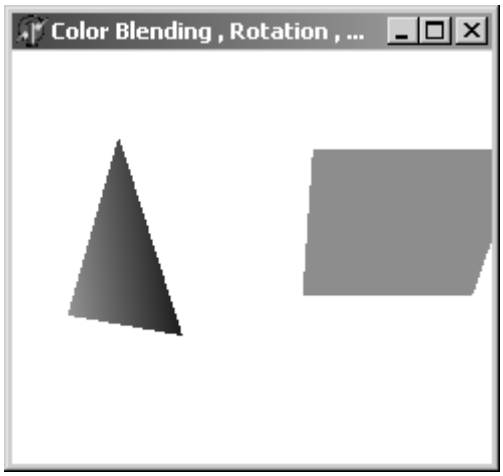
فصل پنجم

اختلاط رنگ ها و دوران



مقدمه :

در این فصل قصد داریم دو نوع رنگ آمیزی مختلف چند ضلعی ها را به همراه دوران آنها حول محورهای مختلف ، تجربه کنیم . رنگ آمیزی Flat سبب ایجاد رنگی توپر (Solid) می شود . رنگ آمیزی Smooth سبب اختلاط (Blending) سه رنگ مشخص شده در هر نقطه (راس) یک مثلث با هم گشته و جلوه ای زیبا را پدید می آورد .

نکته مهم :

تفاوت مهم برنامه این فصل با برنامه های قبلی در استفاده از کنترل Timer برای ایجاد پویا نمایی (انیمیشن) می باشد . برای اجتناب از چشمک زدن (Flickering) صفحه به هنگام پویا نمایی باید از پرچم PFD_DOUBLEBUFFER نیز در برپایی فرمت نقطه ای استفاده شود . در این حالت دیگر تابع glFlush برای نمایش دادن اشکال ترسیم شده کار ساز نبوده و باید از دستور

قدرتمند تری مانند SwapBuffers(wglGetCurrentDC); استفاده شود . پس فراموش نکنید که پرچم زیر را در واحد SPF ایجاد شده درفصول قبلی، بصورت زیر اصلاح نمایید و بجای glFlush از دستور SwapBuffers(wglGetCurrentDC); استفاده کنید :

```
dwFlags   := PFD_SUPPORT_OPENGL Or PFD_DRAW_TO_WINDOW
             Or  PFD_TYPE_RGBA
             Or PFD_DOUBLEBUFFER {to avoid flickering} ; //Flags
```

اگر موارد ذکر شده رعایت نشوند ، یا برنامه اجرا نخواهد شد و یا با چشمک زدن بسیار آزار دهنده ای در خلال ترسیم مجدد فریم ها ، مواجه خواهید شد .

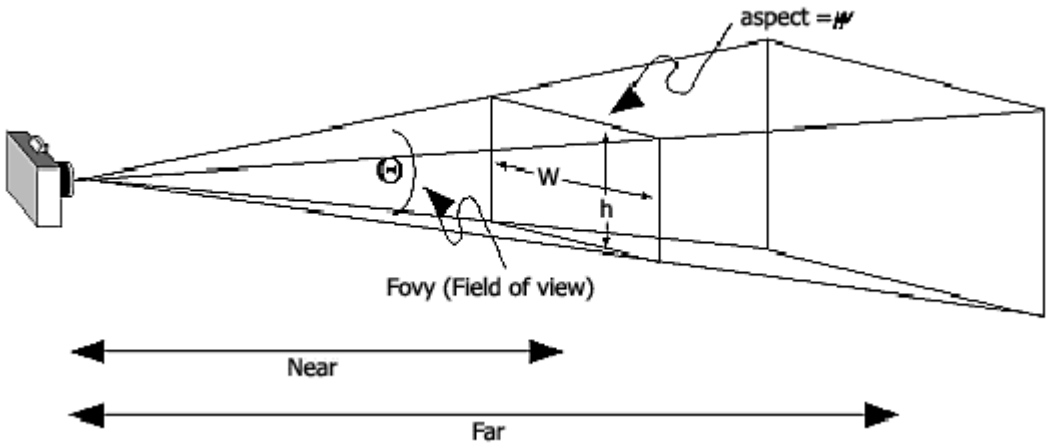
توابع مورد استفاده :

