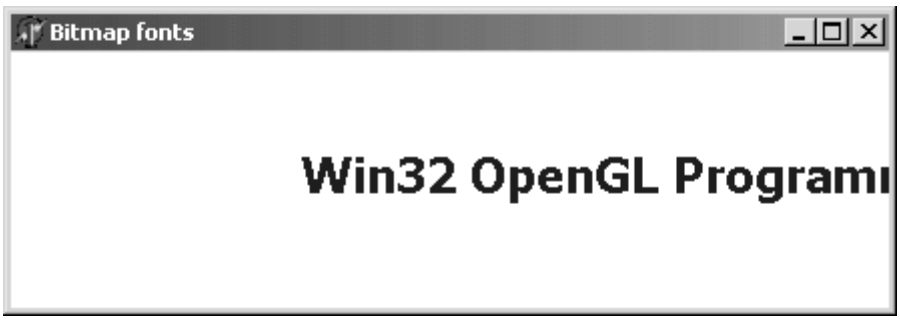
فصل چهاردهم

بکار گیری قلم های بیت مپی



مقدمه :

اگر تا کنون سعی کرده اید که متنی را در هنگام اجرای برنامه OpenGL خود نمایش دهید ، احتمالا ناامید شده اید! در این فصل طرز استفاده از تمام فونت های نصب شده روی کامپیوترتان را خواهید آموخت (فونت های (*.ttf) True Type) . با استفاده از روتین *glPrint* که ما آنرا در این فصل توسعه خواهیم داد ، می توان هر نوع متنی را در هنگام اجرای برنامه OpenGL ، نمایش داد . برای ایجاد این روتین ازتوابع *wgl* که در فصول قبلی کاملا توضیح داده شدند، استفاده می شود . بنابراین این تابع قابل انتقال به سایر سیستم عامل ها نمی باشد و فقط در ویندوز قابل استفاده است .

محدودیت های نمایش متن در OpenGL :

در نگارش فعلی OpenGL که توسط مایکروسافت ارائه شده است ، نمی توان از توابع GDI ویندوز که برای ترسیم متن روی صفحه بکار می روند، در حالت فرمت نقطه ای با بافر دوگانه ،

استفاده کرد . از تابع *TextOut* ویندوز در OpenGL ، تنها هنگامی می توان استفاده کرد که فرمت نقطه ای تک بافر انتخاب شده باشد . به همین جهت از تابع wglUseFontBitmaps برای رفع این محدودیت می توان بهره برد .

الگوریتم نمایش متن ۲ بعدی در OpenGL :

برای نمایش یک متن ابتدا باید تعدادی لیست نمایشی برای ذخیره سازی کاراکترها تولید شود ، مانند base=glGenLists(96); . سپس توسط تابع CreateFont ویندوز ، فونت دلخواه را انتخاب می کنیم . برای ارتباط دادن این فونت با *DC* جاری باید از تابع SelectObject استفاده نمود . تابع wglUseFontBitmaps همانگونه که در فصول قبلی نیز ذکر گردید ، یک مجموعه لیست نمایشی از کاراکترهای بیت مپ گونه ایجاد می کند . کاراکترها از *DC* معینی اخذ می شوند که حاوی اطلاعات فونت جاری است . هنگام پایان برنامه ، حتما این تعداد لیست نمایشی را حذف کنید تا ویندوز را دچار کمبود منابع نسازید .

پس از انتخاب فونت دلخواه و ایجاد تعدادی کاراکتر ، نوبت به نمایش متن مورد نظر می رسد . ابتدا توسط دستور glPushAttrib(GL_LIST_BIT); از تاثیر گذاری سایر لیست های نمایشی در حال استفاده در برنامه ، روی لیست نمایشی کاراکترها ، جلوگیری می نمائیم . در ادامه با فراخوانی تابع glListBase ، مکان کاراکتر مورد نظر خود را در لیست نمایشی کاراکترهای ایجاد شده ، به OpenGL اعلام می کنیم . سپس توسط تابع glCallLists متن مورد نظر خود را ، روی صفحه نمایش می دهیم . در انتها با استفاده از دستور glPopAttrib به تنظیمات قبلی OpenGL بر می گردیم .

