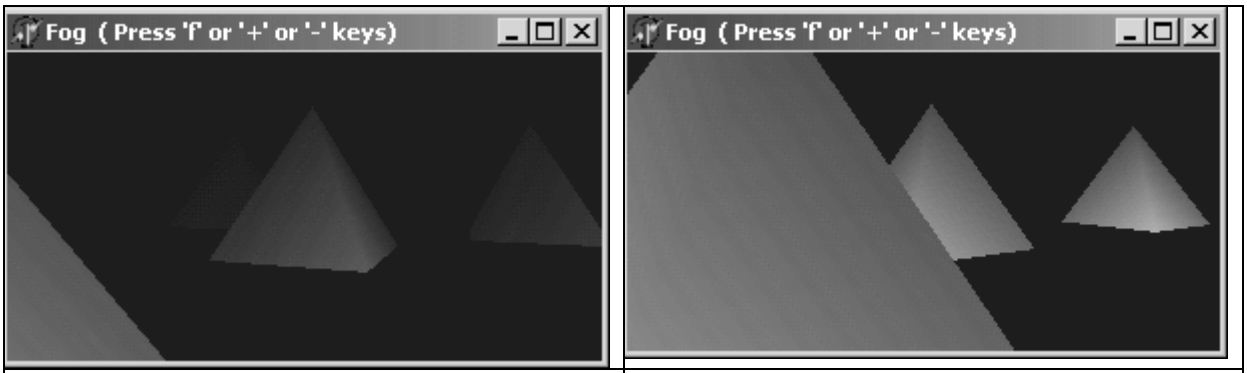
فصل شانزدهم

افزودن مه به صحنه



قبل از افزودن مه	پس از افزودن مه

مقدمه :

برای ایجاد تصاویری واقع گرایانه تر ، جلوه ی ویژه ای در OpenGL تعبیه شده است به نام ((مه)) . توسط آن علاوه بر مه می توان جلوه های ویژه دیگری مانند بخار ، غبار و دود را نیز شبیه سازی کرد . در این فصل به چگونگی استفاده از این جلوه ویژه پرداخته خواهد شد .

الگوریتم نمایش مه در OpenGL :

۱- در ابتدا باید آرایه حاوی رنگ مه را ایجاد نمود .

۲- در ادامه حالت مه بایدتنظیم شود، این کار توسط دستور glFogi(GL_FOG_MODE, GL_EXP2); صورت می گیرد .

۳- رنگ انتخاب شده توسط فرمان glFogfv(GL_FOG_COLOR, fogColor); اعمال می گردد .

۴- سپس با استفاده از دستور glFogf(GL_FOG_DENSITY, g_FogDensity); ضخامت یا چگالی مه تنظیم می شود .

۵- متعاقبا مکان آغازین و انتهایی مه (عمق مه) توسط فرامین زیر تعیین می گردند.

```
glFogf(GL_FOG_START, 0);
glFogf(GL_FOG_END, 10.0);
```

و نهایتا با استفاده از دستور glEnable(GL_FOG); مه را فعال می کنیم .

مروری بر توابع :

تابع به فرمت زبان C	تابع به فرمت زبان دلفی

void glFogf(GLenum pname, GLfloat param);
 	
void glFogi(GLenum pname, GLint param);
 	
void glFogfv(GLenum pname,
const GLfloat *params);
 	
void glFogiv(GLenum pname,
const GLint * params);
	Procedure glFogf(pname: GLenum;
	param: GLfloat);
	stdcall; external 'OPENGL32.DLL';
	Procedure glFogfv(pname: GLenum;
	params: PGLfloat);
	stdcall; external 'OPENGL32.DLL';
	Procedure glFogi(pname: GLenum;
	param: GLint);
	stdcall; external 'OPENGL32.DLL';
	Procedure glFogiv(pname: GLenum;
	params: PGLint);
	stdcall; external 'OPENGL32.DLL';

توضیح :

این توابع پارامترهای مه را تنظیم می کنند .

آرگومانها :

Pname :

در حالت glFogf و glFogi بیانگر تک مقدار تعیین کننده پارامتر مه است و در حالت glFogfv و glFogiv بیانگر پارامتر مه است . در کلیه توابع فوق مقادیر زیر برای pname مجاز هستند :

GL_FOG_MODE :

در این حالت آرگومان *params* عددی است که بیانگر معادله مربوطه برای محاسبات مه است . سه ثابت سمبولیک در این حالت قابل قبول هستند : GL_LINEAR, GL_EXP, and GL_EXP2

معادلات مربوطه به این ثوابت در جدول زیر ارائه شده اند :

ثابت	معادله محاسبه ضریب اختلاط مه ( )
GL_LINEAR	$f = \frac{end - z}{end - start}$
GL_EXP	$f = e^{(-density \cdot z)}$
GL_EXP2	$f = e^{(-density \cdot z)^2}$

: GL_FOG_DENSITY

در حالت فوق آرگومان *params* عددی است که چگالی مه را تعیین می کند . تنها مقادیر غیر منفی قابل قبول هستند . مقدار پیش فرض چگالی مه ، یک می باشد .