در ابتدا مروری خواهیم داشت بر توابع مورد استفاده در مثال این فصل :

تابع به فرمت زبان C	تابع به فرمت زبان دلفی
void gluPerspective(GLdouble fovy, GLdouble aspect, GLdouble zNear, GLdouble zFar);	Procedure gluPerspective(fovy: GLdouble; aspect: GLdouble; zNear: GLdouble; zFar: GLdouble); stdcall; external 'GLU32.DLL';

توضیح :

gluPerspective ماتریس تصویر کردن (Projection) سه بعدی را تنظیم می کند . fovy : زاویه میدان دید در جهت Y و به درجه می باشد . aspect یا aspect ratio میدان دید را در جهت x معین می کند . این مقدار مساوی نسبت عرض(w) به ارتفاع(h) می باشد . zNear : فاصله بیننده تا صفحه نزدیک برش ، که همواره مثبت است . zFar : فاصله بیننده تا صفحه برش دور ، که همواره منفی است .



نمایش آرگومانهای مختلف تابع gluPerspective .

ماتریس ایجاد شده توسط تابع gluPerspective در ماتریس جاری ضرب می شود . برای بارگذاری ماتریس پرسپکتیو به ماتریس جاری ، به همراه این تابع از دستور glLoadIdentity هم استفاده نمایید .

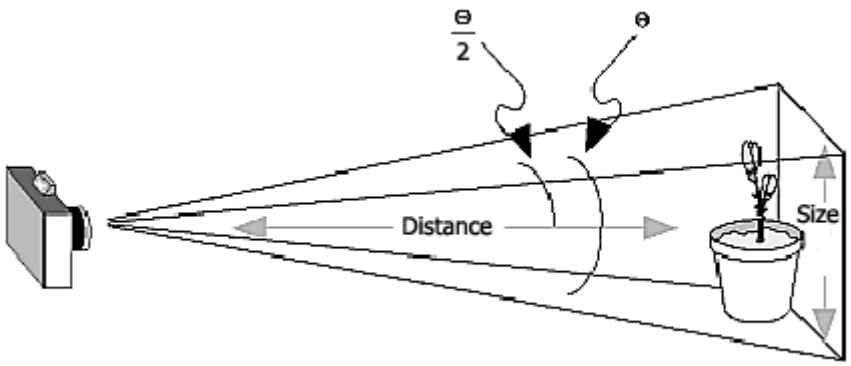
باید خاطر نشان کرد که از تابع فوق فقط برای ایجاد دیدهای متقارن ، در امتداد خط دید ، در هر دوجهت x و y ، می توان استفاده کرد . اما معمولا این همان چیزی است که شما می خواهید. بدون این تغییر ، نقطه دید در مبدا باقی مانده و خط دید به سمت محور منفی z ها خواهد بود. مطلبی را که باید بخاطر داشت این است که اینگونه تبدیلات بدون واحد می باشند و درصورت لزوم فقط باید از واحدهای یکسانی استفاده کرد .

نحوه محاسبه زاویه میدان دید :

برای محاسبه زاویه میدان دید ، می توان از روتین مثلثاتی زیر که به کمک تصویر مربوطه استخراج شده است ، استفاده کرد :

روتین به زبان C	روتین به زبان دلفی

`#define PI 3.1415926535`
`double calculateAngle(`
`  double size, double distance)`
`{`
`  double radtheta, degtheta;`
`  radtheta = 2.0 * atan2 (size/2.0, distance);`
`  degtheta = (180.0 * radtheta) / PI;`
`  return (degtheta);`
`}`
	`function calculateAngle(`
	`  size,distance : double ):double ;`
	`const PI= 3.1415926535;`
	`var`
	`  radtheta, degtheta : double;`
	`begin`
	`  // ArcTan2->> defined in math unit`
	`  radtheta := 2.0 * ArcTan2 (size/2.0, distance);`
	`  degtheta := (180.0 * radtheta) / PI;`
	`  calculateAngle := degtheta;`
	`end;`



محاسبه زاویه میدان دید به کمک علم مثلثات .

