:OpenGL

OpenGL یک API گرافیکی برای ترسیم اشکال دو بعدی و سه بعدی که به صورت زنده (real time) قابل پردازش باشند می باشد. OpenGL به تنهایی قابل استفاده نیست و وابسته به سیستم عامل است. در اینجا ما با استفاده از زبان برنامه نویسی Delphi و بر روی سیستم عامل ویندوز قصد استفاده از این API گرافیکی را داریم.
برای ایجاد صحنه (scene) ، که بتوان در آن از توابع OpenGL استفاده نمود باید سه مرحله را پشته سر بگذاریم.

۱- اینجاد DC (display context)
۲- ایجاد RC (rendering context)
۳- قرار دادن RC بر روی DC

در واقع در ابتدای کار فرمت نقطه ای را با استفاده از توابع سیستم عامل ایجاد می کنیم، بعد صفحه نمایش (RC) و سپس تلفیق این دو با یکدیگر.

ابتدا در form.create در برنامه دلفی این کد را می نویسیم.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  hdc:=GetDC(Panel1.Handle);
  SetDCPixelFormat(hdc,16,16);
  initGL;
end;
```

در خط اول که به رنگ قرمز در آمده است، handle سطحی را که می خواهیم بر روی آن صحنه نمایش را ایجاد کنیم گرفته و در متغیر hdc که از نوع longint است قرار می دهیم. در خط دوم این متغیر را به تابع SetDcPixelFormat می فرستیم. در اینجا ذکر این مطلب واجب است که برای ایجاد RC ، unit از قبل نوشته شده ای به نام SPF (set pixel format) وجود دارد که از آن استفاده نموده و در قسمت uses برنامه باید نام آن را ذکر کنیم. تابع SetDcPixelFormat در داخل این unit قرار داشته و متغیر hrc را که همان (RC) هست را مقدار دهی می کند. این تابع سه آرگمان ورودی دارد. اولی آن ، handle شی که می خواهیم بر روی آن RC را ایجاد کنیم که در اینجا hdc است. پارامتر دوم بافر رنگ است که ۱۶ بیتی در نظر گرفتیم، و پارامتر سوم بافر عمق می باشد که ۱۶ بیتی در نظر گرفتیم. این مقادیر استاندارد می باشند.
در خط سوم زیر برنامه initGL فراخوانی می شود که این زیربرنامه خصوصیات صحنه نمایش ما را تعیین می کند.

```
procedure initGL;
begin
   glShadeModel(GL_SMOOTH);
   glClearColor(0,0,0,0.5);
   glClearDepth(1);
   glEnable(GL_DEPTH_TEST);
   glDepthFunc(GL_LESS);

