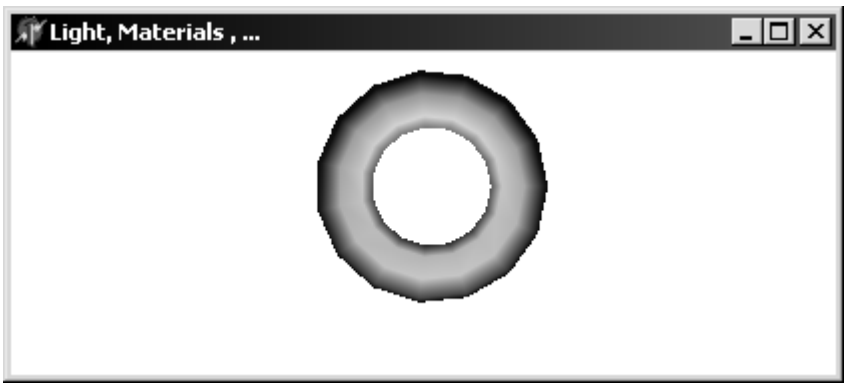
فصل هفتم

نور پردازی ، مواد و دوربین ها



مقدمه :

نور چیست ؟ الکترون ها سطوح انرژی متفاوتی دارند . هنگامیکه الکترونی تحریک می شود به سمت سطح مداری بالاتر می رود و زمانیکه به سطح نرمال خود باز می گردد ، انرژی را به فرم فوتون ساتع می کند . شدت نور و طول موج ، رنگ نوری را که می بینیم معین می کنند.

نور در OpenGL :

OpenGL دو نوع منبع نوری در اختیار برنامه نویس قرار می دهد : سوئی و مکانی . منبع نور سوئی در مکانی به فاصله بی نهایت از اشیاء صحنه قرار گرفته است . بنابراین شعاع های نوری آن هنگامیکه به جسم می رسند ، به موازات شیء به نظر می رسند . نور مکانی ، در نزدیکی یا درون صحنه قرار دارد و جهت شعاع های نوری آن در محاسبات مربوطه بدست می آیند . اینگونه نورپردازی نسبت به نور پردازی سوئی کارآیی بیشتری به دلیل محاسبات ذکر شده

دارد. دستور glLightfv برای تعیین مکان منبع نور و سوئی یا مکانی بودن آن بکار می رود . همچنین در تعیین مقدار رنگ منبع نوری نیز نقش دارد مانند : رنگ محیطی (Ambient) ، رنگ انتشاری (Diffuse) ، رنگ آینه ای (Specular) ، رنگ های درخشنده و غیره .

هنگامیکه منبع نور تعریف گشت ، بردارهای نرمال و خواص مواد اشیاء موجود در صحنه نیز باید تعریف شوند . بردارهای نرمال یک شیء ، جهت آنرا نسبت به منبع نور ، معین می کنند . بردارهای نرمال برای یک یا چند راس می توانند بکاربرده شوند . دستور glNormal برای انجام این کار فراخوانی می شود .

اجزاء رنگی که برای نورتعریف می شوندمعنای متفاوتی با آنچه که برای مواد تعریف می شوند ، دارند . برای نورها ، اعداد متناظر با شدت هر رنگ هستند و برای مواد این اعداد نشانگر خواص انعکاسی این رنگ ها می باشند . دستور glMaterialfv برای تعیین خواص مواد با تعریف مقادیر اجزاء رنگ ماده بکار می رود .

دستور glColorMaterial برای حداکثر کردن کارآیی تغییر خواص مواد فراخوانی می گردد . این تابع هنگامی بکار برده می شود که یک خاصیت ماده باید برای اغلب رئوس در صحنه تغییر کند . هرگونه تغییری در رنگ جاری توسط دستور glColor بطور همزمان خاصیت ماده تعیین شده با glColorMaterial را به روز در می آورد . نورها و همچنین خاصیت مواد باید توسط تابع glEnable فعال گردند تا دستورات مؤثر واقع شوند .

در OpenGL یک منبع نور هنگامی تاثیر خواهد داشت که سطحی وجود داشته باشد تا نور را جذب یا بازتابش نماید. هر سطح از ماده ای ازخواصی معین تشکیل شده است: جذب ، Opacity ، Specular ، Ambient و Emmisive .

نور محیطی (Ambient) : نوری است که توسط محیط تفرق یافته است ، بطوریکه جهت آن قابل تشخیص نیست .

نور انتشاری (Diffuse) : نوری است که هنگامیکه شیء بوسیله نور سوئی روشن می شود ، از آن منعکس می شود .

نور آینه ای (Specular) : از جهتی خاص تابیده شده و تمایل دارد که در جهتی مرجح منتشر شود . فلز درخشنده و یا یک پلاستیک ، جزء آینه ای بالایی دارند .

