که می‌بینید این تابع به دو شکل تعریف شده است. تابع glTranslatef که ما از آن استفاده خواهیم کرد مقادیر اعشاری را قبول می‌کند و تابع glTranslated مقادیر صحیح را می‌پذیرد از آنجا که تابع glTranslated دارای دقت کافی نیست و نسبت به دریچه دید[2] مقادیر آن تغییر می‌کند ما در مثالهای خود فقط از تابع glTranslatef استفاده خواهیم کرد.

پارامترهای x،y و z نقطه‌ای را که اشکال باید به آن انتقال پیدا کنند را تعیین می‌کنند. برای مثال اگر دستور زیر را اجرا کنید مختصات تمامی نقاطی که بعد از تابع glTranslatef معرفی خواهند شد با مختصات (.3,.4,.2) جمع می‌شوند برای مثال مختصات نقطه (.1,.1,.1) به نقطه (.4,.5,.3) تبدیل خواهد شد.

```
glTranslatef 0.3,0.4,0.2
```

حرکت انتقالی در OpenGL موجب انتقال مرکز سیستم مختصات محلی (نقطه 0و0و0) به نقطه تعیین شده توسط تابع glTranslate می‌گردد. این امر بیانگر این مطلب است که اگر شما از بار دیگر از دستور glTranslate استفاده کنید مقادیر تعیین شده توسط تابع با تابع glTranslate قبل از آن جمع می‌شود و سیستم مختصات محلی به نقطه جمع شده انتقال می‌یابد به عنوان مثال در تکه کد زیر سیستم مختصات محلی ابتدا به نقطه (.2,.3,0) و سپس به نقطه (.1,.1,.1) انتقال می‌یابد

```
glTranslatef 0.2,0.2,0
glTranslatef -0.1,-0.2,.1
```

اما در مثال زیر سیستم مختصات محلی ابتدا به نقطه (.2,.3,0) و سپس به نقطه (-.1,-.2,.1) انتقال می‌یابد و علت آن استفاده از دستور glLoadIdentity مابین دو دستور است.

```
glTranslatef 0.2,0.2,0
glLoadIdentity
glTranslatef -0.1,-0.2,.1
```

Rotation(چرخش)

یک دیگر تبدیلاتی که به کمک ماتریس نمامدل در OpenGL انجام می‌گیرد چرخش است.به کمک چرخش در OpenGL شما می‌توانید یک شکل را به دور یک محور خاص مانند محور yها بچرخانید.چرخش در OpenGL بوسیله دستور glRotatef انجام می‌گیرد که به شکل زیر تعریف شده:

```
Sub glRotatef (angle As GLfloat, x As GLfloat, y As GLfloat, z As GLfloat)
```

---

[2] Viewport

پارامتر angle در این دستور میزان چرخش را بر حسب درجه در خلاف عقربه‌های ساعت بیان می‌کند و پارامترهای x،y و z هر کدام بیانگر مختصات راس انتهایی محوری است که راس (0,0,0) ابتدای آن است و چرخش به دور آن انجام می‌گیرد. به عنوان مثال برای چرخش به دور محور yها به اندازه 45 درجه در جهت خلاف عقربه‌های ساعت دستور glRotatef را به شکل زیر به کار می‌بریم.

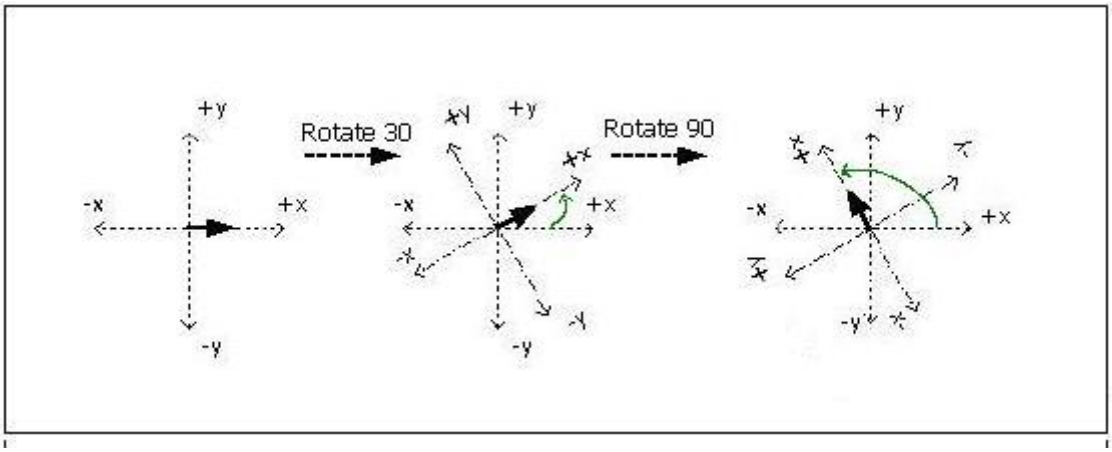
```
glRotatef 45, 0, 1 , 0
```

برای چرخش در جهت عقربه‌های ساعت کافیست مقدار پارامتر angle را منفی بدهید.برای مثال برای چرخش به اندازه 120 درجه در جهت عقربه‌های ساعت از دستور glRotatef به شکل زیر استفاده می‌کنیم.

```
glRotatef -120,0 ,0 ,1
```

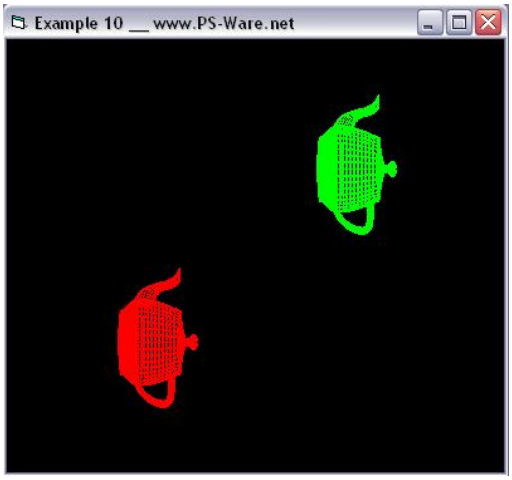
چرخش به دور یک محور خاص بسیار آسان و خوب است اما در بسیاری از موارد اینکار پاسخگوی نیاز شما نیست و شما نیاز به چرخاندن یک شی به دور چند محور دارید به عنوان مثال می‌خواهید یک شی را در ابتدا به اندازه 60 درجه به دور محور xها و سپس به اندازه 40 درجه به دور محور yها بچرخانید برای اینکار شما ممکن است از دو دستور glRotatef پشت سر هم استفاده کنید ولی با امتحان این عمل متوجه می‌شوید که نمی‌توانید نتیجه دلخواه خود را بگیرید علت این امر در این است که شما با چرخش اول سیستم مختصات محلی را در ابتدا به اندازه 60 دجه به دور محور xها می‌چرخانید سپس از همان وضعیت سیستم مختصات محلی را به اندازه 40 درجه می‌چرخانید به عنوان مثال به تکه کد زیر و نتیجه تولید شده توسط آن در شکل زیر دقت کنید.

```
glRotatef 30, 0,0,1
glRotatef 90, 0,0,1
```