توسط تابع glRasterPos2f(x,y); می توان موقعیت متن را روی صفحه ، تعیین کرد . مرکز صفحه (0,0) بوده و در این حالت موقعیت z وجود ندارد .

مروری بر توابع :

تابع (ویندوز) به فرمت زبان C
HFONT CreateFont(int nHeight, int nWidth, int nEscapement, int nOrientation, int fnWeight, DWORD fdwItalic, DWORD fdwUnderline, DWORD fdwStrikeOut, DWORD fdwCharSet, DWORD fdwOutputPrecision, DWORD fdwClipPrecision, DWORD fdwQuality, DWORD fdwPitchAndFamily, LPCTSTR lpszFace);

توضیح :

فونتی منطقی را که توسط هر وسیله ای می تواند بکار گرفته شود ، با مشخصات معینی ایجاد می کند .

آرگومان nHeight : اندازه و ارتفاع فونت است . اگر اندازه فونت منفی باشد ، به این معنا است که ما از ویندوز درخواست می کنیم تا فونتی را برمبنای ارتفاع کاراکتر بیابد و اگر این عدد مثبت باشد ، فونتی برمبنای ارتفاع سلول ، انتخاب خواهد شد .

nWidth : عرض سلولی که کارکتر در آن نمایش داده می شود ، می باشد . اگر مساوی صفر قرار گیرد ، ویندوز از مقدار پیش فرض استفاده خواهد کرد .

nEscapement : زاویه بین خط فرضی که متن روی آن رسم شده است و محور X می باشد (به درجه) و سبب چرخش فونت خواهد شد . این مورد تنها در حالت های 0 ، 90 ، 180 و 270 درجه نتایج خوبی را ارئه می دهد .

nOrientation : در *win95/98/ME* با مقدار nEscapement یکی می باشد .

fnWeight : متغیری است که از ۰ تا ۱۰۰۰ تغییر می کند و بیانگر ضخامت فونت می باشد . در ضمن ثوابت زیر نیز برای آن معمول هستند :

Value	Weight
FW_DONTCARE	0
FW_THIN	100
FW_EXTRALIGHT	200
FW_ULTRALIGHT	200
FW_LIGHT	300
FW_NORMAL	400
FW_REGULAR	400
FW_MEDIUM	500
FW_SEMIBOLD	600
FW_DEMIBOLD	600
FW_BOLD	700
FW_EXTRABOLD	800
FW_ULTRABOLD	800
FW_HEAVY	900
FW_BLACK	900

fdw*Italic* ، fdwUnderline و ~~fdwStrikeOut~~ : می توانند *true* و یا *false* باشند .

fdwCharSet : یکی از مقادیر زیر می تواند باشد :

DEFAULT_CHARSET ، CHINESEBIG5_CHARSET ، BALTIC_CHARSET ، ANSI_CHARSET
GREEK_CHARSET ، GB2312_CHARSET ، EASTEUROPE_CHARSET

RUSSIAN_CHARSET ، OEM_CHARSET ، MAC_CHARSET ، HANGUL_CHARSET
TURKISH_CHARSET ، SYMBOL_CHARSET ، SHIFTJIS_CHARSET

fdwOutputPrecision : دقت خروجی را برمبنای خواص درخواست شده معین می کند و یکی از مقادیر زیر را می تواند داشته باشد .

OUT_CHARACTER_PRECIS ، OUT_DEFAULT_PRECIS ، OUT_DEVICE_PRECIS
OUT_OUTLINE_PRECIS ، OUT_RASTER_PRECIS ،OUT_STRING_PRECIS
OUT_STROKE_PRECIS ، OUT_TT_ONLY_PRECIS ، OUT_TT_PRECIS

fdwClipPrecision : دقت ناحیه برش را تعیین می کند . بدین وسیله می توان دقت برش کاراکتری را که خارج از ناحیه برش قرار گرفته است ، معین نمود و یکی از ثوابت زیر می تواند باشد .

