فصل هجدهم

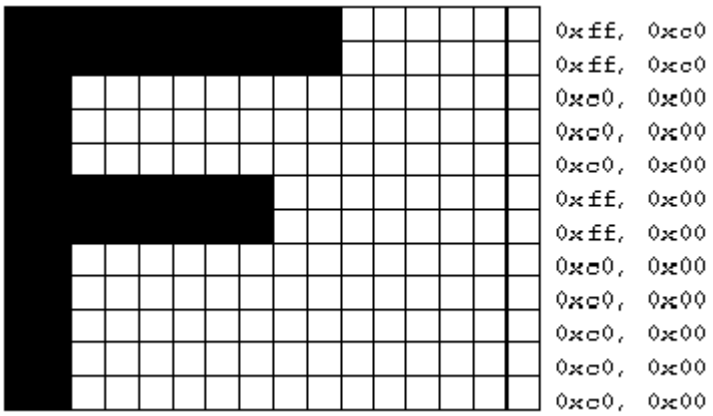
فارسی سازی OpenGL

مقدمه :

از آنجائیکه ما ایرانی ها علاقه وافری به فارسی کردن همه چیز تحت ویندوز داریم (و همچنین سیستم عامل های دیگر) ، در این فصل روش فارسی کردن OpenGL به شما آموخته خواهد شد ! توسط OpenGL می توان الگوی بیتی تمام حروف را تک تک ایجاد نمود و سپس آنها را نمایش داد . باید اعتراف کنم که الگوی بیتی حروف فارسی این فصل را از اصل برنامه های فارسی ساز تحت داس (البته باکمی تغییر) به امانت گرفته ام .

بیت مپ ها و فونت های Raster :

کلاس دیگری از داده ها که می توان آنها را توسط OpenGL رندر کرد ، بیت مپ ها مـی باشـند . بیت مپ ها آرایه ای مستطیلی از نقاط می باشـند کـه در هـر نقطـه آن یـک تـک بیـت قـرار دارد . دستورات glRasterPos و glBitmap برای قرار دادن و نمایش یک بیت مپ بـر روی صـفحه بکـار می روند . عموما بیت مپ ها بـرای نمـایش فونـت هـا کـاربرد دارنـد . بـرای مثـال کـاراکتر F را بصورت زیر می توان طراحی کرد .



داده های بیت مپ ها همواره در قطعاتی که مضربی از 8 bits هستند ذخیره می شوند . بیت هایی که بیت مپ ها را تشکیل می دهند از گوشه سمت چپ پایین شروع می شوند ، سپس ردیف پایین رسم می شود در ادامه ردیف بالایی و به همین ترتیب .

نمایش فونت ها توسط لیست های نمایشی :

یک فونت عموما از مجموعه ای از کاراکترها تشکیل می شود که هر کاراکتر یک عـدد مشخصـه (کد اسکی) را دارا است . برای مثال کد اسکی حرف A معادل ۶۵ مـی باشـد و بـرای ترسـیم آن کافی است که لیست نمایشی شماره ۶۵ فراخوانی شود، که شرح مفصل آن در فصول قبلی رفت .

مطالب ارائه شده در این فصل در مورد فارسـی نویسـی ، تنهـا ابتـدای راه را بـه شـما نشـان می دهند . شما می توانید با توجه به سلیقه و ابتکارتان مـوارد اسـتفاده جـالبی از آن را پیشـنهاد دهید .

مروری بر توابع :

تابع به فرمت زبان C	تابع به فرمت زبان دلفی
void glBitmap(GLsizei width, GLsizei height, GLfloat xorig, GLfloat yorig, GLfloat xmove, GLfloat ymove, const GLubyte *bitmap);	Procedure glBitmap(width: GLsizei; height: GLsizei; xorig: GLfloat; yorig: GLfloat; xmove: GLfloat; ymove: GLfloat; bitmap: PGLubyte); stdcall; external 'OPENGL32.DLL';

توضیح :

بیت مپی را که توسط آرگومان bitmap معین گشته ترسیم می کند . اگر محل قرار گیری آن در صفحه مجاز نباشد ، چیزی رسم نخواهد شد .

height, width : طول و عرض بیت مپ را مشخص می کنند .

yorig, xorig : موقعیت مبدا را در بیت مپ مشخص می کنند . مبدا از گوشه پایین ، سمت چپ اندازه گیری می شود .

ymove, xmove : x و y ایی هستند که به موقعیت raster جاری پس از اینکه بیت مپ رسم می شود ، اضافه می گردند .

تابع به فرمت زبان C	تابع به فرمت زبان دلفی
void glPixelMapfv(GLenum map, GLsizei mapsize, const GLfloat *values);	Procedure glPixelMapfv(map: GLenum; mapsize: GLsizei; values: PGLfloat); stdcall; external 'OPENGL32.DLL';

توضیح :

نگاشت های انتقال نقاط را برپا می سازد .

هنگامیکه داده های تصویر یا بیت مپ از حافظه به درون framebuffer منتقل می شود و یا برعکس ، می توان عملیات مختلفی را روی آن انجام داد . برای مثال تعداد اجزاء را می توان تغییر داد ؛ در حالت نرمال ، جزء قرمز بین یک و صفر قرار دارد ، اما شاید شما بخواهید که آنرا درون بازه دیگری تعریف کنید . به اینگونه عملیات ، عملیات انتقال نقطه گویند .