OpenGL تخمینی از رنگ ماده را بوسیله درصد قرمز ، سبز و آبی که منعکس می کند ، ارائه می دهد . برای مثال : نور سفید و بازتابش قرمز از ماده سبب ایجاد سطحی قرمز می شود و یا نور آبی و بازتابش قرمز از ماده سطحی مشکی را پدید می آورد . بازتابش محیطی مواد با جزء

محیطی منبع نور ، ترکیب می شود . موادی که رنگ Emmisive دارند ، منبع نوری سرچشمه گرفته شده از ماده را شبیه سازی می کنند .

اضافه کردن نور به صحنه بطور خلاصه شامل مراحل زیر است :

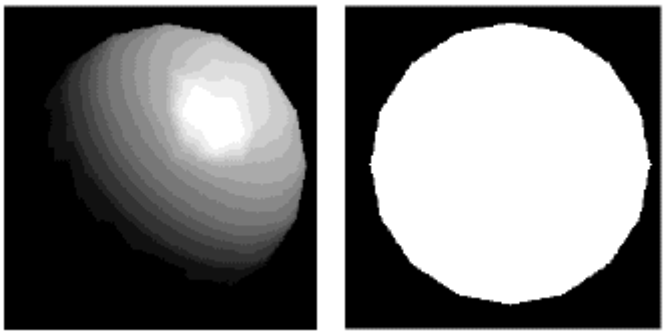
۱- تعریف بردار نرمال برای هر راس تمام اشیاء .

۲- ایجاد ، انتخاب و تعیین مکان یک یا چند منبع نوری .

۳- ایجاد و انتخاب مدل نور پردازی .

۴- تعریف خواص مواد اشیاء در صحنه .

در شکل زیر کره سمت راست را بدون نور پردازی و کره سمت چپ را با نورپردازی انجام شده ملاحظه می فرمایید .



در ادامه مروری مفصل بر توابع مورد استفاده در مثالهای این فصل خواهیم داشت :

تابع به فرمت زبان C	تابع به فرمت زبان دلفی
void glLightf(GLenum light, GLenum pname, GLfloat param);	Procedure glLightf(light: GLenum; pname: GLenum; param: GLfloat); stdcall; external 'OPENGL32.DLL';
void glLightfv(GLenum light, GLenum *pname*, const GLfloat *params);	Procedure glLightfv(light: GLenum; pname: GLenum; params: PGLfloat); stdcall; external 'OPENGL32.DLL';

توضیح :

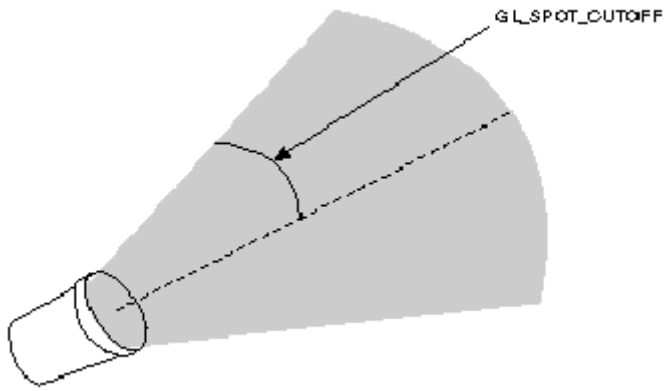
glLightf مقادیر منبع نوری منفرد را تعیین می کند . pname نام پارامتر و params مقدار آن پارامتر می باشد . آرگومان light تعداد نورهای مهیا ، وابسته به نوع dll موجود می باشد ، اما

حداقل ۸ نور پشتیبانی مـی شـود . آنهـا بوسـیله نمادهـای GL_LIGHTi کـه 0<=i<=GL_MAX_LIGHT است معین می شوند .

Pname : پارامتر منبع نوری که می تواند یکی از مقادیر زیر را داشته باشد :

GL_SPOT_EXPONENT : در این صورت آرگومان param شدت توزیع نور را تعیین می کند و ایـن عدد بین ۰ و ۱۸۰ قرار دارد .

GL_SPOT_CUTOFF : در این حالت آرگومـان param حـداکثر زاویـه پخـش شـدن منبـع نـوری را مشخص می کند . فقط مقادیر بین ۰ و ۹۰ درجه و مقدار خاص ۱۸۰ درجه قابل قبول هستند .



GL_LINEAR_ATTENUATION
GL_CONSTANT_ATTENUATION
: GL_QUADRATIC_ATTENUATION

در این حالت آرگومان param یکی از سه فاکتورهای میرایی نور را معین می کند . تنهـا مقـادیر غیر منفی قابل قبول هستند .

تابع glLightfv آرگومان light آن همانند تابع glLightf است . pname پارامتر منبع نور است و فقـط مقادیر زیر قابل قبول هستند :

GL_AMBIENT و GL_DIFFUSE وGL_SPECULAR : در این حالت ها آرگومان param حاوی چهـار عـدد تعیین کننده شدت RGBA نور محیطی ، انتشاری و آینه ای به ترتیب می باشد .

GL_POSITION : در این حالت آرگومان param حاوی چهار عدد خواهد بود که بیـانگر مقعیـت نـور در دستگاه مختصات هموژن می باشند .