همانطور که می‌بینید همانند حرکت‌های انتقالی چرخش‌ها نیز بر روی هم دیگر تاثیر دارند و دلیل این امر این است که هر دوی آنها ماتریس نمامدل را تغییر می‌دهند. بنابراین اگر شما از پشته‌ها استفاده کنید می‌توانید از چرخش های ترکیبی استفاده کنید در ادامه پشته ماتریس‌ها شرح داده خواهد شد.

برای درک بیشتر دو دستور glTranslate و glRotatef می‌توانید مثال شماره 10 همراه این کتاب الکترونیکی (با نام Ex10) را اجرا کنید.در این مثال دو غوری در نقاط (-.4,-.4,0) و (.4,.4,0) رسم می‌شوند که غوری سبز رنگ به دور محور (1,1,0) در جهت خلاف عقربه‌های ساعت و غوری قرمز رنگ در خلاف جهت عقربه‌های ساعت می‌چرخند.شکل زیر نمایی از مثال شماره 10 کتاب را نشان می‌دهد.



شکل 6-2 – نمایی از مثال شماره 10

مقیاس (Scaling)

در بسیاری از تعاریف به بزرگ کردن و یا کوچک کردن یک شی و یا سیستم مختصاتی، مقیاس گفته می‌شود. به عبارت دیگر هنگامی که از مقیاس استفاده می‌کنید، مختصات رئوس یک شی را در یک ضریب دلخواه ضرب و یا تقسیم می‌کنید. به عنوان مثال نقطه‌ای که در مکان (1,1,1) قرار دارد به کمک مقیاس با ضریب 2 به مکان (2,2,2) منتقل می‌شود.حال فرض کنید که یک پاره خط دارید که یک سر آن نقطه (1,1,1) و سر دیگر آن نقطه (-1,-1,-1) است بنابراین اگر این دو نقطه را در 2 ضرب کنید طول پاره خط شما دو برابر خواهد شد. مقیاس در OpenGL به کمک تابع glScale انجام می‌گیرد. این تابع دارای سه پارامتر است که مشخص کننده میزان مقیاس در سه محور مختصاتی (x, y, z) است. تابع glScale در OpenGL به دوشکل زیر تعریف شده که ما تنها از تابع glScalef در مثالهایمان استفاده خواهیم کرد.

```
Sub glScalef(x As GLfloat, y As GLfloat, z As GLfloat)
```

و

```
Sub glScaled(x As GLdouble, y As GLdouble, z As GLdouble)
```

تفاوت این دو تابع تنها در نوع پارامترهای آن است به شکلی که تابع glScalef پارامترهای اعشاری و تابع glScaled پارامترهای صحیح را قبول می‌کند.

مقادیری که پارامترهای x، y و z تعیین می‌کنند بیانگر مقدار ضریب مقیاس در هر یک از محورها است. به هنوان مثال اگر می‌خواهید اندازه کلیه اجسام دو برابر شود می‌توانید تابع glScalef را به شکل زیر به کا ببرید.

```
glScalef 2, 2, 2
```

برای کوچک کردن اشیاء در OpenGL می‌توانید از تابع glScalef با مقادیر کمتر از یک استفاده کنید. به عنوان مثال دستور زیر اندازه کلیه اشیاء را نصف می‌کند.

```
glScalef 0.5, 0.5, 0.5
```

شما به کمک دادن مقدارهای متفاوت به هریک از پارامترهای تابع می‌توانید انقباض و یا انبساط های ناهمگن را در اشکال ایجاد کنید به عنوان مثال تابع زیر نقطه (1,1,1) را به مکان (1,3,2) انتقال می‌دهد.

glScalef 1, 3, 2

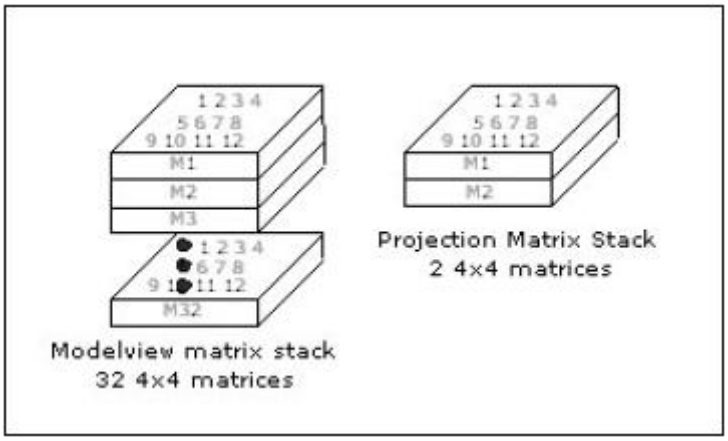
پشته ماتریس‌ها

در OpenGL چهار نمونه متفاوت از پشته ماتریس‌ها موجود است که عبارتند از:

- پشته ماتریس نمامدل (Modelview)
- پشته ماتریس تصویر (Projection)
- پشته ماتریس رنگ‌ها
- پشته ماتریس بافت (Texture)

ماتریس‌هایی را که تا کنون ما در مورد آنها صحبت می‌کردیم و با کمک دستورهای متفاوت در آنها تغییرات ایجاد می‌کردیم در حقیقت ماتریش روین پشته خودشان بودند به عنوان مثال ماتریس نمامدل، ماتریس بالای پشته نمامدل است که ما به آن نام ماتریس نما مدل را می‌دهیم. در کتابخانه VBOGL که ما از آن استفاده می‌کنیم از پشته ماتریس رنگ پشتیبانی نمی‌شود بنابراین ما در ویژوال بیسیک تنها می‌توانیم به پشته‌های نمامدل،تصویر و بافت بپردازیم. پشته نما مدل و تصویر را در همین فصل از کتاب بررسی می‌کنیم اما بررسی پشته ماتریس بافت را به فصل بیت‌مپ‌ها و بافت‌ها موکول می‌کنیم.