CLIP_DEFAULT_PRECIS ، CLIP_CHARACTER_PRECIS، CLIP_STROKE_PRECIS
CLIP_MASK، CLIP_EMBEDDED ،CLIP_LH_ANGLES ، CLIP_TT_ALWAYS

fdwQuality : کیفیت نمایش فونت را مشخص می کند و یکی از ثوابت زیر می تواند باشد .

ANTIALIASED_QUALITY ، DEFAULT_QUALITY ، DRAFT_QUALITY،NONANTIALIASED_QUALITY
PROOF_QUALITY

fdwPitchAndFamily : خانواده فونت درخواستی را معین می نماید . هنگامی بکار گرفته می شود که فونت درخواست شده موجود نباشد و یکی از ثوابت زیر می تواند باشد .

FF_DECORATIVE ، FF_DONTCARE ، FF_MODERN ، FF_ROMAN ، FF_SCRIPT، FF_SWISS
VARIABLE_PITCH ،FIXED_PITCH،DEFAULT_PITCH

lpszFace : نام فونت مورد نظر می باشد . برای بدست آوردن نام دقیق فونت های نصب شده در ویندوز ساده ترین راه استفاده از منوی فونت در MS-Word و یا wordpad ویندوز می باشد .

خروجی این تابع دستگیره ای(handle) است به فونت منطقی ایجاد شده .

تابع (ویندوز) به فرمت زبان C

```
HGDIOBJ SelectObject(
  HDC hdc,          // handle to DC
  HGDIOBJ hgdiobj   // handle to object
);

BOOL DeleteObject( HGDIOBJ hObject   // handle to graphic object
 );
```

توضیح :

SelectObject شیء مشخص شده را به hDC ارتباط می دهد . در این برنامه شیءایی که باید به DC جاری ارتباط یابد ، توسط تابع خلق فونت ایجاد می شود . خروجی تابع ، دستگیره ای است به شیء ایی که جایگزین آن شده است .

DeleteObject تمام منابع منتسب به شیء را آزاد می کند . در صورت موفقیت خروجی آن غیر صفر می باشد .

تابع به فرمت زبان C
BOOL wglUseFontBitmaps(HDC hdc, DWORD first, DWORD count, DWORD listBase);

توضیح :

تبدیلگر فونت سیستم برای استفاده در OpenGL است . این تابع مجموعه ای از لیست های نمایشی بیت مپی را ایجاد می کند که برای استفاده در *RC* جاری OpenGL بکار می روند . برای مثال ، توسط آن می توان اشیاء ترسیم شده را به همراه یک برچسب متنی نمایش داد .

آرگومان hdc : بیانگر *DC* فونت انتخاب شده جاری است .

first : اولین علامت و کاراکتر مورد استفاده از لیست های نمایشی بیت مپی کاراکترها را معین می کند .

count : به تعداد این آرگومان ، لیست نمایشی برای هر کاراکتر و علامت تولید می شود .

listBase : لیست نمایشی آغازگر را معین می کند .

اگر تابع کارش را با موفقیت به پایان برساند ، خروجی آن *true* خواهد بود .

برای مثال دستور ;(wglUseFontBitmaps(hdc,32,96,base ، ۹۶ لیست نمایشی که از کاراکتر ۳۲ (فاصله(Space)) شروع می شوند ، می سازد .