GL_SPOT_DIRECTION : در این حالت آرگومان param حاوی سه عدد می باشـد کـه سـوی نـور را دستگاه مختصات هموژن معین می کند .

GL_SPOT_EXPONENT
GL_SPOT_CUTOFF
GL_CONSTANT_ATTENUATION

GL_LINEAR_ATTENUATION
و GL_QUADRATIC_ATTENUATION : همانند آنچه که در قسمت قبل گفته شد می باشند .

با glEnable(GL_LIGHTING); فعال شده و با glDisable(GL_LIGHT*i*); غیر فعال می شود .

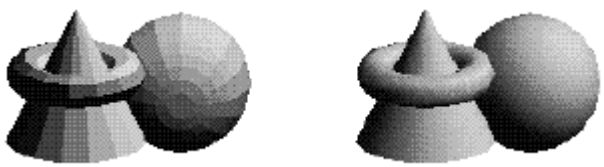
تابع به فرمت زبان C	تابع به فرمت زبان دلفی
void glNormal3f(GLfloat nx, GLfloat ny, GLfloat nz);	Procedure glNormal3f(nx: GLfloat; ny: GLfloat; nz: GLfloat); stdcall; external 'OPENGL32.DLL';
void glNormal3fv(const GLfloat * v);	Procedure glNormal3fv(v: PGLfloat); stdcall; external 'OPENGL32.DLL';

توضیح :

بردار نرمال جاری را تنظیم می کند . nX ، nY و nZ مختصات بردار جدید هستند . در حالت fv ، آرگومان v اشاره گری است به آرایه ای از سه عنصر یاد شده .

بردار نرمال برداری است عمود بر یک صفحه . دو بردار : بردار ۱ (V1) و بردار ۲ (V2) از سه نقطه تشکیل می شوند . با ضرب برونی دو بردار (Cross Product) بردار نرمال حاصل می شود . سپس بردار نرمال به بردار که اندازه آن یک است ، تبدیل می گردد ، تا در روتین های نورپردازی از آن استفاده کرد . البته توسط دستور glEnable(GL_NORMALIZE); می توان قسمت آخر را به صورت خودکار انجام داد که خود باعث کاهش کارآیی برنامه بدلیل انجام محاسبات بیشتر می گردد .

بردارهای نرمال شیءایی جهت سطح را در فضا نسبت به منبع نور تعیین می کنند . اینگونه بردارها درOpenGL برای تعیین اینکه شیء چقدر نور در رئوسش دریافت کرده است ، بکار می روند . باید یاد آوری کرد که سطوح هموار منحنی ها بوسیله تعداد زیادی چند ضلعی تقریب زده می شوند . اگر فقط بردارهای عمود براین چند ضلعی ها بعنوان بردارهای نرمال سطوح بکار برده شوند ، سطح تراش خورده به نظر خواهد رسید (شکل زیر) . با استفاده از نرمال های واقعی ، عملیات رندر کردن بهبود قابل ملاحظه ای پیدا می کند . در انتهای فصل برنامه ای کمکی برای محاسبه بردار نرمال ارائه خواهد گشت که می تواند ایده ای برای برنامه های آتی باشد .



تابع به فرمت زبان C	تابع به فرمت زبان دلفی
void glMaterialfv(GLenum face, GLenum pname, const GLfloat *params);	Procedure glMaterialfv(face: GLenum; pname: GLenum; params: PGLfloat); stdcall; external 'OPENGL32.DLL';
void glMaterialf(GLenum face, GLenum pname, GLfloat param);	Procedure glMaterialf(face: GLenum; pname: GLenum; param: GLfloat); stdcall; external 'OPENGL32.DLL';

توضیح :

این توابع پارامترهای ماده را برای مدل نور پردازی تعیین می کنند .

glMaterialf :

Face : وجه یا وجوهی است که باید به روز در آورده شوند و یکی از سه مقدار GL_FRONT ، GL_BACK و GL_FRONT_AND_BACK می تواند باشد .

Pname : در این حالت فقط GL_SHININESS می تواند باشد و param مقداری است که پارامتر یاد شده به آن تنظیم خواهد شد .

glMaterialfv :

آرگومان face در آن همانند تابع قبلی است و pname مقادیر زیر را می تواند داشته باشد :

GL_AMBIENT و GL_DIFFUSE وGL_SPECULAR : همانند آنچه که در قسمت قبل گفته شد می باشند .

GL_EMISSION : در این حالت پارامتر param حاوی چهار عدد خواهد بود که شدت RGBA نور ساتع شده را معین می کنند .

GL_SHININESS : در این حالت پارامتر param حاوی چهار عدد خواهد بود که جزء آینه ای RGBA را معین می کنند و در بازه ۰ و ۱۲۸ قرار دارند .

GL_AMBIENT_AND_DIFFUSE : که کاملا واضح است .

GL_COLOR_INDEXES : در این param حاوی سه عدد خواهد بود که اندیس های رنگی محیطی ، انتشاری و آینه ای را معین می کنند . این سه مقدار وGL_SHININESS تنها مقادیر مواد بکار گرفته شده در این حالت نور پردازی می باشند .