اگر در مورد پشته‌ها و پشته‌های ماتریس هیچگونه دیدگاهی ندارید شکل زیر می‌تواند به شما کمک کند.



شکل 7-2 – پشته‌های ماتریس‌های نما مدل و تصویر

به کمک پشته ماتریس‌ها شما می‌توانید تنظیمات هریک از ماتریس‌ها را ذخیره کنید سپس تغییراتی را در ماتریس‌ها ایجاد کنید و بعد از آن دوباره تنضیمات ذخیره شده را بازیابی کنید. به عنوان مثال به کمک پشته ماتریس نما مدل شما می‌توانید یک شی را بدون آنکه اشیاء دیگر به همراه آن شی انتقال پیدا کنند، انتقال دهید.

پشته ماتریس نمامدل در اصل به شما این اجازه را می‌دهد که از یک سیستم مختصاتی به یک سیستم مختصاتی دیگر حرکت کنید. به شکلی که به کمک آن می‌توانید به سیستم مختصاتی قبلی باز گردید.برای مثال اگر ما به سمت نقطه (10,5,7) حرکت کنیم سپس ماتریس نمامدل فعلی را در پشته قرار دهیم آنگاه ما سیستم مختصات محلی را به نقطه (10,5,7) انتقال داده‌ایم و اگر سپس با کمک دستور glTranslatef به نقطه (10,0,0) حرکت کنیم ما در سیستم مختصات محلی در نقطه (10,0,0) قرار داریم اما در مختصات جهانی در نقطه (20,5,7) قرار داریم حال اگر ما ماتریس گذاشته شده بر روی پشته ماتریس نمامدل را برداریم ما به همان موقعیت قبلی خودمان در سیستم مختصات محلی قبلی (یعنی نقطه (10,5,7) – که بعد از انتقال به آن نقطه ماتریس را بر روی پشته قرار دادیم)خواهیم برگشت.

در OpenGL دو دستور glPushMatrix, glPopMatrix برای کار با پشته ماتریس‌ها فراهم شده. دستور glPushMatrix ماتریس کنونی را کپی می‌کند و نسخه کپی شده از آن را بر روی پشته ماتریس کنونی (که توسط تابع glMatrixMode تعیین شده) قرار می‌دهد و دستور glPopMatrix ماتریسی را که در بالای پشته (ماتریس کنونی) قرار دارد را بر می‌دارد و جایگزین ماتریس فعال می‌کند. به عنوان مثال ماتریسی را که در بالای پشته ماتریس نمامدل قرار دارد را بر می‌دارد و بر روی ماتریس نمامدل کنونی کپی می‌کند.

اگر ماتریسهای بسیاری در داخل پشته ماتریس قرار دهید، OpenGL یک خطا را تولید می‌کند.شما می‌توانید این پیغام خطا را با استفاده از دستور glGetError می‌توانید شناسایی و کنترل کنید. اما دانستن این نکته ضروری است که OpenGL همیشه در پشته ماتریس نمامدل حداقل می‌تواند 32 ماتریس را نگهداری کند و در بقیه پشته ماتریسها می‌تواند حداقل دو ماتریس را نگهداری کند.شما می‌توانید بیشترین تعداد ماتریسی را که در زمان اجرای برنامه OpenGL توانایی نگهداری آن را دارد توسط تابع glGetIntegerv با ارسال پارامترهای glgMaxModelViewStackDepth، glgMaxProjectionStackDepth و glgMaxTextureStackDepth بترتیب برای پشته ماتریسهای نمامدل،تصویر و بافت استفاده کرد. مثال زیر بیشترین تعدادی که پشته ماتریس‌های نمامدل می‌تواند نگهداری کند را در درون متغییر i قرار می‌دهد.

```
Dim i As GLint
glGetIntegerv glgMaxModelViewStackDepth, i
```

انواع تبدیلات تصویر

همانگونه که پیش از این اشاره کردیم، دو نوع تبدیل تصویر به نامهای تبدیلات تصویر دورنمایی (perspective) و Orthographic داریم.هدف ما در این بخش از فصل شرح کامل این دو نوع از تبدیلات تصویر و بیان شیوه استفاده از آنها است.

تبدیلات تصویر با دو هدف عمده انجام می‌گیرند:

1. محدود کردن محلی که کاربر توانایی دیدن آن را دارد. این عمل با تعیین سطوح صافی به نام سطوح برش انجام می‌گیرد به شکلی که پردازنده گرافیکی فقط اشکالی را که در داخل محدود تعیین شده توسط سطوح برش قرار دارند، را پردازش می‌کند. بنابراین در اطلاف حافظه و سرعت صرفه جویی می‌کند.
2. هدف دوم تعیین شیوه رسم اشکال است. به شکلی که فاصله اشکال تا بیننده بر روی بزرگی و کوچکی آنها تاثیر داشته باشد، یا نه. در حقیقت این هدف تفاوت مابین دو شیوه نمایش perspective و orthographicاست.

قبل از اعمال هرگونه تبدیلات تصویر، باید از انتخاب شدن ماتریس تصویر (Projection) اطمینان حاصل کنید. برای این عمل بهتر است آن را انتخاب کنید. بنابراین می‌توانید از دستور glMatrixMode با ثابت mmProjection استفاده کنید:

```
glMatrixMode mmProjection
```

بهتر است.بعد از انتخاب نوع ماتریس، تمامی موارد ذخیره شده در آن را پاک کنید تا مقادیر قبلی بر روی مقادیر جدید اعمال شده تاثیر نگذارند.بنابراین در قدم بعدی از تابع glLoadIdentity استفاده می‌کنیم.

```
glLoadIdentity
```

هنگامی که ماتریس تصویر را انتخاب کردید (و احتمالا آن را پاک کردید). شما این آمادگی را دارید که نوع تصویر (Projection) را انتخاب کنید.از آنجا که ما دو نوع تصویر داریم و ساده‌ترین آنها orthographic است، در ابتدا به شرح آن می‌پردازیم و سپس به پرطرفدارترین نوع تصویر یعنی دورنمایی یا همان Projection می‌پردازیم.