آرگومان Map : یک ثابت سمبولیک بوده و مقادیر مجاز آن در جدول زیر ارائه شده اند :

Map Name	Address	Value
GL_PIXEL_MAP_I_TO_I	color index	color index
GL_PIXEL_MAP_S_TO_S	stencil index	stencil index
GL_PIXEL_MAP_I_TO_R	color index	R
GL_PIXEL_MAP_I_TO_G	color index	G
GL_PIXEL_MAP_I_TO_B	color index	B
GL_PIXEL_MAP_I_TO_A	color index	A
GL_PIXEL_MAP_R_TO_R	R	R
GL_PIXEL_MAP_G_TO_G	G	G
GL_PIXEL_MAP_B_TO_B	B	B
GL_PIXEL_MAP_A_TO_A	A	A

Mapsize : اندازه نگاشتی است که تعریف می شود و باید توانی از ۲ باشد .

Values : آرایه ای از مقادیر mapsize .

تمام اجزاء رنگ را می توان بوسیله جداول lookup ، قبل از اینکه روی صفحه نمایش داده شوند ، اصلاح کرد . این جداول به شرح زیر هستند :

GL_PIXEL_MAP_I_TO_I

Lookup Index: color index
Lookup Value: color index
Initial Size: 1
Initial Value: 0.0

GL_PIXEL_MAP_S_TO_S

Lookup Index: stencil index
Lookup Value: stencil index
Initial Size: 1
Initial Value: 0.0

GL_PIXEL_MAP_I_TO_R

Lookup Index: color index
Lookup Value: R
Initial Size: 1
Initial Value: 0.0

GL_PIXEL_MAP_I_TO_G

Lookup Index: color index
Lookup Value: G
Initial Size: 1
Initial Value: 0.0

GL_PIXEL_MAP_I_TO_B

Lookup Index: color index
Lookup Value: B
Initial Size: 1
Initial Value: 0.0

GL_PIXEL_MAP_I_TO_A

Lookup Index: color index
Lookup Value: A
Initial Size: 1
Initial Value: 0.0

GL_PIXEL_MAP_R_TO_R

Lookup Index: R
Lookup Value: R
Initial Size: 1
Initial Value: 0.0

GL_PIXEL_MAP_G_TO_G

Lookup Index: G
Lookup Value: G
Initial Size: 1
Initial Value: 0.0

GL_PIXEL_MAP_B_TO_B

Lookup Index: B
Lookup Value: B
Initial Size: 1
Initial Value: 0.0

GL_PIXEL_MAP_A_TO_A

Lookup Index: A
Lookup Value: A
Initial Size: 1
Initial Value: 0.0

برای اینکه درک کنیم جداول فوق چگونه کار می کنند به مثال ساده زیر توجه کنید .
فرض کنید می خواهید جدولی از اندیس های رنگی که ۲۵۶ مدخل دارد را بوجود آورید (در حالت GL_PIXEL_MAP_I_TO_I) . اینکار توسط تابع glPixelMap صورت می گیرد . فرض کنید می خواهید اندیس های کوچکتر از ۱۰۱ را به صفر و تمام اندیس های بالاتر از ۱۰۱ را به ۲۵۵ تبدیل کنید . اینکار توسط glPixelTransfer صورت می گیرد .

تابع به فرمت زبان C	تابع به فرمت زبان دلفی
void glPixelTransferi(GLenum pname, GLint param);	Procedure glPixelTransferi(pname: GLenum; param: GLint); stdcall; external 'OPENGL32.DLL';

توضیح :

این تابع حالت های انتقال نقطه را تنظیم می کند . پارامتر pname در آن باید یکی از مقادیر جدول زیر باشد :

Pname	Type	Initial value	Valid range
GL_MAP_COLOR	Boolean	false	true/false
GL_MAP_STENCIL	Boolean	false	true/false
GL_INDEX_SHIFT	integer	0	(– ∞,∞)
GL_INDEX_OFFSET	integer	0	(– ∞,∞)
GL_RED_SCALE	float	1.0	(– ∞,∞)
GL_GREEN_SCALE	float	1.0	(– ∞,∞)
GL_BLUE_SCALE	float	1.0	(– ∞,∞)
GL_ALPHA_SCALE	float	1.0	(– ∞,∞)
GL_DEPTH_SCALE	float	1.0	(– ∞,∞)
GL_RED_BIAS	float	0.0	(– ∞,∞)
GL_GREEN_BIAS	float	0.0	(– ∞,∞)
GL_BLUE_BIAS	float	0.0	(– ∞,∞)
GL_ALPHA_BIAS	float	0.0	(– ∞,∞)
GL_DEPTH_BIAS	Float	0.0	(– ∞,∞)

برای مثال اگر می خواهید RGB را به luminance مطابق استاندارد NTSC تبدیل کنید ، باید GL_RED_SCALE را به 0.30 ، GL_GREEN_SCALE را به 0.59 و GL_BLUE_SCALE را به 0.11 تنظیم کنید .