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

توضیح :

بعضی از جنبه های خاص OpenGL را کنترل می کند . پارامتر target معین می کند که کدام رفتار باید کنترل شود و پارامتر mode رفتار مورد نظر را مشخص می کند . هر دو آرگومان بوسیله مقادیر ثابت زیر تعریف شده اند :

Target :

GL_POINT_SMOOTH_HINT, GL_LINE_SMOOTH_HINT, GL_POLYGON_SMOOTH_HINT	کیفیت نقاط ، خطوط و چند ضلعی ها را در هنگام نمایش مشخص می کنند .
GL_FOG_HINT	میزان دقت در محاسبه میزان مه را معین می نماید. محاسبات مربوط به مه را به ازای هر نقطه (GL_NICEST) و یا هر راس (GL_FASTEST) انجام می دهد
GL_PERSPECTIVE_CORRECTION_HINT	کیفیت رنگ و بافت مختصات موردنظررا تعیین می کند

Mode :

GL_FASTEST	کارآمدترین گزینه را انتخاب می کند .
GL_NICEST	با کیفیت ترین گزینه را انتخاب می کند .
GL_DONT_CARE	هیچگونه برتری را تعیین نمی کند .

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

توضیح :

glShadeModel آرگومانهای ورودی آن GL_SMOOTH و GL_FLAT هستند که شرح آنها در قسمت مقدمه این فصل آورده شد . مقدار پیش فرض GL_SMOOTH است .

تابع به فرمت زبان C	تابع به فرمت زبان دلفی
void glDepthFunc(GLenum *func*);	Procedure glDepthFunc(func: GLenum); stdcall; external 'OPENGL32.DLL';

توضیح :

glDepthFunc نقاط و اشیایی را رندر می کند که نسبت بـه مقـادیر z بـافر ، حالـت func را داشـته باشند . func مساوی مقادیر زیر است :

GL_LESS : کمتـر (مقـدار پـیش فـرض) ، GL_LEQUAL : کمتـر یـا مسـاوی ، GL_EQUAL : مسـاوی ، GL_GREATER : بیشتر ، GL_NOTEQUAL : مخالف ، GL_GEQUAL : بزرگتر یـا مسـاوی ، GL_ALWAYS : همواره پذیرفته می شود و GL_NEVER هرگز پذیرفته نمی شود .

این تابع مقدار z نقطه در حال رسم را بـا مقـدار z موجـود در عمـق بـافر مقایسـه مـی کنـد و در صورت برآورده شدن شرط مقایسه ، نقطه نمایش داده خواهد شد و یا خیـر... . البتـه ایـن حالـت توسط دستور glEnable(GL_DEPTH_TEST) فعال می شود .

تابع به فرمت زبان C	تابع به فرمت زبان دلفی
void glClearDepth(GLclampd depth);	Procedure glClearDepth(depth: Glclampd); stdcall; external 'OPENGL32.DLL';

توضیح :

glClearDepth مقدار پاک شدن عمق بافر را معین می سازد . این تابع مقدار عمقی را مشخص می کند که با تابع glClear برای پاک کردن عمق بافر بکار می رود . آرگومان ورودی آن بـین صـفر و یک قرار دارد . مقدار پیش فرض یک می باشد .

تابع به فرمت زبان C	تابع به فرمت زبان دلفی
void glRotatef(GLfloat angle, GLfloat x, GLfloat y, GLfloat z);	Procedure glRotatef(angle: GLfloat; x: GLfloat; y: GLfloat; z: GLfloat); stdcall; external 'OPENGL32.DLL';

توضیح :

glRotatef ماتریس جاری را در ماتریس دوران ضرب مـی کنـد . angle : زاویـه دوران در خـلاف جهت عقربه های ساعت و به درجه می باشد . x ، Y و z مختص های بـرداری هسـتند کـه دوران حول آن صورت می گیرد . بدین وسیله کل صحنه دوران خواهد یافت . زاویه دوران بین صـفر و

۳۹۵ قرار دارد . همانطور که در فصول قبلی ذکر گردید ، از تابع glLoadIdentity برای تنظیم مجـدد دید می توان استفاده کرد تا نتایج غیر منتظره ای مشاهده نگردد .

تذکر :

برای اجرای برنامه زیر لازم است که یک کنترل Timer را روی فرم برنامه خـود قـرار دهیـد و در زیر روال رخداد مربوط به آن کد ارائه شده را بنویسید .