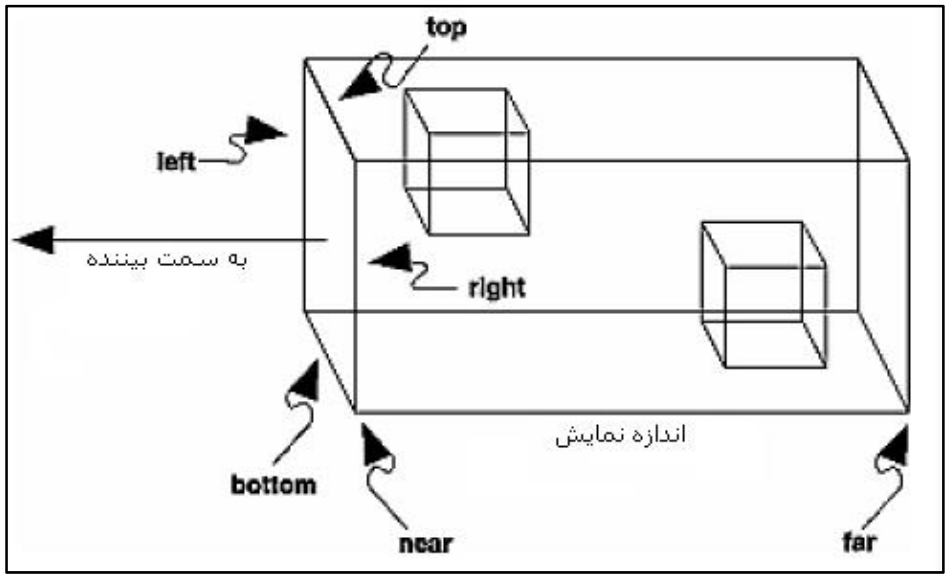
Orthographic Projection

در حالت ارتوگرافیک رابط OpenGL نسبت به دور و یا نزدیک بودن اشیاء بی تفاوت است و تمامی آنها را در همان اندازه اصلی خودشان نشان می‌دهد. به عنوان مثال یک کره با شعاع 0.5 در موقعیت (0،0،0) هم اندازه با یک کره دیگر با همان اندازه در موقعیت (0,0,-4) است. این در صورتی است که در جهان واقعی بینسان نیست و کره دورتر کوچکتر از کره نزدیکتر به نظر می‌آید.این شیوه رسم در برنامه‌های گرافیک سه‌بعدی مانند 3D Studio Max و یا Blender کاربرد دارد و پنجره‌هایی مانند پنجره Front اشکال را به این شکل نشان می‌دهند و یا در نرم افزارهای CAD کاربرد فراوان دارد اما در بازی‌های سه بعدی کامپیوتری معمولا (می‌توان گفت در تمامی موارد) از این شیوه استفاده نمی‌شود. از آنجا که OpenGL یک استاندارد صنعتی برای گرافیک کامپیوتری است این شیوه ترسیم اشکال به صورت پیش فرض انتخاب شده (البته این تنها دلیل انتخاب این شیوه نیست). یکی دیگر از مواردی را که می‌توان برای انتخاب شدن این شیوه به صورت پیش فرض نام برد علت محاسبات کمتر است.

در شیوه ارتوگرافیک OpenGL یک مکعب مستطیل را تعریف می‌کند و تمامی اشکالی را که در درون این مکعب مستطیل قرار گرفته‌اند را رسم می‌کند. برای اینکه محدوده این مکعب مستطیل را تعریف کنیم می‌توانیم از دستور glOrtho استفاده کنیم. در زیر نمایی کلی از این دستور را می‌بینید.

```
Sub glOrtho(left As GLdouble, right As GLdouble, bottom As GLdouble, top As GLdouble,
zNear As GLdouble, zFar As GLdouble)
```

در این دستور پارامترهای left و right بترتیب تعیین کننده سطوح چپ و راست مکعب مستطیل در امتداد محور xها هستند و پارامترهای top و button بیانگر سطوح بالا و پایین در امتداد محور yها. همچنین پارامترهای zFar و zNear بیانگر سطوح دور و نزدیک این مکعب مستطیل در امتداد محور z است. شکل زیر نمایی از این مکعب مستطیل را به همراه پارامترهای دستور glOrtho را نشان می‌دهد.



شکل 2-8 – نمایی از ارتوگرافیک و پارامترهای دستور glOrtho

بدلیل اینکه ارتوگرافیک معمولا برای رسم اشکال دو بعدی به کار گرفته می‌شود. کتابخانه GLU یا OpenGL utility یک تابع دیگر برای تنظیم ارتوگرافیک آماده کرده که در آن پارامترهای far و near که مربوط به فاصله هستند تعریف نشده است. نام این تابع gluOrtho2D است که به صورت زیر در OpenGL تعریف شده:

```
Sub gluOrtho2D (left As GLdouble, right As GLdouble, bottom As GLdouble, top As GLdouble)
```

تمامی پارمترهای تابع gluOrtho2D همانند پارامترهای همنام خود در تابع glOrtho هستند.

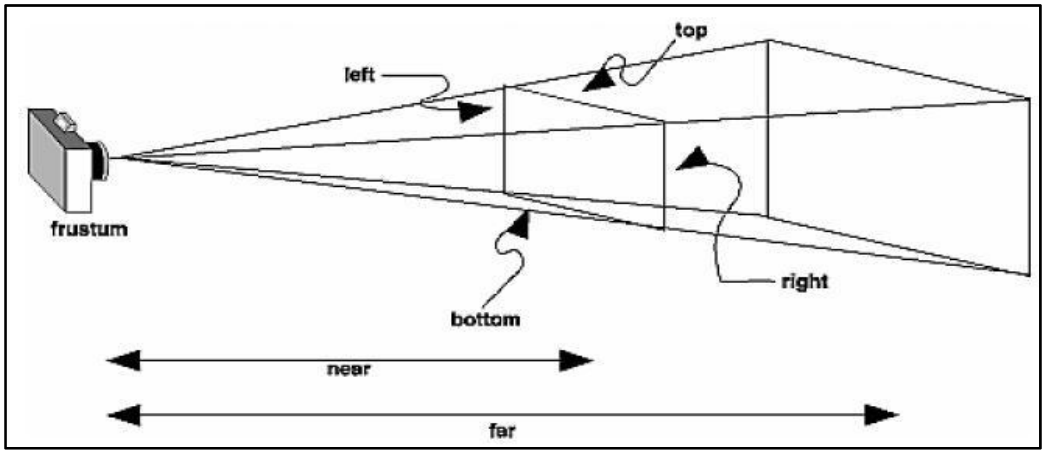
Perspective

با اینکه تبدیلات ارتوگرافیک بسیار آسان،جالب و دقیق به نظر می‌رسند. اما به کمک دورنمایی شما می‌توانید صحنه‌های واقعی‌تری را خلق کنید.در تصاویری که از دورنمایی استفاده می‌کنند اشیائی که از دوربین دورتر هستند، کوچکتر از اندازه واقعیشان به نظر می‌رسند. این افکت معمولا با نام foreshortening یا همان کوتاه نمایی (تجسم اجسام در عمق) شناخته می‌شود.میزان کوچک نمایی اشیاء در حالت perspective هرم ناقص[3] نامیده می‌شود.این هرم همانند یک هرم واقعی است که سر آن بریده شده و روبه‌روی بیننده قرار دارد. رابط OpenGL با تغییر اندازه انتهای کوچکتر هرم به انتهای بزرگتر هرم عمل کوتاهنمایی را انجام می‌دهد به شکلی که اجسامی که به انتهای کوچکتر نزدیک هستند بیشتر از اجسامی که در انتهای بزرگتر هستند انبساط پیدا می‌کنند. در حقیقت OpenGL با عمل کشش دادن و یا فشرده کردن اشیاء را بزرگتر و کوچکتر از حالت عادی نمایش می‌دهد.