تابع به فرمت زبان C	تابع به فرمت زبان دلفی
void glPushAttrib(GLbitfield mask); void glPopAttrib(void);	Procedure glPushAttrib(mask: GLbitfield); stdcall; external 'OPENGL32.DLL'; Procedure glPopAttrib; stdcall; external 'OPENGL32.DLL';

توضیح :

glPushAttrib معین می کند که کدام یک از متغیرهای حالت باید درپشته خواص ذخیره شوند . آرگومان ورودی آن مجموعه ای از ثوابت سمبولیک می باشد که در ذیل ارائه شده اند . برای مثال ثابت GL_LIGHTING_BIT سبب ذخیره شدن تمام متغیرهای حالت مربوط به نورپردازی مانند رنگ ماده جاری ، نورهای نفوذی ، انتشاری ، محیطی و بازتابش شده ، مجموعه ای از نورهای

فعال شده و جهت نور نقطه ای ، می شود . از آرگومان استثنایی GL_ALL_ATTRIB_BITS می توان برای ذخیره سازی تمام حالت های تعریف شده برای این تابع استفاده کرد .

Mask Bit	Attribute Group
GL_ACCUM_BUFFER_BIT	accum-buffer
GL_ALL_ATTRIB_BITS	--
GL_COLOR_BUFFER_BIT	color-buffer
GL_CURRENT_BIT	current
GL_DEPTH_BUFFER_BIT	depth-buffer
GL_ENABLE_BIT	enable
GL_EVAL_BIT	eval
GL_FOG_BIT	fog
GL_HINT_BIT	hint
GL_LIGHTING_BIT	lighting
GL_LINE_BIT	line
GL_LIST_BIT	list
GL_PIXEL_MODE_BIT	pixel
GL_POINT_BIT	point
GL_POLYGON_BIT	polygon
GL_POLYGON_STIPPLE_BIT	polygon-stipple
GL_SCISSOR_BIT	scissor
GL_STENCIL_BUFFER_BIT	stencil-buffer
GL_TEXTURE_BIT	texture
GL_TRANSFORM_BIT	transform
GL_VIEWPORT_BIT	viewport

glPopAttrib سبب به حالت نخست در آمدن متغیرهای حالت ذخیره شده ، توسط آخرین دستور glPushAttrib می شوند .

تابع به فرمت زبان دلفی	تابع به فرمت زبان C
Procedure glRasterPos2f(x: GLfloat; y: GLfloat); stdcall; external 'OPENGL32.DLL'; Procedure glRasterPos2fv(v: PGLfloat); stdcall; external 'OPENGL32.DLL';	void glRasterPos2f(GLfloat x, GLfloat y); void glRasterPos2fv(const GLfloat *v);

توضیح :

مکان جاری تصویر (Raster) را تنظیم می کند . آرگومانهای آن مختصات و یا آرایه ای حاوی مختصات این مکان هستند . با مختصاتی که توسط این تابع ارائه می شود همانند تابع *glVertx* رفتار خواهد شد ؛ آنها توسط ماتریس های *modelView* و *projection* جاری تحت تاثیر قرا خواهند گرفت .

برنامه فصل :