برنامه فصل :

```
unit Ch18;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes,
  Graphics, Controls, Forms,
  Dialogs ,OpenGL , spf ;

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

var
  Form1: TForm1;

  f_Hdc : LongInt;

implementation
{$R *.dfm}

const
     OPENGL_WIDTH =24;
     OPENGL_HEIGHT =13;
var
    boxA : array[0..2] of single =(0, 0, 0);
    boxB : array[0..2] of single =(-100, 0, 0);
    boxC : array[0..2] of single =(100, 0, 0);
    boxD : array[0..2] of single =(0, 95, 0);
    boxE : array[0..2] of single =(0, -105, 0);

 farsi_raster_font : array[0..89,0..13] of GLubyte =(
// 128 - 175
($0 ,$0 ,$0 ,$0 ,$0 ,$0 ,$8 ,$1c,$8 ,$0 ,$0 ,$0 ,$0 ,$0 ),
($0 ,$0 ,$20,$30,$38,$18,$c ,$c ,$4 ,$4 ,$4 ,$4 ,$0 ,$0 ),
($0 ,$0 ,$42,$7e,$7c,$30,$18,$18,$8 ,$8 ,$8 ,$8 ,$0 ,$0 ),
($0 ,$0 ,$49,$7f,$76,$30,$18,$18,$8 ,$8 ,$8 ,$8 ,$0 ,$0 ),
($0 ,$0 ,$8 ,$10,$20,$18,$8 ,$10,$20,$21,$3e,$1c,$0 ,$0 ),
($0 ,$0 ,$0 ,$8 ,$8 ,$c ,$16,$23,$41,$49,$7f,$36,$0 ,$0 ),
($0 ,$0 ,$42,$7e,$3e,$4 ,$4 ,$4 ,$6 ,$3 ,$3 ,$1 ,$0 ,$0 ),
($0 ,$0 ,$41,$41,$63,$36,$1c,$1c,$8 ,$8 ,$8 ,$8 ,$0 ,$0 ),
($0 ,$0 ,$8 ,$8 ,$8 ,$8 ,$1c,$1c,$36,$63,$41,$41,$0 ,$0 ),
($0 ,$0 ,$18,$3c,$44,$44,$3c,$4 ,$6 ,$3 ,$3 ,$1 ,$0 ,$0 ),
($0 ,$0 ,$0 ,$0 ,$0 ,$c ,$18,$18,$18,$0 ,$0 ,$0 ,$0 ,$0 ),
($0 ,$0 ,$0 ,$0 ,$0 ,$0 ,$0 ,$0 ,$ff,$0 ,$0 ,$0 ,$0 ,$0 ),
($0 ,$0 ,$3c,$42,$40,$60,$10,$c ,$c ,$0 ,$c ,$c ,$0 ,$0 ),
($0 ,$1 ,$3e,$40,$8 ,$8 ,$8 ,$8 ,$8 ,$0 ,$0 ,$0 ,$0 ,$0 ),
($0 ,$0 ,$c ,$10,$3c,$0 ,$1 ,$1 ,$fe,$0 ,$0 ,$0 ,$0 ,$0 ),
```