در OpenGL دو شیوه برای تعیین پارامترهای این هرم ناقص وجود دارد که با تنظیم کردن پارمترهای آنها شما در حقیقت پارامترهای دورنمایی را تعیین کرده‌اید.اولین شیوه استفاده از تابع glFrustum است که به شکل زیر تعریف شده است:

```
Sub glFrustum(left As GLdouble, right As GLdouble, bottom As GLdouble, top As GLdouble, zNear As GLdouble, zFar As GLdouble)
```

پارامترهای left،Right،Top و bottom تعیین کننده مختصات x و y نزدیکترین سطح برش[4] و پارامترهای near و Far بترتیب فاصله سطوح برش دور ونزدیک را تعیین می‌کنند.شکل زیر بیانگر پارامترهای تابع glFrustum و حالت perspective است.



شکل 9 – 2 –نمایی از perspective و پارامترهای تابع glFrustum

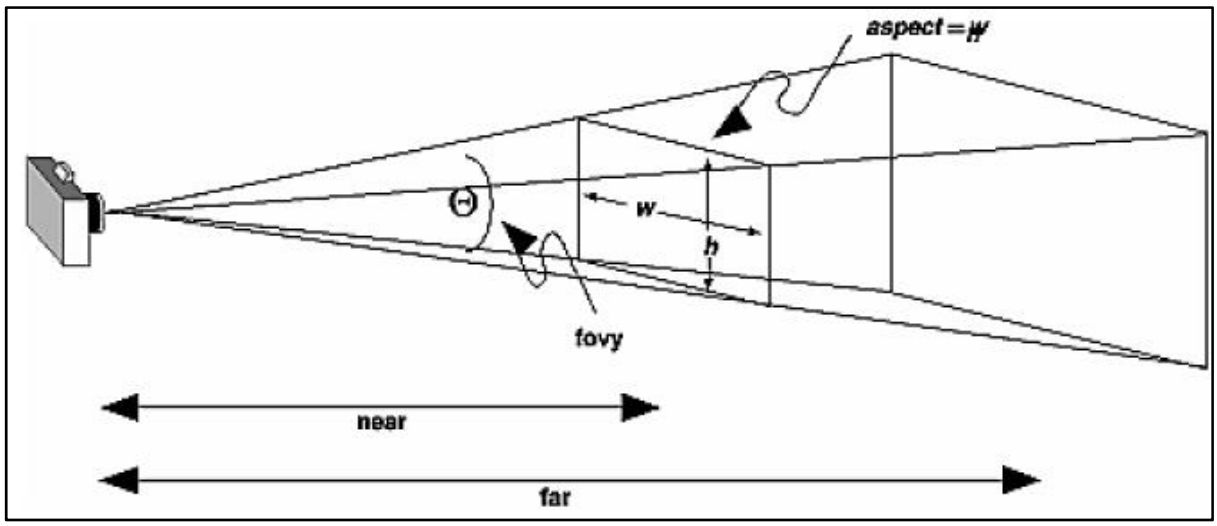
[3] frustum

[4] سطح برش: سوطح برش صفحاتی هستند که OpenGL با تعیین آنها فقط محدوده داخل آنها را نمایش می‌دهد.در حقیقت OpenGL با این عمل از پردازش اضافی اشکالی که در دریچه دید کاربر قرار ندارند ممانعت به عمل می‌آورد.

استفاده از تابع glFrustum به شما این اجازه را می‌دهد که یک هرم نا متقارن ایجاد کنید. که ممکن است در بعضی از موارد مفید باشد ولی استفاده از آن عمومی نیست.برای توضیح بیشتر در مورد اینکه کاربر در هنگام استفاده از هرم چه می‌بیند بطور مستقیم و جزء به جزء قابل درک نیست. در عوض فکر کردن در مورد اندازه نمایش[5] آسان‌تر است.کتابخانه glu یک دستور را فراهم کرده که به شما اجازه می‌دهد به طور مستقیم اندازه نمایش را تعیین کنید.این تابع به صورت زیر تعرف می‌شود که دارای کاربرد فراوان است:

```
Sub gluPerspective (fovy As GLdouble, aspect As GLdouble, zNear As GLdouble, zFar As GLdouble)
```

پارامتر fovy بیانگر زاویه دید بر حسب درجه است (این زاویه در امتداد محور yها محاسبه می‌شود.).پارامتر aspect نسبت تقسیم عرض صفحه نمایش به طول آن است .که این تعیین کننده اندازه دید نمایش در امتداد محور yها است.پارامترهای far و near همانند پارامترهای خود در دستور glFrustum() هستند.



شکل 10 -2 – نمایی از پارمترهای تابع gluPerspective

مقدار 45 تا 90 درجه بهترین مقادیر ممکن برای پارامتر fovy است. به صورتی که اکثر برنامه‌نویسان از این مقدار استفاده می‌کنند.

تنظیم دیچه دید (Viewport)

بعضی از دستورهای projection (تبدیلات تصویر) که آنها را پیش از این شرح داده‌ایم.به طور کامل به اندازه دریچه دید وابسته هستند (برای مثال پارامتر aspect در دستور gluPerspective). همانطور که پیش از این اشاره کردم تبدیلات دریچه دید در OpenGL پس از تبدیلات تصویر رخ می‌دهد.بنابراین هم اکنون که انواع تبدیلات تصویر را بررسی کردیم بهترین زمان برای بررسی تبدیلات دریچه دید است.