برای اجرای برنامه به یک کنترل تایمر نیز نیاز می باشد .

```
unit Ch14;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics,
  Controls, Forms,Dialogs ,OpenGL , SPF, ExtCtrls ;
type
  TForm1 = class(TForm)
    Timer1: TTimer;
    procedure FormResize(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure FormPaint(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
        f_Hdc : LongInt;
implementation
{$R *.dfm}
var
  base : GLuint;   // Base Display List For The Font Set
  cnt1 : GLfloat;  // 1st Counter Used To Move Text & For Coloring
  cnt2 : GLfloat;  // 2nd Counter Used To Move Text & For Coloring
  h_DC : HDC;      // Private GDI Device Context

procedure BuildFont;  // Build Our Bitmap Font
var font: HFONT;      // Windows Font ID
begin
  h_DC := Form1.Canvas.Handle ;
  base := glGenLists(96);     // Storage For 96 Characters
```

```
  font := CreateFont(-24,     // Height Of Font
                     0,       // Width Of Font
                     0,       // Angle Of Escapement
                     0,       // Orientation Angle
                     FW_BOLD, // Font Weight
                     0,       // Italic
                     0,       // Underline
                     0,       // Strikeout
                     ANSI_CHARSET,      // Character Set Identifier
                     OUT_TT_PRECIS,     // Output Precision
                     CLIP_DEFAULT_PRECIS,// Clipping Precision
                     ANTIALIASED_QUALITY,// Output Quality
                     FF_DONTCARE or DEFAULT_PITCH, // Family And Pitch
                     'Tahoma');                    // Font Name

  SelectObject(h_DC, font);  // Selects The Font We Want
  // Builds 96 Characters Starting At Character 32
  wglUseFontBitmaps(h_DC, 32, 96, base);
end;

procedure KillFont;           // Delete The Font
begin
  glDeleteLists(base, 96);            // Delete All 96 Characters
end;

procedure glPrint(text : pchar);    // Custom GL "Print" Routine
begin
  if (text = '') then           // If There's No Text
        Exit;                         // Do Nothing
  glPushAttrib(GL_LIST_BIT);        // Pushes The Display List Bits
  glListBase(base - 32);      // Sets The Base Character to 32
  // Draws The Display List Text
  glCallLists(length(text), GL_UNSIGNED_BYTE, text);
  glPopAttrib();                    // Pops The Display List Bits
end;

// This Will Be Called Right After The GL Window Is Created
procedure InitGL(Width: GLsizei; Height: GLsizei);
var fWidth, fHeight  : GLfloat;
begin
  glClearColor(1.0, 1.0, 1.0, 0.0);// Clear The Background Color To white
  glClearDepth(1.0);                // Enables Clearing Of The Depth Buffer
  glDepthFunc(GL_LESS);                   // The Type Of Depth Test To Do
  glEnable(GL_DEPTH_TEST);      // Enables Depth Testing
  glShadeModel(GL_SMOOTH);      // Enables Smooth Color Shading

  glMatrixMode(GL_PROJECTION);          // Select The Projection Matrix
  glLoadIdentity();                 // Reset The Projection Matrix

  fWidth := Width;
  fHeight := Height;
  // Calculate The Aspect Ratio Of The Window
  gluPerspective(45.0, fWidth/fHeight,0.1,100.0);

  glMatrixMode(GL_MODELVIEW);           // Select The Modelview Matrix

  BuildFont();                      // Build The Font
end;
```

```
procedure DrawGLScene();
begin
  // Clear The Screen And The Depth Buffer
  glClear(GL_COLOR_BUFFER_BIT or GL_DEPTH_BUFFER_BIT);
  glLoadIdentity();                  // Reset The View
  glTranslatef(0.0,0.0,-1.0);        // Move One Unit Into The Screen
  // Pulsing Colors Based On Text Position
  glColor3f(1.0*cos(cnt1),1.0*sin(cnt2),1.0-0.5*cos(cnt1+cnt2));
  // Position The Text On The Screen
  glRasterPos2f(-0.2+0.35*cos(cnt1), 0.35*(sin(cnt2)));
  glPrint('Win32 OpenGL Programming'); // Print GL Text To The Screen
  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
  if (Height=0) then Height:=1;
  // Reset The Current Viewport And Perspective Transformation
  glViewport(0, 0, Width, Height);
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();

  gluPerspective(45.0, Width/Height, 0.1, 100.0);
  glMatrixMode(GL_MODELVIEW);
  InvalidateRect(Handle, nil, False);// DrawGLScene;  Draw the scene.
end;

procedure TForm1.FormDestroy(Sender: TObject);
begin
 KillFont;
 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(Width , Height);
end;

procedure TForm1.Timer1Timer(Sender: TObject);
begin
  cnt1 := cnt1 + 0.1;                                  // Increase The First Counter
  cnt2 := cnt2 + 0.081;
  Application.ProcessMessages; //DoEvents
  InvalidateRect(Handle, nil, False);//   DrawGLScene();
end;

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

end.
```