($0 ,$0 ,$0 ,$0 ,$0 ,$c ,$10,$1c,$20,$c ,$10,$0 ,$0 ,$0),
($0 ,$8 ,$c ,$8 ,$8 ,$8 ,$8 ,$8 ,$8 ,$0 ,$0 ,$0 ,$0 ,$0),
($0 ,$4 ,$4 ,$4 ,$4 ,$4 ,$4 ,$4 ,$3 ,$0 ,$0 ,$0 ,$0 ,$0),
($0 ,$0 ,$0 ,$0 ,$0 ,$0 ,$40,$81,$81,$7e,$0 ,$0 ,$10,$0 ,$0),
($0 ,$0 ,$0 ,$0 ,$0 ,$0 ,$0 ,$1 ,$1 ,$fe,$0 ,$0 ,$20,$0 ,$0),
($0 ,$0 ,$0 ,$0 ,$0 ,$0 ,$40,$81,$81,$7e,$0 ,$0 ,$28,$10,$0),
($0 ,$0 ,$0 ,$0 ,$0 ,$0 ,$0 ,$1 ,$1 ,$fe,$0 ,$0 ,$28,$10,$0),
($0 ,$0 ,$0 ,$14,$0 ,$40,$81,$81,$7e,$0 ,$0 ,$0 ,$0 ,$0),
($0 ,$0 ,$0 ,$14,$0 ,$0 ,$1 ,$1 ,$fe,$0 ,$0 ,$0 ,$0 ,$0),
($0 ,$0 ,$8 ,$14,$0 ,$40,$81,$81,$7e,$0 ,$0 ,$0 ,$0 ,$0),
($0 ,$0 ,$8 ,$14,$0 ,$0 ,$1 ,$1 ,$fe,$0 ,$0 ,$0 ,$0 ,$0),
($0 ,$0 ,$0 ,$0 ,$18,$24,$2 ,$3f,$40,$80,$88,$80,$41,$3e),
($0 ,$0 ,$0 ,$0 ,$0 ,$18,$24,$3 ,$fc,$0 ,$0 ,$8 ,$0 ,$0),
($0 ,$0 ,$0 ,$0 ,$18,$24,$2 ,$3f,$40,$80,$94,$88,$41,$3e),
($0 ,$0 ,$0 ,$0 ,$0 ,$18,$24,$3 ,$fc,$0 ,$0 ,$28,$10,$0),
($0 ,$0 ,$0 ,$0 ,$18,$24,$2 ,$3f,$40,$80,$80,$80,$41,$3e),
($0 ,$0 ,$0 ,$0 ,$0 ,$18,$24,$3 ,$fc,$0 ,$0 ,$0 ,$0 ,$0),
($0 ,$0 ,$10,$0 ,$18,$24,$2 ,$3f,$40,$80,$80,$80,$41,$3e),
($0 ,$0 ,$8 ,$0 ,$0 ,$18,$24,$3 ,$fc,$0 ,$0 ,$0 ,$0 ,$0),
($0 ,$0 ,$0 ,$0 ,$4 ,$2 ,$1 ,$21,$3e,$0 ,$0 ,$0 ,$0 ,$0),
($0 ,$8 ,$0 ,$0 ,$4 ,$2 ,$1 ,$21,$3e,$0 ,$0 ,$0 ,$0 ,$0),
($0 ,$0 ,$0 ,$0 ,$0 ,$0 ,$0 ,$1 ,$1 ,$1 ,$2 ,$24,$18,$0),
($0 ,$0 ,$0 ,$2 ,$0 ,$0 ,$0 ,$1 ,$1 ,$1 ,$2 ,$24,$18,$0),
($0 ,$0 ,$2 ,$5 ,$0 ,$0 ,$0 ,$1 ,$1 ,$1 ,$2 ,$24,$18,$0),
($0 ,$0 ,$0 ,$0 ,$0 ,$0 ,$15,$15,$9a,$88,$88,$88,$70,$0),
($0 ,$0 ,$0 ,$0 ,$0 ,$0 ,$15,$15,$ea,$0 ,$0 ,$0 ,$0 ,$0),
($0 ,$0 ,$4 ,$a ,$0 ,$0 ,$15,$15,$9a,$88,$88,$88,$70,$0),
($0 ,$4 ,$a ,$0 ,$0 ,$0 ,$15,$15,$ea,$0 ,$0 ,$0 ,$0 ,$0),
($0 ,$0 ,$0 ,$0 ,$0 ,$2 ,$5 ,$29,$1e,$88,$88,$88,$70,$0),
($0 ,$0 ,$0 ,$0 ,$0 ,$6 ,$29,$31,$de,$0 ,$0 ,$0 ,$0 ,$0),
($0 ,$0 ,$2 ,$0 ,$0 ,$2 ,$5 ,$29,$1e,$88,$88,$88,$70,$0),
($0 ,$4 ,$0 ,$0 ,$0 ,$6 ,$29,$31,$de,$0 ,$0 ,$0 ,$0 ,$0),
($0 ,$20,$30,$20,$20,$26,$29,$31,$fe,$0 ,$0 ,$0 ,$0 ,$0),
// 225 - 255
($0 ,$20,$34,$20,$20,$26,$29,$31,$fe,$0 ,$0 ,$0 ,$0 ,$0),
($0 ,$0 ,$0 ,$0 ,$c ,$12,$10,$3e,$40,$80,$80,$80,$42,$3c),
($0 ,$0 ,$0 ,$0 ,$c ,$1e,$c ,$12,$21,$40,$40,$40,$21,$1e),
($0 ,$0 ,$0 ,$0 ,$0 ,$1c,$3c,$18,$e7,$0 ,$0 ,$0 ,$0 ,$0),
($0 ,$0 ,$0 ,$0 ,$0 ,$6 ,$9 ,$8 ,$ff,$0 ,$0 ,$0 ,$0 ,$0),
($0 ,$8 ,$0 ,$0 ,$c ,$12,$10,$3e,$40,$80,$80,$80,$42,$3c),