به طور ذاتی، دریچه دید .جهت گیری و ایجاد پنجره دوبعدی را که شما در آن پردازش تصویر را انجام می‌دهید تعیین می‌کند و تصاویر نهایی را نمایش می‌دهد.و این تنظیمات به کمک دستور glViewport انجام می‌گیرند:

```
Sub glViewport(x As GLint, y As GLint, width As GLsizei, height As GLsizei)
```

پارامترهای x و y مختصات گوشه نقطه چپ پایین دریچه دید را تعیین می‌کنند و پارامترهای width و height اندازه پنجره را بر حسب پیکسل مشخص می‌کنند.

[5] اندازه نمایشبیانگر پهنا و زاویه‌ای است که کاربر می‌بیند.

هنگامی که شما یک محتوای رندر (context render) را برای اولین با ایجاد می‌کنید و آن را به پنجره خود پیوست می‌دهید، OpenGL به صورت اتوماتیک اندازه دریچه دید را با ابعاد پنجره برنامه شما مطابقت می‌دهد. این عمل برای بسیاری از نرم‌افزارها از جمله بازیها کافی است اما در مواردی خاص می‌خواهید که اندازه دریچه دیدتان را به روز کنید (به عنوان مثال در هنگام تغییر اندازه پنجره – ما از این به روز رسانی در تابع ReSizeGLScene اسفاده کرده‌ایم تا در هنگام تغییر اندازه فرم برنامه اندازه دریچه دید یا به عبارت ساده اندازه اشکال هم به آن نسبت بزرگ و کوچک شوند.). در مواردی دیگر ممکن است بخواهید که محدوده نمایش اشکال را در پنجره(فرم) محدود کنید و یا چندین محدوده متفاوت بسازید (یا همان Multi Viewport) همانند نرم افزارهایی مانند 3D Studio max که دارای 4 دریچه دید متفاوت است.

در بازیهای کامپیوتری معمولا یک دریچه دید کافی است و تنها لازم است با تغییر اندازه پنجره نمایش آن را به روز کنید. برای این عمل ما در مدول OGLEX6 یک تابع با نام ReSizeGLScene نوشتیم. این تابع هنگام تغییر اندازه فرم برنامه فرخوانی می‌شود که مواردی از جمله perspective و دریچه دید را تنظیم می‌کند.

دستکاری محوطه دید (Viewpoint[6])

در طول این فصل ما شیوهای گوناگون دستکاری محوطه دید یا همان دوربین را به شما معرفی می‌کنیم.اولین گزینه استفاده از دستور gluLookAt است که به شما اجازه می‌دهد که مکان محوطه دید را تعیین کنید.در این حالت یک بردار مستقیم از دوربین تا نقطه‌ای که می‌خواهیم مرکز دیدمان به آن متمرکز شود رسم می‌کنیم.گزینه دوم استفاده از دو تابع glTranslatef و تابع glRotatef در کنار هم است که بترتیب برای موقعیت دهی وجهت دهی محوطه دید استفاده می‌شوند.

حال اجازه دهید که بورت تک تک به شرح این گزینه‌ها بپردازیم.

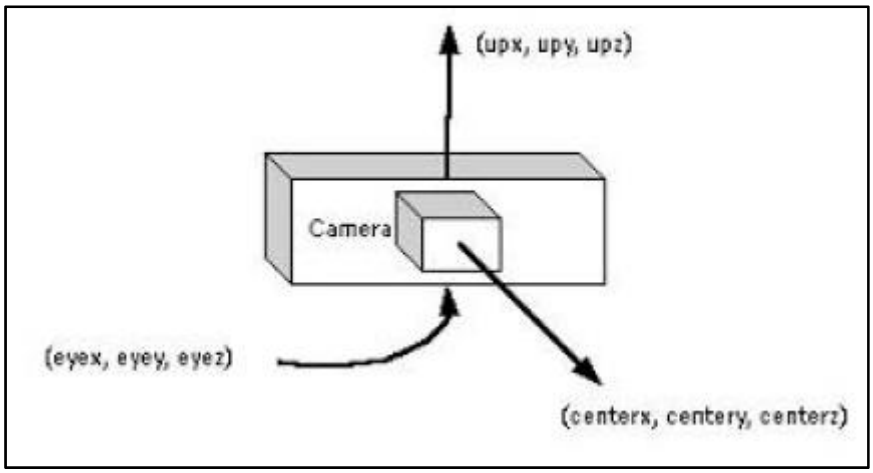
استفاده از دستور gluLookAt

تابع gluLookAt که به صورت زیر تعریف می‌شود به منظور تعیین محل دوربین و جهت گیری آن استفاده می‌شود:

```
Sub gluLookAt(eyex As GLdouble, eyey As GLdouble, eyez As GLdouble, centerx As GLdouble, centery As GLdouble, centerz As GLdouble, upx As GLdouble, upy As GLdouble, upz As GLdouble)
```

یه پارامتر اول این تابع یعنی پارامترهای eyex،eyey و eyez برای تعیین موقعیت دوربین به کار می‌روند (که مقدار پیش فرض آنها، همان مقدار پیش فرض دوربین یا نقطه (0و0و0) است.سه پارامتر دیگر نقطه‌ای را که دوربین به آن نگاه می‌کند را مشخص می‌کنند.(دوربین در حالت پیش فرض به منفی محور zها نگاه می‌کند). و سه پارامتر آخر بیانگر بردار up هستند. بردار Up برداری است که دوربین با توجه به آن در امتداد محور zها می‌چرخد. با بیان ساده‌تر این بردار همان چرخش را تعریف می‌کند ولی به جای استفاده از درجه از بردار استفاده شده که قابلیت‌های فوق العاده زیادی را به شما می‌دهد. مقدار پیش فرض این 3 پارامتر برابر است با بردار که از نقطه (0و0و0) تا نقطه (0و1و0) کشیده شده است. شکل زیر نمایی از پارامترهای gluLookAt را بر روی دوربین نمایش می‌دهد.

[6] هرچند معنی لغوی این کلمه نقطه دید و با نقطه نظر است اما برای مطابقت معنای این کلمه با محتوای آن در گرافیک از لغت محوطه دید استفاده خواهیم کرد.