: GL_FOG_START و GL_FOG_END :

در حالت های فوق آرگومان *params* عددی است که بیانگر فواصل نزدیک و دور بکار گرفته شده در معادله مه خطی می باشد . مقدار پیش فرض آنها صفر و یک است .

: GL_FOG_INDEX

در این حالت آرگومان *params* عددی است که اندیس رنگی مه را تعیین می کند . مقدار پیش فرض آن صفر است .

توابع glFogfv و glFogiv علاوه بر ثابت های فوق ، ثابت GL_FOG_COLOR را نیز می پذیرند :

: GL_FOG_COLOR

در این حالت آرگومان *params* حاوی آرایه ای از چهار عدد است (RGBA) که بیانگر رنگ مه می باشند . مقدار پیش فرض آن (0,0,0,0) است .

برنامه فصل :

برای اجرای این برنامه به یک کنترل تایمر با Interval مساوی ۶۰ نیاز دارید.

```
unit Ch16;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes,
  Graphics, Controls, Forms, Dialogs, SPF,OpenGL, ExtCtrls ;
type
  TForm1 = class(TForm)
```

```
    Timer1: TTimer;
    procedure FormResize(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FormPaint(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure FormKeyPress(Sender: TObject; var Key: Char);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;

  f_Hdc : LongInt;

implementation
{$R *.dfm}
var
  g_FogDensity : single = 0.2;  // This holds how thick the fog is
  bFog : Boolean = true;// This holds if we want fog ON of OFF
  rotY : single = 0.0;
  // We chose blue for our fog color.
  // Let's make the Fog Color BLUE
  fogColor : array[0..3] of single= (0.0,0.0,1.0,1.0);

///////////////// CREATE PYRAMID \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\*
///
///   This creates a pyramid with the center being (X, Y, Z).
///   The width and height depend on the WIDTH and HEIGHT passed in
///
///////////////// CREATE PYRAMID \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\*
procedure CreatePyramid( x,  y,  z : single; width, height : Integer);
begin
   glBegin(GL_TRIANGLES);
      // These vertices create the Back Side
      // Top point
      glColor3ub(255, 0, 0);   glVertex3f(x, y + height, z);
      // Bottom left point
      glColor3ub(0, 255, 255); glVertex3f(x - width, y - height, z - width);
      // Bottom right point
      glColor3ub(255, 0, 255); glVertex3f(x + width, y - height, z - width);

      // These vertices create the Front Side
      // Top point
      glColor3ub(255, 0, 0);   glVertex3f(x, y + height, z);
      // Bottom right point
      glColor3ub(0, 255, 255); glVertex3f(x + width, y - height, z + width);
      // Bottom left point
      glColor3ub(255, 0, 255); glVertex3f(x - width, y - height, z + width);

      // These vertices create the Front Left Side
      // Top point
      glColor3ub(255, 0, 0);   glVertex3f(x, y + height, z);
      // Front bottom point
      glColor3ub(255, 0, 255); glVertex3f(x - width, y - height, z + width);
      // Bottom back point
```

```
    glColor3ub(0, 255, 255); glVertex3f(x - width, y - height, z - width);

    // These vertices create the Front right Side
    // Top point
    glColor3ub(255, 0, 0);   glVertex3f(x, y + height, z);
    // Bottom back point
    glColor3ub(255, 0, 255); glVertex3f(x + width, y - height, z - width);
    // Front bottom point
    glColor3ub(0, 255, 255); glVertex3f(x + width, y - height, z + width);

  glEnd();

        // Now, when we want to render a different shape,
        // we just start a whole new glBegin() and glEnd();

   glBegin(GL_QUADS);
    // These vertices create the bottom of the pyramid
    // Front left point
    glColor3ub(0, 0, 255); glVertex3f(x - width, y - height, z + width);
    // Front right point
    glColor3ub(0, 0, 255); glVertex3f(x + width, y - height, z + width);
    // Back right point
    glColor3ub(0, 0, 255); glVertex3f(x + width, y - height, z - width);
    // Back left point
    glColor3ub(0, 0, 255); glVertex3f(x - width, y - height, z - width);
   glEnd();
end;

procedure RenderScene();
begin
// Clear The Screen And The Depth Buffer
        glClear(GL_COLOR_BUFFER_BIT or GL_DEPTH_BUFFER_BIT);
        glLoadIdentity();     // Reset The matrix