($0 ,$4 ,$0 ,$0 ,$e ,$1e,$c ,$12,$21,$40,$40,$40,$21,$1e),
($0 ,$0 ,$8 ,$0 ,$0 ,$1c,$3c,$18,$e7,$0 ,$0 ,$0 ,$0 ,$0),
($0 ,$4 ,$0 ,$0 ,$0 ,$6 ,$9 ,$8 ,$ff,$0 ,$0 ,$0 ,$0 ,$0),
($0 ,$4 ,$0 ,$0 ,$6 ,$49,$89,$87,$7e,$0 ,$0 ,$0 ,$0 ,$0),
($0 ,$4 ,$0 ,$0 ,$6 ,$9 ,$9 ,$7 ,$fe,$0 ,$0 ,$0 ,$0 ,$0),
($0 ,$0 ,$a ,$0 ,$0 ,$2 ,$5 ,$45,$83,$81,$81,$42,$3c,$0),
($0 ,$a ,$0 ,$0 ,$6 ,$9 ,$9 ,$7 ,$fe,$0 ,$0 ,$0 ,$0 ,$0),
($2 ,$4 ,$8 ,$8 ,$4 ,$2 ,$41,$81,$7e,$0 ,$0 ,$0 ,$0 ,$0),
($0 ,$3 ,$4 ,$8 ,$8 ,$6 ,$1 ,$1 ,$fe,$0 ,$0 ,$0 ,$0 ,$0),
($12,$24,$8 ,$8 ,$4 ,$2 ,$41,$81,$7e,$0 ,$0 ,$0 ,$0 ,$0),
($0 ,$13,$24,$8 ,$8 ,$6 ,$1 ,$1 ,$fe,$0 ,$0 ,$0 ,$0 ,$0),
($0 ,$1 ,$3 ,$1 ,$1 ,$1 ,$1 ,$41,$81,$81,$81,$42,$3c,$0),
($0 ,$21,$21,$11,$11,$9 ,$9 ,$7 ,$1e,$0 ,$0 ,$0 ,$0 ,$0),
($0 ,$1 ,$3 ,$1 ,$1 ,$1 ,$1 ,$1 ,$fe,$0 ,$0 ,$0 ,$0 ,$0),
($0 ,$0 ,$0 ,$0 ,$0 ,$0 ,$6 ,$9 ,$3e,$40,$40,$40,$40,$40),
($0 ,$0 ,$0 ,$0 ,$0 ,$6 ,$9 ,$9 ,$f6,$0 ,$0 ,$0 ,$0 ,$0),
($0 ,$0 ,$0 ,$0 ,$0 ,$8 ,$0 ,$41,$81,$81,$81,$42,$3c,$0),
($0 ,$0 ,$0 ,$8 ,$0 ,$0 ,$1 ,$1 ,$fe,$0 ,$0 ,$0 ,$0 ,$0),

```
($0 ,$0 ,$0 ,$0 ,$0 ,$6 ,$9 ,$9 ,$7 ,$1 ,$2 ,$44,$38,$0 ),
($0 ,$0 ,$0 ,$0 ,$c ,$12,$11,$11,$e ,$0 ,$0 ,$0 ,$0 ,$0 ),
($0 ,$0 ,$0 ,$0 ,$0 ,$1c,$24,$28,$f3,$24,$14,$c ,$0 ,$0 ),
($0 ,$0 ,$0 ,$8 ,$1c,$26,$25,$19,$ee,$0 ,$0 ,$0 ,$0 ,$0 ),
($0 ,$0 ,$0 ,$0 ,$0 ,$0 ,$0 ,$0 ,$7 ,$48,$8c,$82,$82,$7c),
($0 ,$0 ,$0 ,$0 ,$0 ,$3 ,$44,$88,$8e,$81,$81,$7e,$0 ,$0 ),
($0 ,$0 ,$0 ,$0 ,$0 ,$0 ,$1 ,$1 ,$fe,$0 ,$0 ,$28,$0 ,$0 ),
($0 ,$0 ,$0 ,$0 ,$0 ,$0 ,$0 ,$0 ,$0 ,$0 ,$0 ,$0 ,$0 ,$0 ),

//***** latin number patterns
($0 ,$0 ,$0 ,$7c,$c6,$ce,$de,$f6,$e6,$c6,$c6,$7c,$0 ,$0 ),
($0 ,$0 ,$0 ,$18,$38,$78,$18,$18,$18,$18,$18,$7e,$0 ,$0 ),
($0 ,$0 ,$0 ,$7c,$c6,$6 ,$c ,$18,$30,$60,$c6,$fe,$0 ,$0 ),
($0 ,$0 ,$0 ,$7c,$c6,$6 ,$6 ,$3c,$6 ,$6 ,$c6,$7c,$0 ,$0 ),
($0 ,$0 ,$0 ,$c ,$1c,$3c,$6c,$cc,$fe,$c ,$c ,$1e,$0 ,$0 ),
($0 ,$0 ,$0 ,$fe,$c0,$c0,$c0,$fc,$6 ,$6 ,$c6,$7c,$0 ,$0 ),
($0 ,$0 ,$0 ,$38,$60,$c0,$c0,$fc,$c6,$c6,$c6,$7c,$0 ,$0 ),
($0 ,$0 ,$0 ,$fe,$c6,$6 ,$c ,$18,$30,$30,$30,$30,$0 ,$0 ),
($0 ,$0 ,$0 ,$7c,$c6,$c6,$c6,$7c,$c6,$c6,$c6,$7c,$0 ,$0 ),
($0 ,$0 ,$0 ,$7c,$c6,$c6,$c6,$7e,$6 ,$6 ,$c ,$78,$0 ,$0 )
);

   English_rasters :array[0..94,0..12] of GLubyte =(
($00, $00, $00, $00, $00, $00, $00, $00, $00, $00, $00, $00, $00),