end;
```

هر خصوصیتی که می خواهیم در دستورات ترسیمی خود به کار ببریم در این قسمت تعیین می کنیم. خط اول glShadeModel برای تعیین نوع رنگ آمیزی سطوح است . خط دوم برای تعیین رنگ زمینه پشتی است که دارای ۴ پارامتر است. ۳ پارامتر اول RGB و آخری Alpha می باشد. (0,0,0) باعث اینجاد رنگ مشکی می شود

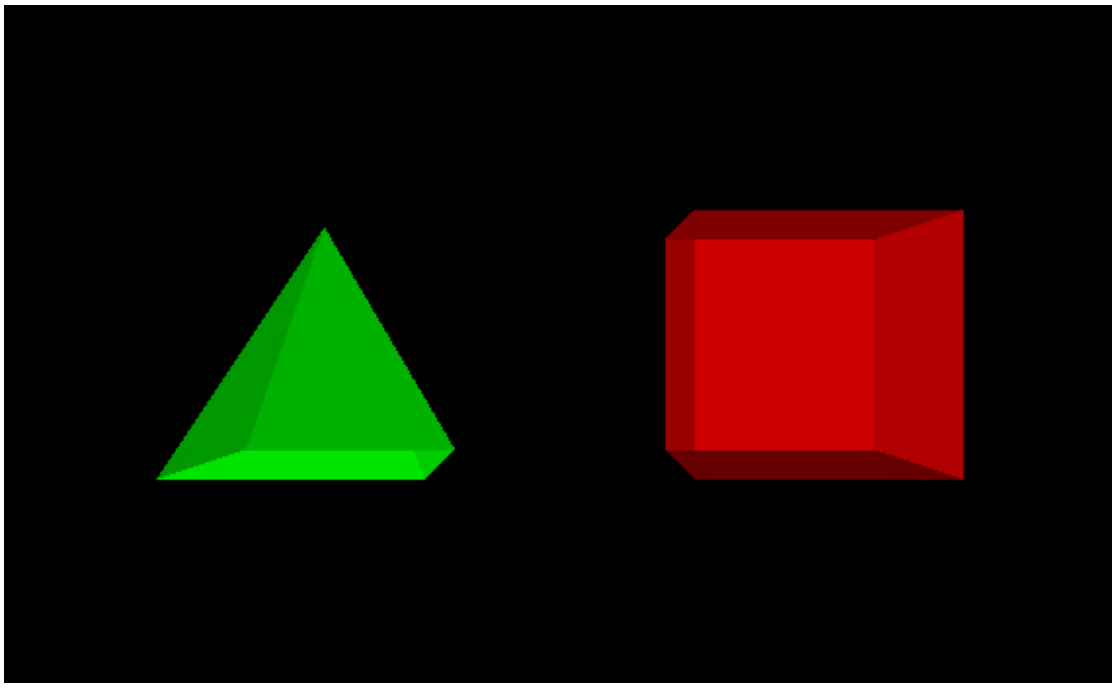
و اگر (1,1,1) داده می شد رنگ زمینه به رنگ سفید تغییر پیدا می کرد. سه خط بعدی برای پاک کردن بافر عمق به کار می رود .

glClearDepth(1); عمق پاک کردن بافر را مشخص می کند.

glEnable(GL_DEPTH_TEST); خصوصیت تست بافر عمق را مشخص می کند.

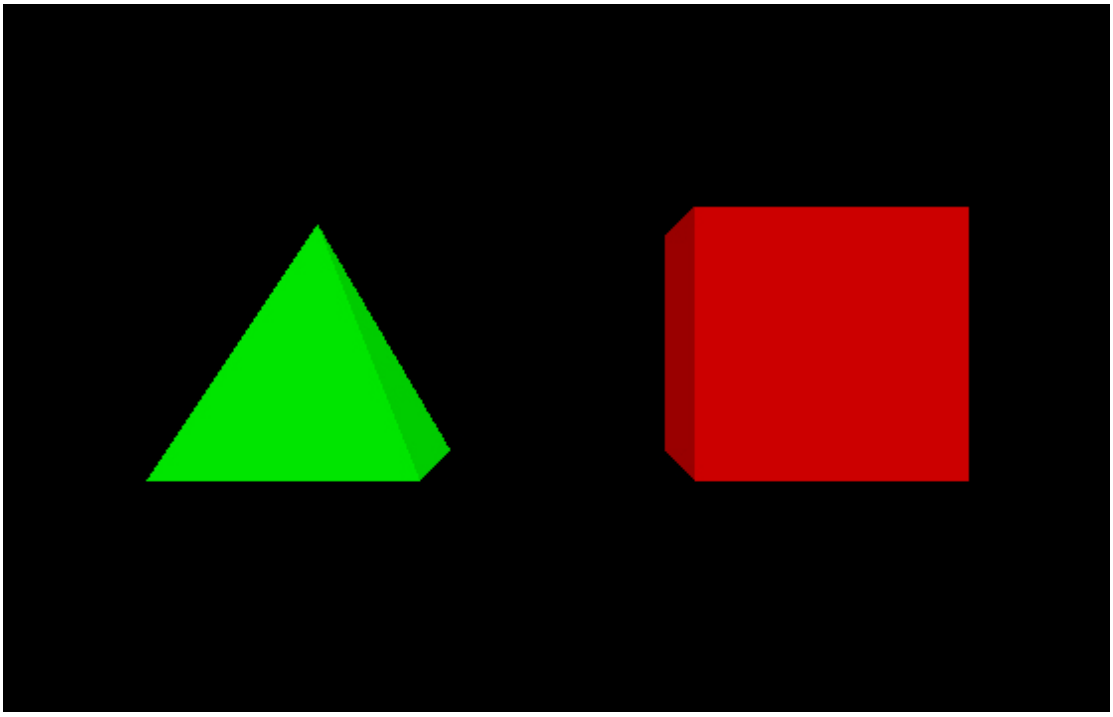
glDepthFunc(GL_LESS); باعث می شود که سطوحی که جلوتر هستند ، عناصر پشته آن را پاک می کند.

اگر این سه خط کد را ننویسیم باعث می شود که آن هایی هم که در پشت سطح به وجود می آیند را هم نشان می دهد.



اگر این سه خط کد نباشد. تصویر بالا پس از اجرا حاصل می گردد.

در تصویر بعد این سه خط کد را نوشته و سپس برنامه را اجرا کردیم.



بعد از ایجاد RC حال نوبت به قرار گیری آن بر روی DC می رسد که این کار را در FormPaint به صورت زیر انجام می دهیم.