تابع به فرمت زبان C	تابع به فرمت زبان دلفی
void glColorMaterial(GLenum face, GLenum mode);	Procedure glColorMaterial(face: GLenum; mode: GLenum); stdcall; external 'OPENGL32.DLL';

توضیح :

این تابع سبب می گردد تا رنگ ماده ، رنگ جاری را پی گیری کند .

Face : معین می کند که جلو ، عقب یـا هـردوی آن هـا بایـد رنـگ جـاری را پـی گیـری کننـد و پارامترهای مجاز آن همانند تابع glMaterialf می باشند .

Mode : معین می کند کـه کـدامیک از پارامترهـای مـاده بایـد رنـگ جـاری را پـی گیـری کننـد . پارامترهای مجاز آن به صورت زیر هستند :

GL_AMBIENT_AND_DIFFUSE ، GL_SPECULAR ، GL_DIFFUSE ، GL_AMBIENT ، GL_EMISSION

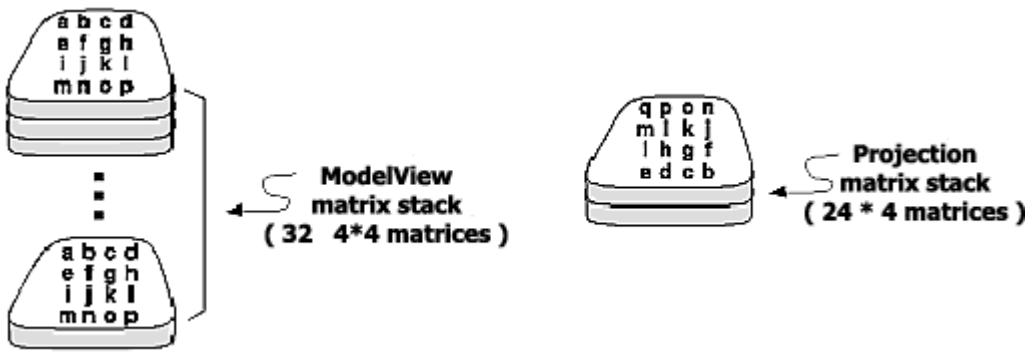
این تابع توسط دستور glEnable(GL_COLOR_MATERIAL); فعال می شود .

تابع به فرمت زبان دلفی	تابع به فرمت زبان C
Procedure glPushMatrix; stdcall; external 'OPENGL32.DLL';	void glPushMatrix(void);
Procedure glPopMatrix; stdcall; external 'OPENGL32.DLL';	void glPopMatrix(void);

توضیح :

سبب Push و Pop شدن پشته ماتریس جاری می شوند . به ازای هر حالت ماتریسی یک پشـته ماتریسی وجود دارد . در حالت GL_MODELVIEW ، عمق پشته ماتریسی حداقل ۳۲ اسـت . در دو حالت دیگر GL_PROJECTION و GL_TEXTURE ، عمق پشته حـداقل ۲ اسـت . تـابع glPushMatrix سبب انتقال پشته ماتریس جاری یکی به سمت پایین می شود و ماتریس جاری را تکثیرمی کنـد . بعد از فراخـوانی تـابع glPushMatrix ، مـاتریس بـالایی پشـته نسـبت بـه یکـی پـایین آن ، واحـد می باشد . glPopMatrix ، ماتریس جاری پشته را با یکـی زیـر آن تعـویض مـی کنـد . احتمـالا از تعاریف MSDN ایی فوق چیزی متوجه نشده اید ! پس به تعاریف زیر توجه کنید :

ماتریس های ModelView و Projection تنها ، بالاترین اعضای یک پشته ماتریسی می باشند (شکل زیر) .

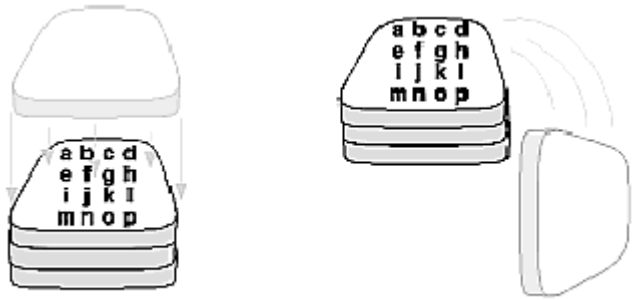


پشته ماتریسی برای ساخت مدلهای پیچیده که از اجزای ساده تشکیل شده اند ، مفید مـی باشـد . فرض کنید می خواهیم یک خودرو که چهار چرخ دارد را ترسیم کنیم و هر چرخ بـه ماشـین بـا ۵ پیچ اتصال یافته است . یک روتین برای ترسیم چرخ و یک روتین دیگر برای ترسـیم پـیچ نوشـته

می شود ، زیرا تمام چرخ ها و پیچ ها یکسان بـه نظـر مـی رسـند . هنگامیکـه خـودرو را ترسـیم می کنید ، چهار مرتبه باید روتین ترسیم چرخ را فراخوانی کنید ، البته بـا انتقـالات و دوران هـای لازم و سپس باید پیچ ها در موقعیت مناسب ترسیم شوند . آن بصورت مشابهی برای هر چـرخ ، ترسیم خواهد گردید : بخاطر بیاوریم که کجا هستیم ، به مکان جدید منتقل شویم ، ترسیم جدیدی در آن جا صورت گیرد و به همین ترتیب .