برنامه فصل ... :

```
unit Ch05;

interface

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

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

var
  Form1: TForm1;
  rtri: GLfloat;      // Angle For The Triangle
  rquad: GLfloat;     // Angle For The Quad
  f_Hdc : LongInt;

implementation

{$R *.DFM}

function calculateAngle(size,distance : double ):double ;
const PI= 3.1415926535;
var
  radtheta, degtheta : double;
begin
  // defined in math math unit
  radtheta := 2.0 * ArcTan2 (size/2.0, distance);
  degtheta := (180.0 * radtheta) / PI;
  calculateAngle := degtheta;
end;
```

```
procedure InitGL();       // All Setup For OpenGL Goes Here
begin
  glShadeModel(GL_SMOOTH);          // Enables Smooth Color Shading
  glClearColor(1.0, 1.0, 1.0, 0.5);    // Black Background
  glEnable(GL_DEPTH_TEST);           // Enables Depth Testing
  glDepthFunc(GL_LESS);                        // The Type Of Depth Test To Do
  glClearDepth(1.0);                 // Depth Buffer Setup
  //Realy Nice perspective calculations
  glHint(GL_PERSPECTIVE_CORRECTION_HINT,GL_NICEST);
end;

procedure DrawGLScene();    // Here's Where We Do All The Drawing
begin
  // Clear The Screen And The Depth Buffer
  glClear(GL_COLOR_BUFFER_BIT or GL_DEPTH_BUFFER_BIT);
  glLoadIdentity();                  // Reset The View
  glTranslatef(-1.5,0.0,-6.0);
  glRotatef(rtri,0.0,1.0,0.0);       // Rotate The Triangle On The Y axis
  glBegin(GL_POLYGON);
    glColor3f(1.0,0.0,0.0);          // Set The Color
    glVertex3f( 0.0, 1.0, 0.0);
    glColor3f(1.0,1.0,0.0);          // Set The Color
    glVertex3f(-1.0,-1.0, 0.0);
    glColor3f(0.0,1.0,1.0);          // Set The Color
    glVertex3f( 1.0,-1.0, 0.0);
  glEnd();
   glLoadIdentity();
  glTranslatef(1.5,0.0,-6.0);
  glColor3f(0.5,0.5,1.0);            // Set The Color To Blue One Time Only
  glRotatef(rquad,1.0,0.0,0.0);      // Rotate The Quad On The X axis
  glBegin(GL_QUADS);
    glVertex3f(-1.0, 1.0, 0.0);   glVertex3f( 1.0, 1.0, 0.0);
    glVertex3f( 1.0,-1.0, 0.0);   glVertex3f(-1.0,-1.0, 0.0);
  glEnd();

  //glFlush(); {does not work with PFD_DOUBLEBUFFER}
   SwapBuffers(f_Hdc); { works with PFD_DOUBLEBUFFER }
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  f_Hdc := GetDC(handle);
  SetDCPixelFormat(f_Hdc,24,32);// Create a rendering context.
  InitGL;
end;

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

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

```
procedure TForm1.FormResize(Sender: TObject);
 var  degAngle, gldAspect : GLdouble;
begin
  wglMakeCurrent(f_Hdc,hrc);
  gldAspect := Width / Height;
  degAngle:= calculateAngle(Height,350);
  // Reset The Current Viewport And Perspective Transformation
  glViewport(0 , 0, Width, Height);
  glMatrixMode(GL_PROJECTION);       // Select The Projection Matrix
  glLoadIdentity();                  // Reset The Projection Matrix
  // Calculate The Aspect Ratio Of The Window
  gluPerspective(degAngle,gldAspect,0.1,100.0);
  glMatrixMode(GL_MODELVIEW);        // Select The Modelview Matrix
  glLoadIdentity();                  //Reset The Modelview Matrix

  InvalidateRect(Handle, nil, False);// DrawGLScene;  Draw the scene.
end;

procedure TForm1.Timer1Timer(Sender: TObject);
begin
  rtri := (rtri + 5.5)  ;   // Increase The Rotation Variable For The Triangle
  rquad := (rquad - 5.5) ;  // Decrease The Rotation Variable For The Quad

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

end.
```