// This determines where the camera's position and view is
        gluLookAt(0, 0, 6,     0, 0, 0,     0, 1, 0);
// Rotate the pyramids along the Y axis
        glRotatef(rotY, 0.0, 1.0, 0.0);
// This creates a 3D pyramid in the center at (0, 0, 0)
        CreatePyramid(0, 0, 0, 1, 1);
// This creates a 3D pyramid back and to the right of the middle pyramid
        CreatePyramid(3, 0, -3, 1, 1);
// This creates a 3D pyramid back and to the left of the middle pyramid
        CreatePyramid(-3, 0, -3, 1, 1);
// This creates a 3D pyramid in right front of the camera
        CreatePyramid(0, 0, 5, 1, 1);
// Below we just increment our rotation degree by a small amount
// each frame. This increase the rotation of the cube along each axis.
  rotY :=rotY+ 2.6;  // Increment the Y rotation to increase the angle
// Swap the backbuffers to the foreground
  SwapBuffers(f_Hdc);
end;

procedure InitializeOpenGL();
begin
  glClearColor(0,0 , 1, 1);
  glEnable(GL_DEPTH_TEST); // Enables Depth Testing

  glFogi(GL_FOG_MODE, GL_EXP2);      // Fog Mode
```

```
  // We pass in our array to set the fog color
  glFogfv(GL_FOG_COLOR, @fogColor);// Set Fog Color
  // The density is default 1.0,
  //but we will start with 0.2 (g_FogDensity).
  glFogf(GL_FOG_DENSITY, g_FogDensity);// How Dense Will The Fog Be
  // The "FOG_HINT" is how accurate the fog is calculate.
  // We don't really care,so we say so :)  GL_DONT_CARE let's openGL
  //choose either per vertex or per pixel fog.
  // Otherwise, we have the option to choose
  //GL_NICEST or GL_FASTEST if desired.
  glHint(GL_FOG_HINT, GL_DONT_CARE); // The Fog's calculation accuracy
  glFogf(GL_FOG_START, 0);  // Fog Start Depth
  glFogf(GL_FOG_END, 10.0); // Fog End Depth
  // To turn fog off, you can call glDisable(GL_FOG);
  glEnable(GL_FOG); // This enables our OpenGL Fog
end;

procedure TForm1.FormResize(Sender: TObject);
begin
   wglMakeCurrent(f_Hdc,hrc); //activate the RC
  if (height=0) then  // Prevent A Divide By Zero error
        height:=1;    // Make the Height Equal One
  glViewport(0,0,width,height); // Make our viewport the whole window
  // We could make the view smaller inside
  // Our window if we wanted too.
  // The glViewport takes (x, y, width, height)
  // This basically means, what our our drawing boundries
  glMatrixMode(GL_PROJECTION);  // Select The Projection Matrix
  glLoadIdentity();            // Reset The Projection Matrix
  // Calculate The Aspect Ratio Of The Window
  gluPerspective(45.0, width / height, 0.1, 150.0);
  glMatrixMode(GL_MODELVIEW);  // Select The Modelview Matrix
  glLoadIdentity();        // Reset The Modelview Matrix
  InvalidateRect(Handle, nil, False);// DrawGLScene;  Draw the scene.
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.
 InitializeOpenGL();
end;

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

procedure TForm1.Timer1Timer(Sender: TObject);
begin
  Application.ProcessMessages; //DoEvents
  InvalidateRect(Handle, nil, False);//   DrawScene();
```

```
end;

procedure TForm1.FormKeyPress(Sender: TObject; var Key: Char);
begin

if lowercase(key)='f'  then
  begin
     bFog := not bFog;   // This turns our bool to the opposite value
     if bFog then           // If bFog is true
          glEnable(GL_FOG)  // Turn on fog
     else
       glDisable(GL_FOG);   // Else turn OFF the fog
  end;

 if key='-' then
   begin
    // Decrease the fog density
    g_FogDensity := g_FogDensity - 0.015;
    // How Dense Will The Fog Be
    glFogf(GL_FOG_DENSITY, g_FogDensity);
    // Make sure we don't go below 0
    if(g_FogDensity < 0) then g_FogDensity := 0;
   end;

 if key='+' then
   begin
    g_FogDensity :=g_FogDensity+ 0.015;          // Increase the fog density
    glFogf(GL_FOG_DENSITY, g_FogDensity); // How Dense Will The Fog Be
    // Make sure we don't go above 1
    if(g_FogDensity > 1) then g_FogDensity := 1;
   end;

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

end.
```