از آنجائیکه انتقالات بصورت ماتریسی ذخیره می شوند ، پشته ماتریسی بـرای انجـام اینگونـه کارها مفید می باشد . تمام دستورات ماتریسی که تاکنون مورد بررسی قرار گرفتند بـا بـالاترین ماتریس ، یعنی ماتریس جاری سروکار داشتند .

دستورglPushMatrix ماتریس جاری را کپی کرده و روی پشته در بالاترین سطح آن قرار می دهد . بطور خلاصه glPushMatrix یعنی : ((بخاطر داشته باش کجا هسـتی)) و glPopMatrix یعنـی : ((برگرد به جایی که بودی)) . glPushMatrix مانند دکمه حافظه روی ماشین حساب عمل می کند و کار آن ذخیره کردن ماتریس جاری است . توسـط glPopMatrix تنظیمـات ذخیـره شـده قبلـی را می توان فراخوانی کرد . برای فراخوانی تنظیمات ذخیره شده باید آنها را درجهـت عکـس ذخیـره شدن ، بارگذاری نمود .



تصویری از Push و Pop شدن .

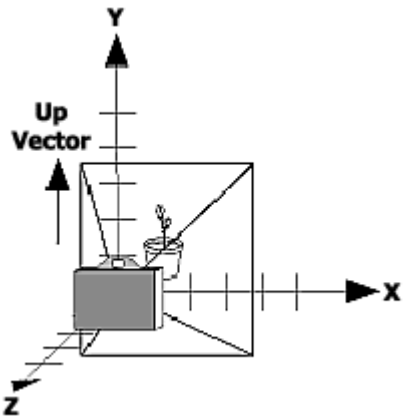
این روش بسیار مؤثرتر از کارکردن با ماتریس های منفرد می باشـد ، خصوصـا اگـر توسـط سخت افزار نیز پشتیبانی شود .

تابع به فرمت زبان C	تابع به فرمت زبان دلفی
void gluLookAt(GLdouble eyex, GLdouble eyey, GLdouble eyez, GLdouble centerx, GLdouble centery, GLdouble centerz, GLdouble upx, Gldouble upy, GLdouble upz);	Procedure gluLookAt(eyex: GLdouble; eyey: GLdouble; eyez: GLdouble; centerx: GLdouble; centery: GLdouble; centerz: GLdouble; upx: GLdouble; upy: GLdouble; upz: GLdouble); stdcall; external 'GLU32.DLL';

توضیح :

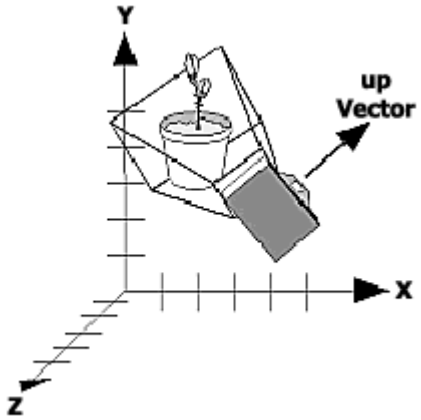
ماتریس انتقال دید را تعریف می کند .

eyeX ، eyeY و eyeZ مختص های مکان دید را تعریف می کنند . centerX ، centerY و centerZ مختص های مکان مرجع را و upX ، upY و upZ جهت بردار بالا را تعیین می نمایند .



پیش فرض های تابع gluLookAt .

gluLookAt(0,0,0,0,0,-100,0,1,0);



مثال : موقعیت دوربین (4,2,1)

موقعیت نقطه مرجع (2,4,-3)

بردار سوئی (2,2,-1) که نقطه دید را ۴۵ درجه چرخانیده است .

پس از ساخت یک صحنه ، لازم می باشد تا از یک نقطه مشخص و دلخواه به آن نگاه شود تا اشیاء بخوبی به نظر برسند . از این تابع بدین منظور استفاده می شود .

نقطه مرجع ، نقطه ای است در میان صحنه . برای مثال اگر صحنه ای را در مبدا بنا کرده اید ، این نقطه همان مبدا خواهد بود . تعیین بردار Up کمی نیازمند دقت می باشد . برای مثال اگر می خواهید شبیه ساز پروازی را طراحی کنید ، Up جهتی است عمود بر بالای هواپیما به سمت آسمان ، هنگامیکه هواپیما روی زمین قرار دارد . در شکل های فوق مکان پیش فرض دوربین مشخص گردیده است و دوربین بدون فراخوانی این تابع در آنجا قرار دارد .

تذکر : قبل از تغییر مکان دوربین حتما از glLoadIdentity استفاده کنید .