```
procedure TForm1.FormPaint(Sender: TObject);
begin
  wglMakeCurrent(hdc,hrc);
  DrawGLScene;
end;
```

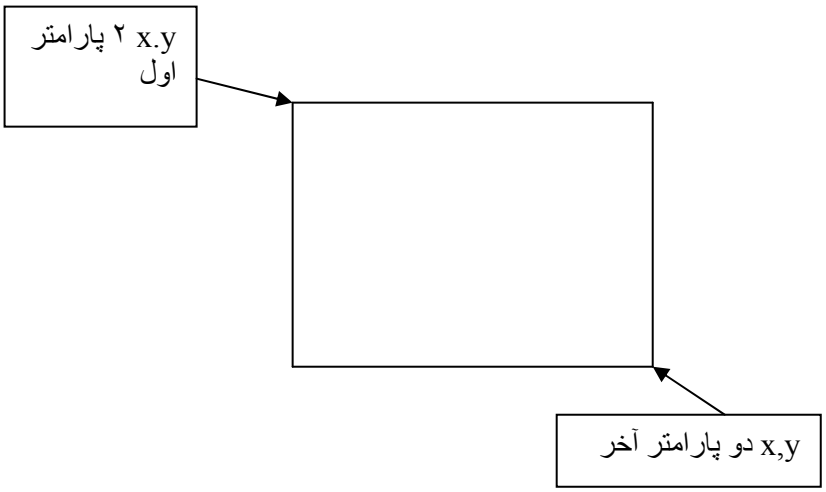
با استفاده از تابع wglMakeCurrent متغیر (hrc) که در مرحله قبل به دست آمد را بر روی hdc قرار می دهیم. در این حالت ما دیگر می توانیم تر سیمات سه بعدی خود را انجام دهیم . در خط بعد نیز زیر برنامه DrawGLSene قرار دارد که در این زیر برنامه از توابع OpenGL برای ترسیم اشکال خود استفاده می کنیم، که بعدا به طور مفصل توضیح داده خواهد شد.

تنها چیز باقی مانده، ایجاد ماتریس ها و نحوه دید است که در زیر برنامه FormResize به صورت زیر این کار را انجام می دهیم.

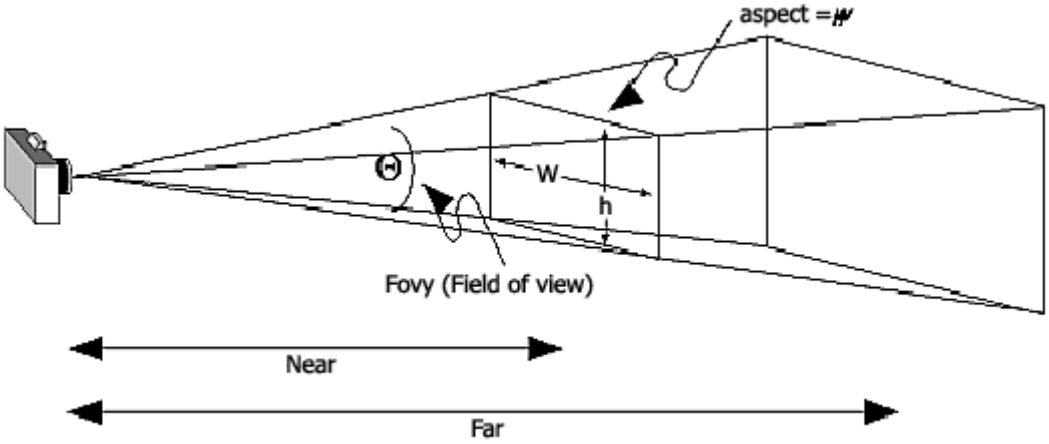
```
procedure TForm1.FormResize(Sender: TObject);
begin