($00, $00, $18, $18, $00, $00, $18, $18, $18, $18, $18, $18, $18),
($00, $00, $00, $00, $00, $00, $00, $00, $00, $36, $36, $36, $36),
($00, $00, $00, $66, $66, $ff, $66, $66, $ff, $66, $66, $00, $00),
($00, $00, $18, $7e, $ff, $1b, $1f, $7e, $f8, $d8, $ff, $7e, $18),
($00, $00, $0e, $1b, $db, $6e, $30, $18, $0c, $76, $db, $d8, $70),
($00, $00, $7f, $c6, $cf, $d8, $70, $70, $d8, $cc, $cc, $6c, $38),
($00, $00, $00, $00, $00, $00, $00, $00, $00, $18, $1c, $0c, $0e),
($00, $00, $0c, $18, $30, $30, $30, $30, $30, $30, $30, $18, $0c),
($00, $00, $30, $18, $0c, $0c, $0c, $0c, $0c, $0c, $0c, $18, $30),
($00, $00, $00, $00, $99, $5a, $3c, $ff, $3c, $5a, $99, $00, $00),
($00, $00, $00, $18, $18, $18, $ff, $ff, $18, $18, $18, $00, $00),
($00, $00, $30, $18, $1c, $1c, $00, $00, $00, $00, $00, $00, $00),
($00, $00, $00, $00, $00, $00, $ff, $ff, $00, $00, $00, $00, $00),
($00, $00, $00, $38, $38, $00, $00, $00, $00, $00, $00, $00, $00),
($00, $60, $60, $30, $30, $18, $18, $0c, $0c, $06, $06, $03, $03),
($00, $00, $3c, $66, $c3, $e3, $f3, $db, $cf, $c7, $c3, $66, $3c),
($00, $00, $7e, $18, $18, $18, $18, $18, $18, $18, $78, $38, $18),
($00, $00, $ff, $c0, $c0, $60, $30, $18, $0c, $06, $03, $e7, $7e),
($00, $00, $7e, $e7, $03, $03, $07, $7e, $07, $03, $03, $e7, $7e),
($00, $00, $0c, $0c, $0c, $0c, $0c, $ff, $cc, $6c, $3c, $1c, $0c),
($00, $00, $7e, $e7, $03, $03, $07, $fe, $c0, $c0, $c0, $c0, $ff),
($00, $00, $7e, $e7, $c3, $c3, $c7, $fe, $c0, $c0, $c0, $e7, $7e),
($00, $00, $30, $30, $30, $30, $18, $0c, $06, $03, $03, $03, $ff),
($00, $00, $7e, $e7, $c3, $c3, $e7, $7e, $e7, $c3, $c3, $e7, $7e),
($00, $00, $7e, $e7, $03, $03, $03, $7f, $e7, $c3, $c3, $e7, $7e),
($00, $00, $00, $38, $38, $00, $00, $38, $38, $00, $00, $00, $00),
($00, $00, $30, $18, $1c, $1c, $00, $00, $1c, $1c, $00, $00, $00),
($00, $00, $06, $0c, $18, $30, $60, $c0, $60, $30, $18, $0c, $06),
($00, $00, $00, $00, $ff, $ff, $00, $ff, $ff, $00, $00, $00, $00),
($00, $00, $60, $30, $18, $0c, $06, $03, $06, $0c, $18, $30, $60),
($00, $00, $18, $00, $00, $18, $18, $0c, $06, $03, $c3, $c3, $7e),
($00, $00, $3f, $60, $cf, $db, $d3, $dd, $c3, $7e, $00, $00, $00),
($00, $00, $c3, $c3, $c3, $c3, $ff, $c3, $c3, $c3, $66, $3c, $18),
($00, $00, $fe, $c7, $c3, $c3, $c7, $fe, $c7, $c3, $c3, $c7, $fe),
```