شکل 2-11 – نمایی از تابع gluLookAt

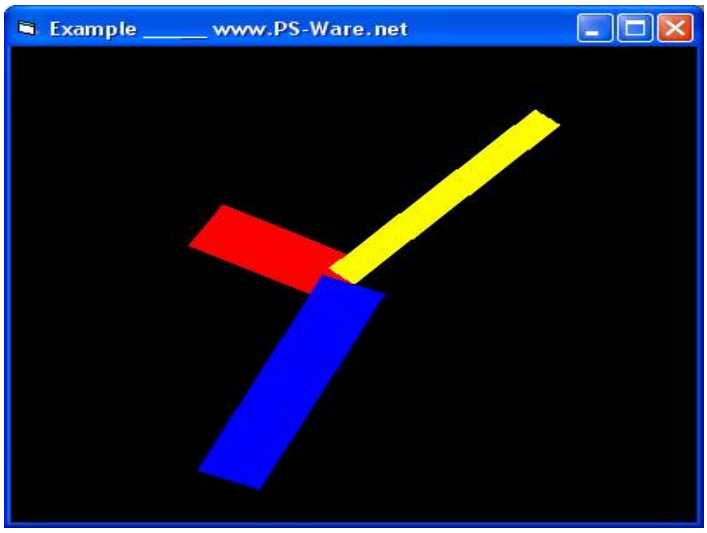
استفاده از توابع glRotate و glTranslate

یکی از موارد استفاده از توابع glTranslate و glRotate هنگامی است که شما نمی‌خواهید برای تغییر مکان و جهت گیری دوربین از تابع gluLookAt استفاده کنید. در این حالت شما به جای تغییر موقعیت دوربین، موقعیت اشیاء را نسبت به دوربین تغییر می‌دهید به عنوان مثال در یک بازی به جای تغییر موقعیت دوربین در هنگام راه رفتن کاراکتر بازی، می‌توانید موقعیت اشیاء را نسبت به دوربین تغییر دهید تا کاراکتر بازی متحرک به نظر برسد مثلا به در عوض هر قدم یک بار تابع glTranslatef را اجرا کنید.

در فصل بعد این قسمت از سری آموزشی برنامه‌نویسی OpenGL در ویژوال بیسیک این روش‌ها را با مثالهای عملی بررسی خواهیم کرد.

مثال پایان فصل

در این مثال قصد داریم یک ساعت به کمک توابع مدلینگ و پشته ماتریس نما مدل پیاده سازی کنیم.البته ساعتی که ما در این قسمت آن را پیاده سازی می‌کنیم دارای ظاهر چندان زیبایی نیست. اما ما این ساعت را درفصل نگاشت بافت (قسمت سوم) با قرار دادن تکتچرها(بافتها – به عبارت دیگر عکسها) بر روی عقربه‌های ساعت و پس زمینه آن به یک ساعت واقعی‌تر تبدیل خواهیم کرد. در این قسمت از فصل ما به شرح چگونگی کار توابع این مثال خواهیم پرداخت.



شکل 2-12 – نمایی از مثال شماره 12

در ابتدا در فرم اصلی یک برنامه یک Timer قرار دادیم و خاصیت Interval آن را به 500 تنظیم کردیم تا در هر نیم ثانیه یک تیک بزند. با هر بار فراخوانی تابع Timer1_Timer() میزان درجه‌ای که هریک از عقربه‌ها باید طی کنند تا به نقطه خودشان برسند محاسبه می‌شود. متغییرهای deg_s و deg_m و deg_h متغییرهای عمومی بوده که در ماژول OGLUtils تعریف شده‌اند.

در تابع DrawGLScene ما برای رسم هر عقربه یکبار پشته ماتریس را ذخیره کرده‌ایم و سپس دستور glRotatef را برای چرخاندن مختصات بردار محلی استفاده کریم و سپس از آن به رسم عقربه پرداخته‌ایم و به کمک دستور glPopMatrix ماتریس ذخیره شده در پشته ماتریس را برداشته‌ایم. علت اینکه از پشته ماتریس نما مدل استفاده کردایم این است که چرخش حاصل از تابع glRotatef بر روی سایر عقربه‌ها تاثیر نگذارد.

این مثال در عین ساده بودن بیانگر مفاهیم بسیار زیادی است که بیشتر هدف ما را از این فصل پوشش می‌دهد. همچنین اگر قبل از این سورس کدهای مربوط به ساعت‌های آنالوگ را در ویژوال بیسیک دیده باشید متوجه خواهید شد که آنها از محاسبت پیچیده مثلثاتی برای رسم عقربه‌های استفاده کرده‌اند. این در صورتی است که ما به کمک پشته ماتریسها و تبدیلات مدلینگ اینکار را بسیار آسان‌تر انجام داده‌ایم. شما راحتی استفاده از ماتریسها را بجای این توابع مثلثاتی هنگامی که از بافت‌ها برای تزئیین کردن عقربه‌ها استفاده می‌کنیم را بهتر متوجه خواهید شد. این در صورتی است که با توابع مثلثاتی انجام اینکار تقریبا غیر ممکن خواهد بود.

راهنمایی‌ها و مشکلات برنامه‌نویسی

پشته چیست؟

اکثر برنامه‌نویسان ویژوال بیسیک از جمله برنامه‌نویسان تازه‌کار با مبحثی با نام ساختمان داده‌ها آشنایی ندارد.علت این امر نا مفهوم بدون این مبحث در ویژوال بیسیک است.ساختمان داده‌ها مبحثی است که به شیوه ذخیره و بازیابی داده‌ها در کامپیوتر و شیوه بررسی الگوریتم‌ها در کامپیوتر اختصاص دارد. پشته یکی از این شیوه‌ها است که برای ذخیره و بازیابی دادههای یک برنامه به کار می‌رود. برای اینکه بتوانید پشته‌ها را در نظر بگیرید بهتر است به مثال شستن ظروف آشپزخانه بپردازیم.بنابراین فرض کنید که داده‌ها شما ظروف هستند و شما پروسس پاک کردن آنها را انجام می‌هید. در این حالت شما در ابتدا یک ظرف دارید اما قبل از پاک کردن آن ظرف دیگری بر روی آن قرار می‌دهید و این عمل را همچنان تا گذاشتن 10 ظرف غذا خوری بر روی هم انجام می‌دهید. مساله‌ای که در اینجا مطرح می‌شود این است که ظرفی که آخر وارد شده و بر روی ظروف دیگر قرار گرفته زودتر شسته می‌شود. و همچنان این روال ادامه دارد اگر در وسط کار ظرف A را نیز بر روی ظروف شسته نشده قرار دهید ظرف A را زودتر خواهید شست. و بعد از شستن آن به سراغ ظروف بعدی خواهید رفت. به این روش قرار دادن داده‌ها و پردازش، پشته گفته می‌شود که خلاف یک صف نانوایی است (یعنی آنکه زودتر وارد می‌شود، زودتر هم خارج می‌شود.) یا در اصطلاح آنکه دیرتر وارد شده زدوتر خارج می‌شود. در زندگی روزمره می‌توان مثالهای بسیاری را از پشته یافت.هر برنامه کامپیوتری که شما می‌نویسید دارای یک پشته است که در داخل این پشته آدرس زیرروال‌های (فانکشن) خوانده شده تو در تو ذخیره می‌گردد که پس از پایان هر کدام برنامه به زیر روال قبلی که زیر روال به پایان رسیده آن را فراخوانی کرده بود مراجعه کند.