تابع به فرمت زبان C	تابع به فرمت زبان دلفی
void glLightModelf(GLenum pname, GLfloat param);	Procedure glLightModelf(pname: GLenum; param: GLfloat); stdcall; external 'OPENGL32.DLL';
void glLightModelfv(GLenum pname, const GLfloat *params);	Procedure glLightModelfv(pname: GLenum; params: PGLfloat); stdcall; external 'OPENGL32.DLL';

توضیح :

پارامترهای مدل نورپردازی را تنظیم می کنند .

glLightModelf :

Pname : نام پارامتر نور پردازی است و یکی از مقادیر زیر را می تواند داشته باشد :

GL_LIGHT_MODEL_LOCAL_VIEWER : در این حالت آرگومان param چگونگی محاسبه زاویه بازتابش آینه ای را مشخص می کند . اگر مقدار آن صفر باشد ، زاویه بازتابش آینه ای در جهت منفی محور Z ها خواهد بود .

GL_LIGHT_MODEL_TWO_SIDE : در این حالت param معین می سازد که محاسبات نورپردازی برای یک یا دو سمت چند ضلعی ها بکار رود . بر روی نقاط ، خطوط و بیت مپ ها تاثیر ندارد. اگر param مساوی صفر باشد نور پردازی یک سمت انجام شده و فقط پارامترهای ماده جلو در محاسبات نورپردازی بکار گرفته می شوند؛ در غیر اینصورت هر دو سمت محاسبه خواهند شد .

glLightModelfv :

این تابع علاوه بر پشتیبانی از دو حالت ذکر شده برای تابع قبل ازGL_LIGHT_MODEL_AMBIENT نیز می تواند استفاده کند . در این حالت پارامتر param حاوی چهار عدد خواهد بود که تعیین کننده شدت RGBA محیطی کل صحنه می باشد .

اولین برنامه فصل :

در این برنامه با طرز استفاده از نورها و مواد در OpenGL آشنا خواهیم شد :

تذکر : این برنامه ، به یک کنترل تایمر برای اجرا شدن نیز نیاز دارد .

```
unit Ch07_01;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
  Forms, Dialogs, OpenGL, SPF, Math, ExtCtrls;
```

```
type
  TForm1 = class(TForm)
    Timer1: TTimer;
    procedure FormResize(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FormPaint(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }

  end;

var
  Form1: TForm1;

  position : array[0..3] of GLfloat = (0,0,1.5,1);
  spin : GLfloat;
  mat_specular : array[0..3] of GLfloat = (1,1,1,1);
  diffuseMaterial : array[0..3] of GLfloat = (0.5, 0.5, 0.5, 1);

  f_Hdc : LongInt;

implementation

{$R *.DFM}

{ doughnut:
 * draws a doughnut, centered at (0, 0, 0) whose axis is aligned with
 * the z-axis.  The doughnut's major radius is R, and minor radius is r.}
procedure doughnut( R1 , R2 : GLdouble;
                    nsides , rings : GLint;
                    thetype : GLenum);
var
   i , j : Integer;
   theta , phi , theta1 , phi1 : GLdouble ;
   p0 , p1 , p2 , p3 :  array[0..2] of GLdouble;
   n0, n1, n2,n3 :  array[0..2] of GLdouble;

begin
    for i := 0 To rings - 1  do
      begin
        theta := i * 2 * 3.1415927 / rings ;
        theta1 := (i + 1) * 2 * 3.1415927 / rings ;
        for j := 0 To nsides - 1  do
          begin
            phi := j * 2 * 3.1415927 / nsides;
            phi1 := (j + 1) * 2 * 3.1415927 / nsides;
            p0[0] := Cos(theta) * (R2 + R1 * Cos(phi));
            p0[1] := -Sin(theta) * (R2 + R1 * Cos(phi));
            p0[2] := R1 * Sin(phi);
            p1[0] := Cos(theta1) * (R2 + R1 * Cos(phi));
            p1[1] := -Sin(theta1) * (R2 + R1 * Cos(phi));
            p1[2] := R1 * Sin(phi);
            p2[0] := Cos(theta1) * (R2 + R1 * Cos(phi1));
```