($00, $00, $7e, $e7, $c0, $c0, $c0, $c0, $c0, $c0, $c0, $e7, $7e),
($00, $00, $fc, $ce, $c7, $c3, $c3, $c3, $c3, $c3, $c7, $ce, $fc),
($00, $00, $ff, $c0, $c0, $c0, $c0, $fc, $c0, $c0, $c0, $c0, $ff),
($00, $00, $c0, $c0, $c0, $c0, $c0, $c0, $fc, $c0, $c0, $c0, $ff),
($00, $00, $7e, $e7, $c3, $c3, $cf, $c0, $c0, $c0, $c0, $e7, $7e),
($00, $00, $c3, $c3, $c3, $c3, $c3, $ff, $c3, $c3, $c3, $c3, $c3),
($00, $00, $7e, $18, $18, $18, $18, $18, $18, $18, $18, $18, $7e),
($00, $00, $7c, $ee, $c6, $06, $06, $06, $06, $06, $06, $06, $06),
($00, $00, $c3, $c6, $cc, $d8, $f0, $e0, $f0, $d8, $cc, $c6, $c3),
($00, $00, $ff, $c0, $c0, $c0, $c0, $c0, $c0, $c0, $c0, $c0, $c0),
($00, $00, $c3, $c3, $c3, $c3, $c3, $c3, $db, $ff, $ff, $e7, $c3),
($00, $00, $c7, $c7, $cf, $cf, $df, $db, $fb, $f3, $f3, $e3, $e3),
($00, $00, $7e, $e7, $c3, $c3, $c3, $c3, $c3, $c3, $c3, $e7, $7e),
($00, $00, $c0, $c0, $c0, $c0, $c0, $fe, $c7, $c3, $c3, $c7, $fe),
($00, $00, $3f, $6e, $df, $db, $c3, $c3, $c3, $c3, $c3, $66, $3c),
($00, $00, $c3, $c6, $cc, $d8, $f0, $fe, $c7, $c3, $c3, $c7, $fe),
($00, $00, $7e, $e7, $03, $03, $07, $7e, $e0, $c0, $c0, $e7, $7e),
($00, $00, $18, $18, $18, $18, $18, $18, $18, $18, $18, $18, $ff),
($00, $00, $7e, $e7, $c3, $c3, $c3, $c3, $c3, $c3, $c3, $c3, $c3),
($00, $00, $18, $3c, $3c, $66, $66, $c3, $c3, $c3, $c3, $c3, $c3),
($00, $00, $c3, $e7, $ff, $ff, $db, $db, $c3, $c3, $c3, $c3, $c3),
($00, $00, $c3, $66, $66, $3c, $3c, $18, $3c, $3c, $66, $66, $c3),
($00, $00, $18, $18, $18, $18, $18, $18, $3c, $3c, $66, $66, $c3),
($00, $00, $ff, $c0, $c0, $60, $30, $7e, $0c, $06, $03, $03, $ff),
($00, $00, $3c, $30, $30, $30, $30, $30, $30, $30, $30, $30, $3c),
($00, $03, $03, $06, $06, $0c, $0c, $18, $18, $30, $30, $60, $60),
($00, $00, $3c, $0c, $0c, $0c, $0c, $0c, $0c, $0c, $0c, $0c, $3c),
($00, $00, $00, $00, $00, $00, $00, $00, $00, $c3, $66, $3c, $18),
($ff, $ff, $00, $00, $00, $00, $00, $00, $00, $00, $00, $00, $00),
($00, $00, $00, $00, $00, $00, $00, $00, $00, $18, $38, $30, $70),
($00, $00, $7f, $c3, $c3, $7f, $03, $c3, $7e, $00, $00, $00, $00),
($00, $00, $fe, $c3, $c3, $c3, $c3, $fe, $c0, $c0, $c0, $c0, $c0),
($00, $00, $7e, $c3, $c0, $c0, $c0, $c3, $7e, $00, $00, $00, $00),
($00, $00, $7f, $c3, $c3, $c3, $c3, $7f, $03, $03, $03, $03, $03),
($00, $00, $7f, $c0, $c0, $fe, $c3, $c3, $7e, $00, $00, $00, $00),
($00, $00, $30, $30, $30, $30, $30, $fc, $30, $30, $30, $33, $1e),
($7e, $c3, $03, $03, $7f, $c3, $c3, $c3, $7e, $00, $00, $00, $00),
($00, $00, $c3, $c3, $c3, $c3, $c3, $c3, $fe, $c0, $c0, $c0, $c0),
($00, $00, $18, $18, $18, $18, $18, $18, $18, $00, $00, $18, $00),
($38, $6c, $0c, $0c, $0c, $0c, $0c, $0c, $0c, $00, $00, $0c, $00),
($00, $00, $c6, $cc, $f8, $f0, $d8, $cc, $c6, $c0, $c0, $c0, $c0),
($00, $00, $7e, $18, $18, $18, $18, $18, $18, $18, $18, $18, $78),
($00, $00, $db, $db, $db, $db, $db, $db, $fe, $00, $00, $00, $00),
($00, $00, $c6, $c6, $c6, $c6, $c6, $c6, $fc, $00, $00, $00, $00),
($00, $00, $7c, $c6, $c6, $c6, $c6, $c6, $7c, $00, $00, $00, $00),
($c0, $c0, $c0, $fe, $c3, $c3, $c3, $c3, $fe, $00, $00, $00, $00),
($03, $03, $03, $7f, $c3, $c3, $c3, $c3, $7f, $00, $00, $00, $00),
($00, $00, $c0, $c0, $c0, $c0, $c0, $e0, $fe, $00, $00, $00, $00),
($00, $00, $fe, $03, $03, $7e, $c0, $c0, $7f, $00, $00, $00, $00),
($00, $00, $1c, $36, $30, $30, $30, $30, $fc, $30, $30, $30, $00),
($00, $00, $7e, $c6, $c6, $c6, $c6, $c6, $c6, $00, $00, $00, $00),
($00, $00, $18, $3c, $3c, $66, $66, $c3, $c3, $00, $00, $00, $00),
($00, $00, $c3, $e7, $ff, $db, $c3, $c3, $c3, $00, $00, $00, $00),
($00, $00, $c3, $66, $3c, $18, $3c, $66, $c3, $00, $00, $00, $00),
($c0, $60, $60, $30, $18, $3c, $66, $66, $c3, $00, $00, $00, $00),
($00, $00, $ff, $60, $30, $18, $0c, $06, $ff, $00, $00, $00, $00),
($00, $00, $0f, $18, $18, $18, $38, $f0, $38, $18, $18, $18, $0f),
($18, $18, $18, $18, $18, $18, $18, $18, $18, $18, $18, $18, $18),

```
($00, $00, $f0, $18, $18, $18, $1c, $0f, $1c, $18, $18, $18, $f0),
($00, $00, $00, $00, $00, $00, $06, $8f, $f1, $60, $00, $00, $00)
);

 OpenGL_bits1 : array[0..38] of GLubyte =(
   $00, $03, $00,
   $7f, $fb, $ff,
   $7f, $fb, $ff,
   $00, $03, $00,
   $3e, $8f, $b7,
   $63, $db, $b0,
   $63, $db, $b7,
   $63, $db, $b6,
   $63, $8f, $f3,
   $63, $00, $00,
   $63, $00, $00,
   $63, $00, $00,
   $3e, $00, $00);