برای نگهداری پشته‌ها می‌توان از یک آرایه با دو متغییر استفاده کرد. آرایه برای نگهداری آیتمها یا همان داده‌ها استفاده می‌شود. که به عنوان مثال به صورت زیر تعریف می‌شود:

```
Dim Stack (10) AS String
```

این آرایه تنها می‌تواند 10 داده از نوع رشته‌ای بپذیرد بنابراین شما یک پشته با حداکثر ظرفیت 10 آیتم خواهید داشت. برای اینکه تعداد عناصر این پشته را بیان کنیم از یک متغییر با نام N استفاده می‌کنیم و مقدار 10 را ردون آن قرار می‌هیم.

```
Dim N as Integer
N = 10
```

از یک متغییر دیگر برای نگهداری تعداد خانه‌های پر آرایه (یا همان بالاترین عضو پر آرایه) استفاده می‌کنیم. به این متغییر نام Top را می‌دهیم و مقدار اولیه آن را برابر با صفر قرار می‌دهیم.حال دو تابع برای قرار دادن (تابع Push) و برداشتن از پشته (Pop) می‌نویسیم.

```
Function Pop() As String
If Top_S = 0 Then
   MsgBox "Stack is empty", vbCritical, "Error"
   Exit Function
Else
   Top_S = Top_S - 1
   Pop = Stack(Top_S)
End If
End Function
```

```
Sub Push(Item As String)
If Top_S = N Then
   MsgBox "Stack is full", vbCritical, "Error"
   Exit Sub
Else
   Stack(Top_S) = Item
   Top_S = Top_S + 1
End If
End Sub
```

تابع Push در ابتدا بررسی می‌کند که آیا پشته پر است یا نه. اگر پشته پر بود این تابع یک اخطار می‌دهد و از انجام بقیه عملیات صرف نظر می‌کند. در غیر این صورت در ابتدا داده را در بالاترین عضو خال پشته (TOP_S) قرار داده و سپس یک واحد TOP_S را افزایش می‌دهد.به این ترتیب داده‌ها به ترتیب وارد پشته می‌شوند.

تابع Pop در ابتدا خالی بودن پشته را با چک کردن متغییر TOP_S با مقدار صفر انجام می‌دهد.در صورت خالی بودن این تابع یک اخطار را تولید می‌کند و مقدار تهی را بر می‌گرداند. در غیر این صورت تابع در ابتدا یک واحد از TOP_S کم می‌کند و سپس آن را بر می‌گرداند.

برای دیدن یک مثال عملی از پشته می‌توانید از مثال Stack که همراه این کتاب الکترونیکی آمده است استفاده کنید.

کنترل خطا در OpenGL

خطا همیشه ممکن هستند در هنگام کار کردن با OpenGL رخ دهند و ما باید به طور مدوام به کنترل خطاها بپردازیم زیرا حتی اگر کد برنامه هم درست باشد ممکن است به علتی مواردی همچون محدودیت منابع سیستم یک خطا تولید شود.و از آنجا که OpenGL این خطاها را گزارش نمی‌دهد و حتی آنها موجب توقف برنامه شما نمی‌شوند شما می‌توانید نتیجه نا مطلوبی را بگیرید.

برای کنترل خطاها و مطلع شدن از روی دادن آنها می‌توانید از دستور glGetError استفاده کنید. که شکل کلی آن به صورت زیر آمده است:

```
Function glGetError() As glErrorConstants
```

مقادیری که این دستور بر می‌گرداند به همراه توضیح آنها در جدول زیر آورد شده.

نام خطا	توضیح
glerrNoError = 0	هیچ خطایی رخ نداده است.
glerrInvalidValue = 1281	یک مقدار نا معتبر در تابع وارد شده است.
glerrInvalidOperation = 1282	عملیات نا معتبر رخ داده: از جمله این موارد می‌توان به فراموش کردن بستن بلوک glBegin/glEnd و یا استفاده از دستورهای غیر مجاز در ما بین همین بلوک نام برد.
glerrOutOfMemory = 1285	حافظه به اندازه کافی موجود نیست.
glerrStackOverflow = 1283	پشته سر ریز شده است.(پشته پر است)
glerrStackUnderflow = 1284	پشته ته ریز شده است.(پشته خال است)
glerrInvalidEnum = 1280	نوع ثابت استفاده شده در تابع شناخته شده نیست.معمولا در تابع glBegin رخ می‌دهد.

تکه کد زیر شیوه استفاده از دستور glGetError را نشان می‌دهد.

```
glBegin bmTriangles
        glVertex2f -.5,0
        glVertex2f .5,0
        glVertex2f 0,.5
glEnd
IF glGetError <> glerrNoError then
        Msgbox "Error"
        Exit Sub
End IF
```

کتابخانه VBOGL

بسیاری از دوستان پس از نوشته شدن قسمت اول این سری آموزشی پس از تماس با خواستار این کتابخانه شدند. لازم به ذکر است (و همانگونه که در قسمت اول نیز اعلام کردیم) شما می‌توانید کتابخانه به همراه تمرین شماره اول (از سری اول) موجود است. شما همچنین می‌توانید کتابخانه VBOGL را از سایتهای زیر دریافت کنید:

www.ps-ware.net
www.nehe.gamedev.net
http://www.sourceforge.net

در صورت مواجه شدن با هرگونه سوال یا مشکل در مورد OpenGL و یا ویژوال بیسیک می‌توانید از طریق ایمیل زیر با من در تماس باشید.

Email: pswin@ps-ware.net

برنامه‌نویسی OpenGL در ویژوال بیسیک 6 (قسمت دوم)

دسته‌بندی: برنامه‌نویسي و گرافیک کامپیوتری
پوشش می‌دهد: شیوه آموزش و مبانی اولیه برنامه‌نویسی OpenGL در ویژوال بیسیک 6
سطح کاربر: متوسط به بالا

تهیه شده: توسط پویا شاهین‌فر
انتشارات: انتشارات مجازی PS-Ware (PS-Press)
تعداد صفحات: 33 صفحه
تاریخ انتشار: 16 تیر 1386

---

www.PS-Ware.net

www.press.PS-Ware.net