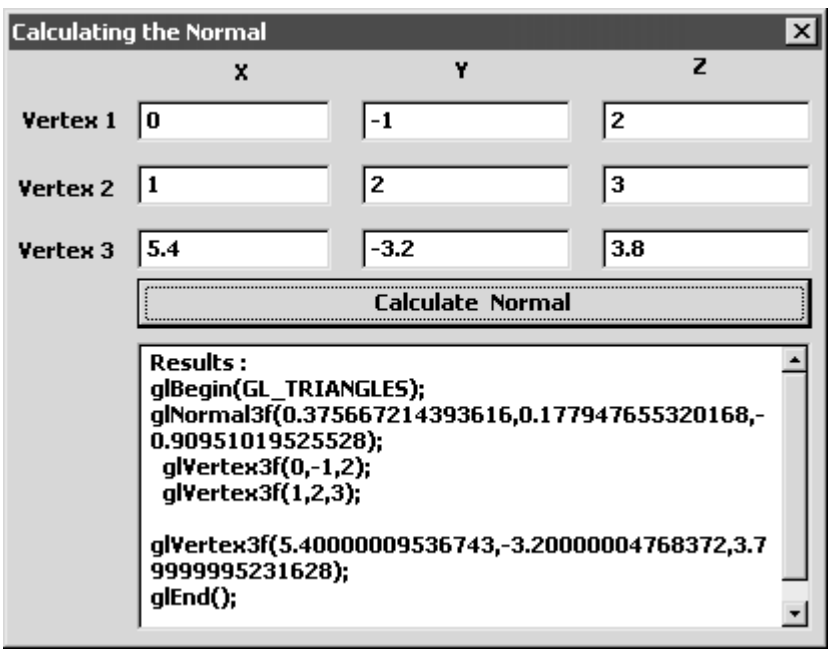
```
        p2[1] := -Sin(theta1) * (R2 + R1 * Cos(phi1));
        p2[2] := R1 * Sin(phi1);
        p3[0] := Cos(theta) * (R2 + R1 * Cos(phi1));
        p3[1] := -Sin(theta) * (R2 + R1 * Cos(phi1));
        p3[2] := R1 * Sin(phi1);
        n0[0] := Cos(theta) * Cos(phi);
        n0[1] := -Sin(theta) * Cos(phi);
        n0[2] := Sin(phi);
        n1[0] := Cos(theta1) * Cos(phi);
        n1[1] := -Sin(theta1) * Cos(phi);
        n1[2] := Sin(phi);
        n2[0] := Cos(theta1) * Cos(phi1);
        n2[1] := -Sin(theta1) * Cos(phi1);
        n2[2] := Sin(phi1);
        n3[0] := Cos(theta) * Cos(phi1);
        n3[1] := -Sin(theta) * Cos(phi1);
        n3[2] := Sin(phi1);
        glBegin (thetype);
           glNormal3dv (@n3);
              glVertex3dv (@p3);
           glNormal3dv (@n2);
              glVertex3dv (@p2);
           glNormal3dv (@n1);
              glVertex3dv (@p1);
           glNormal3dv (@n0);
              glVertex3dv (@p0);
        glEnd;
     end;
  end;

End;

procedure auxSolidTorus(innerRadius , outerRadius : GLdouble);
begin
//       Another mostly-direct port.
     doughnut(innerRadius, outerRadius, 8, 15, GL_QUADS);
end;

procedure initGL();
begin
//  Initialize material property, light source, lighting model,
//  and depth buffer.

  glClearColor (1, 1, 1, 0);
  glShadeModel (GL_SMOOTH);
  glEnable (GL_LIGHTING);
  glEnable (GL_LIGHT0);
  glMaterialfv (GL_FRONT, GL_DIFFUSE, @diffuseMaterial);
  glMaterialfv (GL_FRONT, GL_SPECULAR, @mat_specular);
  glMaterialf (GL_FRONT, GL_SHININESS, 25);
  glColorMaterial (GL_FRONT, GL_DIFFUSE);
  glEnable (GL_COLOR_MATERIAL);
  glEnable (GL_DEPTH_TEST);
End ;

procedure DrawGLScene();
 begin
  glClear (GL_COLOR_BUFFER_BIT Or GL_DEPTH_BUFFER_BIT);
```

```
   glPushMatrix;
   gluLookAt (0, 0, 5, 0, 0, 0, 0, 1, 0);

   glPushMatrix;
   glRotated (spin, 1, 0, 0);
   glLightfv (GL_LIGHT0, GL_POSITION, @position);

   glTranslated (0, 0, 1.5);
   glDisable (GL_LIGHTING);
   //glColor3f (1, 0, 0);
    auxSolidTorus(0.1, 0.1);
   glEnable (GL_LIGHTING);
   glPopMatrix;
    auxSolidTorus(0.275, 0.85);
   glPopMatrix;

   //glFlush;
   SwapBuffers (f_Hdc);
End;

procedure TForm1.FormResize(Sender: TObject);
begin
  wglMakeCurrent(f_Hdc,hrc); //activate the RC
// Prevent A Divide By Zero If The Window Is Too Small
// By Making The Height One
  if (Height=0) then   Height:=1;
    glViewport (0, 0, width, height);
   glMatrixMode (GL_PROJECTION);
   glLoadIdentity;
   gluPerspective (40, width / height, 1, 20);
   glMatrixMode (GL_MODELVIEW);
   glLoadIdentity;
   InvalidateRect(Handle, nil, False);// DrawGLScene;  Draw the scene.
end;

procedure TForm1.Timer1Timer(Sender: TObject);
begin
  spin := spin + 5 ;

  diffuseMaterial[0] := diffuseMaterial[0] + 0.01 ;
  diffuseMaterial[1] := diffuseMaterial[1] + 0.02 ;
  diffuseMaterial[2] := diffuseMaterial[2] - 0.03 ;

  If (diffuseMaterial[0] > 1) Then
    begin
      diffuseMaterial[0] := 0;
      diffuseMaterial[1] := 0.1;
   // if (diffuseMaterial[2] > .5) Then diffuseMaterial[2] := 0.2;
    end;
  glColor4fv (@diffuseMaterial);

  Application.ProcessMessages; //DoEvents
  InvalidateRect(Handle, nil, False);//   DrawGLScene();
end;

procedure TForm1.FormDestroy(Sender: TObject);
begin
 CleanUp(f_Hdc);// Clean up and terminate.
```

```
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
 f_Hdc := GetDC(handle);
 SetDCPixelFormat(f_Hdc,16,16);// Create a rendering context.
 InitGL;
end;

procedure TForm1.FormPaint(Sender: TObject);
begin
 wglMakeCurrent(f_Hdc,hrc); //activate the RC
 DrawGLScene();
end;

end.
```