  wglMakeCurrent(hdc,hrc);
  glViewport(0,0,Panel1.Width,Panel1.Height);
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity;
  gluPerspective(45,Panel1.Width/Panel1.Height,1,100);
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity;
  InvalidateRect(Handle,nil,false);

  Panel1.Width:=Form1.Width-20;
  Panel1.Height:=Form1.Height-120;
  RadioGroup1.Top:=panel1.Height+10;
end;
```

اولین خط کد که همان تابع قبلی است یعنی قرار گیری RC بر روی DC در خط دوم تابع glViewport قرار دارد که این تابع برای تعیین درگاه دید صحنه سه بعدی به کار می رود. که دارای ۴ پارامتر است. ۲ پارامتر اول

x,y مبدا که در اینجا صفر است. و ۲ پارامتر آخر x,y مقصد می باشد.



با استفاده از دستور glmatrixmode نوع ماتریس مورد نظر برای ترسیم اشکال خود را تعیین می کنیم ، که در اینجا Projection می باشد. ماتریس Projection ماتریسی است که تمامی ترسیمات ما در آن قرار می گیرد.
سپس با استفاده از دستور glloadidentity ماتریس مورد نظر را در نقطه 0,0,0 (x,y,z) قرار می دهیم. در واقع کار دستور glloadidentity تعیین نقاط (x,y,z) در نقاط (0,0,0) است یعنی ماتریس را reset می کند.
سپس نوع دید را تعیین می کنیم . که دید ما در اینجا perspective می باشد . پس با دستور gluPerspective این دید را ایجاد می کنیم. این دستور ۴ پارامتر دارد. اولی تعیین زاویه دید است که به صورت درجه است (fovy). دومی چپ و راست(w) دید را تعیین می کند. پارامتر سوم بافر نزدیک (z near) و پارامتر چهارم بافر دور (z far) را درست می کند.



در دستور بعد ماتریس modelview را ایجاد می کنیم . این ماتریس برای جابه جایی اشکال ترسیمی و کلیه تغییرات بر روی اشکال مانند چرخش و مقیاس را انجام می دهد. که بعد از تعیین ماتریس از دستور glloadidentity برای reset کردن ماتریس استفاده می شود.
از تابع InvalidateRect برای بروز رسانی صفحه تصویر استفاده می شود. که سه پارامتر آن به صورت پیش فرض می باشند.

تا اینجا کار صحنه سه بعدی را برای کشیدن ترسیمات سه بعدی آماده کرده ایم.
در زیر برنامه DrawGLScene دستورات زیر را می نویسیم.

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

بافرهای عمق و رنگ را پاک می کنیم.