برنامه دوم فصل :

تعداد نه TextBox را روی صفحه فرم قرار دهید و نـام آنهـا را بـه ترتیـب بـه txtV1X و ... تغییـر دهید . سپس مطابق شکل با استفاده از تعداد شش Label : نام محورهای x ، y ، z و رئوس یک تا سه را روی فرم پیاده کنید . برای انجام محاسبات یک دکمه بنام btnCalculate نیز روی فـرم قـرار دهید و برای نمایش محاسبات یک Memo مطابق شکل روی فـرم بگذاریـد . کـد کامـل مربـوط بـه قسمت های مختلف فرم ، در ذیل آورده شده است :



```
unit ch07_02;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls;
type
  TForm1 = class(TForm)
    txtV1X: TEdit;
    txtV1Y: TEdit;
    txtV1Z: TEdit;
    txtV2X: TEdit;
    txtV2Y: TEdit;
    txtV2Z: TEdit;
    txtV3X: TEdit;
    txtV3Y: TEdit;
    txtV3Z: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    btnCalculate: TButton;
    Memo1: TMemo;
    procedure btnCalculateClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
type
  ToutPut = array[0..2] of single;

implementation
{$R *.DFM}
procedure CalculateNormal(
            V1X,V1Y,V1Z,
            V2X,V2Y,V2Z,
            V3X,V3Y,V3Z : single;
            var outPut: ToutPut);
var
  vector1x,vector1y,vector1z : single;
  vector2x,vector2y,vector2z : single;
  outx,outy,outz : single;
  length : Real;
begin
          //Vector 1
          vector1x := V1X-V2X;
          vector1y := V1Y-V2Y;
          vector1z := V1Z-V2Z;
          //Vector 2
          vector2x := V2X-V3X;
          vector2y := V2Y-V3Y;
          vector2z := V2Z-V3Z;
          //Apply the Cross Product
          outx := vector1y*vector2z - vector1z * vector2y;
```

```
        outy := vector1z*vector2x - vector1x * vector2z;
        outz := vector1x*vector2y - vector1y * vector2x;
        //Normalize to a unit vector
        length := sqrt(outx*outx+outy*outy+outz*outz);
        if(length=0) then length := 1.0;
      outPut[0] := outx/length;
        outPut[1] := outy/length;
        outPut[2] := outz/length;
end;

function generateOpenGL_Code(
          V1X,V1Y,V1Z,
          V2X,V2Y,V2Z,
          V3X,V3Y,V3Z : single ) : string;
var
 outPut :  ToutPut;
 strs : string;
begin
 strs := 'glBegin(GL_TRIANGLES);' +#13#10 ;

 CalculateNormal(
        V1X,V1Y,V1Z,
        V2X,V2Y,V2Z,
        V3X,V3Y,V3Z,
        outPut);

 strs := strs +
   'glNormal3f('+ floattostr(outPut[0]) + ',' + floattostr(outPut[1]) +','+
   floattostr(outPut[2]) + ');' +#13#10 ;

 strs := strs +
  ' glVertex3f('+ floattostr(V1X) +','+ floattostr(V1Y) +','+
  floattostr(V1Z) +');' +#13#10 ;

 strs :=strs +
  ' glVertex3f('+ floattostr(V2X) +','+ floattostr(V2Y) +','+
  floattostr(V2Z) +');' +#13#10 ;

 strs :=strs +
  ' glVertex3f('+ floattostr(V3X) +','+ floattostr(V3Y) +','+
  floattostr(V3Z) +');' +#13#10 ;

 generateOpenGL_Code := strs + 'glEnd();' +#13#10 ;
end;

procedure TForm1.btnCalculateClick(Sender: TObject);
begin
 Memo1.Text := 'Results : ' + #13#10 +
  generateOpenGL_Code(
   strtofloat(txtV1X.text),strtofloat(txtV1Y.text),strtofloat(txtV1Z.text),
   strtofloat(txtV2X.text),strtofloat(txtV2Y.text),strtofloat(txtV2Z.text),
   strtofloat(txtV3X.text),strtofloat(txtV3Y.text),strtofloat(txtV3Z.text)
  );

end;

end.
```