```
  glLoadIdentity;
```

مختصات را به روز رسانی می کنیم

```
  glTranslatef(-5,0,-20);
```

برای انتقال اشیا از این تابع استفاده می شود.

```
  glRotatef(rot_triangle,0,1,0);
```

برای چرخش استفاده می شود. که پارامتر اول ان درجه چرخش است که مقدار متغیر آن در تایمر برنامه به دست می آید.

```
    glBegin(GL_TRIANGLES);
```

آغاز به کار ترسیمات با این دستور است که پارامتر آن شیوه طراحی را نشان می دهد . که در اینجا با هر سه نقطه به عنوان یک سطح برخورد می کند.

```
      glColor3f(0,0.9,0);
```

رنگ سطح را نشان می دهد.

```
      glVertex3f(-2.5,-2.5,2.5);
```

برای تعیین یک راس از شکل استفاده می شود.

```
      glVertex3f(2.5,-2.5,2.5;(
      glVertex3f(0,2.5,0;(

      glColor3f(0,0.8,0;(
      glVertex3f(2.5,-2.5,2.5;(
      glVertex3f(2.5,-2.5,-2.5;(
      glVertex3f(0,2.5,0;(

      glColor3f(0,0.7,0;(
      glVertex3f(2.5,-2.5,-2.5;(
      glVertex3f(-2.5,-2.5,-2.5;(
      glVertex3f(0,2.5,0;(

      glColor3f(0,0.6,0;(
      glVertex3f(-2.5,-2.5,-2.5;(
      glVertex3f(-2.5,-2.5,2.5;(
      glVertex3f(0,2.5,0;(


  glEnd;



glLoadIdentity;
```

```
glTranslatef(5,0,-20;(
glRotatef(rot_cube,1,1,0;(
    glBegin(GL_QUADS);باهر چهار راس به عنوان یک سطح برخورد می کند.

        glColor3f(0.9,0,0;(
        glVertex3f(-2.5,-2.5,-2.5;(
        glVertex3f(-2.5,2.5,-2.5;(
        glVertex3f(2.5,2.5,-2.5;(
        glVertex3f(2.5,-2.5,-2.5;(

        glColor3f(0.8,0,0;(
        glVertex3f(-2.5,-2.5,2.5;(
        glVertex3f(-2.5,2.5,2.5;(
        glVertex3f(2.5,2.5,2.5;(
        glVertex3f(2.5,-2.5,2.5;(

        glColor3f(0.7,0,0;(
        glVertex3f(2.5,2.5,-2.5;(
        glVertex3f(2.5,2.5,2.5;(
        glVertex3f(2.5,-2.5,2.5;(
        glVertex3f(2.5,-2.5,-2.5;(

        glColor3f(0.6,0,0;(
        glVertex3f(-2.5,2.5,-2.5;(
        glVertex3f(-2.5,2.5,2.5;(
        glVertex3f(-2.5,-2.5,2.5;(
        glVertex3f(-2.5,-2.5,-2.5;(

        glColor3f(0.5,0,0;(
        glVertex3f(-2.5,2.5,-2.5;(
        glVertex3f(-2.5,2.5,2.5;(
        glVertex3f(2.5,2.5,2.5;(
        glVertex3f(2.5,2.5,-2.5;(

        glColor3f(0.4,0,0;(
        glVertex3f(-2.5,-2.5,-2.5;(
        glVertex3f(-2.5,-2.5,2.5;(
        glVertex3f(2.5,-2.5,2.5;(
        glVertex3f(2.5,-2.5,-2.5;(
```

```
    glEnd;
```

```
  SwapBuffers(hdc);
```
بافر ها را به روز رسانی می کند. در واقع تمام ترسیمات را بر روی صفحه کشیده و از ترسیم آنها د ر صفحه مطمئن می شود.

```
end;
```