 OpenGL_bits2 : array[0..38] of GLubyte =(
   $00, $00, $00,
   $ff, $ff, $01,
   $ff, $ff, $01,
   $00, $00, $00,
   $f9, $fc, $01,
   $8d, $0d, $00,
   $8d, $0d, $00,
   $8d, $0d, $00,
   $cc, $0d, $00,
   $0c, $4c, $0a,
   $0c, $4c, $0e,
   $8c, $ed, $0e,
   $f8, $0c, $00);

 logo_bits : array[0..47] of GLubyte =(
   $00, $66, $66,
   $ff, $66, $66,
   $00, $00, $00,
   $ff, $3c, $3c,
   $00, $42, $40,
   $ff, $42, $40,
   $00, $41, $40,
   $ff, $21, $20,
   $00, $2f, $20,
   $ff, $20, $20,
   $00, $10, $90,
   $ff, $10, $90,
   $00, $0f, $10,
   $ff, $00, $00,
   $00, $66, $66,
   $ff, $66, $66 );

  fontOffset : GLuint ;
  fOffset : GLuint ;

procedure makeRasterFont();
var
    i : GLuint;
```

```
begin
   glPixelStorei(GL_UNPACK_ALIGNMENT, 1);
   fontOffset := glGenLists (128);
   for i := 32 to 127-1 do
   begin
     glNewList(i+fontOffset, GL_COMPILE);
     glBitmap(8, 13, 0.0, 2.0, 10.0, 0.0, @English_rasters[i-32]);
     glEndList();
   end;
end;


procedure makeFarsiRasterFont();
var
    i : GLuint;
begin
   glPixelStorei(GL_UNPACK_ALIGNMENT, 1);
   fOffset := glGenLists (128);
   for i := 32 to 127-1 do
   begin
     glNewList(i+fontOffset, GL_COMPILE);
     glBitmap(8, 13, 0.0, 2.0, 10.0, 0.0, @farsi_raster_font[i-32]);
     glEndList();
   end;
end;

procedure InitializeOpenGL();
begin
   glPixelStorei (GL_UNPACK_ALIGNMENT, 1);
   glClearColor (0.0, 0.0, 0.0, 0.0);
   glShadeModel (GL_FLAT);
   makeFarsiRasterFont();
   makeRasterFont();
end;

procedure printFarsiString(s : Pchar );
begin
   glPushAttrib (GL_LIST_BIT);
    glListBase(fOffset);
    glCallLists(length(s), GL_UNSIGNED_BYTE,  s);
   glPopAttrib ();
end;

procedure printString(s : Pchar );
begin
   glPushAttrib (GL_LIST_BIT);
    glListBase(fontOffset);
    glCallLists(length(s), GL_UNSIGNED_BYTE,  s);
   glPopAttrib ();
end;

procedure Reshape( width, height : GLint);
begin
   glViewport(0, 0, width, height);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   gluOrtho2D(-175, 175, -175, 175);
   glMatrixMode(GL_MODELVIEW);
```

```
end;

procedure DrawFontDemo();
begin
   glColor3f(0,1,0);
   glRasterPos2i(-170, -150);
   printString('The quick brown fox jumps over a lazy dog.');
   glColor3f(1,1,0);
   glRasterPos2i(-170, -130);
   printFarsiString('abcdefghijk');
end ;

procedure Draw();
var
    mapI, mapIA, mapIR : array[0..1] of single;
begin
   glClear(GL_COLOR_BUFFER_BIT);

   mapI[0] := 0.0;
   mapI[1] := 1.0;
   mapIR[0] := 0.0;
   mapIR[1] := 0.0;
   mapIA[0] := 1.0;
   mapIA[1] := 1.0;

   glPixelMapfv(GL_PIXEL_MAP_I_TO_R, 2, @mapIR);
   glPixelMapfv(GL_PIXEL_MAP_I_TO_G, 2, @mapI);
   glPixelMapfv(GL_PIXEL_MAP_I_TO_B, 2, @mapI);
   glPixelMapfv(GL_PIXEL_MAP_I_TO_A, 2, @mapIA);
   glPixelTransferi(GL_MAP_COLOR, 1);

   glColor3f(1,1,1);
   glRasterPos3fv(@boxA);
   glPixelStorei(GL_UNPACK_ROW_LENGTH, 24);
   glPixelStorei(GL_UNPACK_SKIP_PIXELS, 8);
   glPixelStorei(GL_UNPACK_SKIP_ROWS, 2);
   glPixelStorei(GL_UNPACK_LSB_FIRST, 0);
   glPixelStorei(GL_UNPACK_ALIGNMENT, 1);
   glBitmap(16, 12, 8.0, 0.0, 0.0, 0.0, @logo_bits);

   glPixelStorei(GL_UNPACK_ROW_LENGTH, 0);
   glPixelStorei(GL_UNPACK_SKIP_PIXELS, 0);
   glPixelStorei(GL_UNPACK_SKIP_ROWS, 0);
   glPixelStorei(GL_UNPACK_LSB_FIRST, 1);
   glPixelStorei(GL_UNPACK_ALIGNMENT, 1);

   glColor3f(1,1,1);
   glRasterPos3fv(@boxB);
   glBitmap(OPENGL_WIDTH, OPENGL_HEIGHT, OPENGL_WIDTH,
               0.0, OPENGL_WIDTH, 0.0, @OpenGL_bits1);
   glBitmap(OPENGL_WIDTH, OPENGL_HEIGHT, OPENGL_WIDTH,
          0.0, OPENGL_WIDTH, 0.0,  @OpenGL_bits2);

   glColor3f(0,1,1);
   glRasterPos3fv(@boxC);
   glBitmap(OPENGL_WIDTH, OPENGL_HEIGHT, OPENGL_WIDTH,
         0.0, OPENGL_WIDTH, 0.0,  @OpenGL_bits1);
   glBitmap(OPENGL_WIDTH, OPENGL_HEIGHT, OPENGL_WIDTH,
```

```
          0.0, OPENGL_WIDTH, 0.0,  @OpenGL_bits2);

   glColor3f(1,0,1);
   glRasterPos3fv(@boxD);
   glBitmap(OPENGL_WIDTH, OPENGL_HEIGHT, OPENGL_WIDTH,
      0.0, OPENGL_WIDTH, 0.0,  @OpenGL_bits1);
   glBitmap(OPENGL_WIDTH, OPENGL_HEIGHT, OPENGL_WIDTH,
      0.0, OPENGL_WIDTH, 0.0, @OpenGL_bits2);

   glColor3f(1,0,0);
   glRasterPos3fv(@boxE);
   glBitmap(OPENGL_WIDTH, OPENGL_HEIGHT, OPENGL_WIDTH,
        0.0, OPENGL_WIDTH, 0.0,  @OpenGL_bits1);
   glBitmap(OPENGL_WIDTH, OPENGL_HEIGHT, OPENGL_WIDTH, 0.0,
          OPENGL_WIDTH, 0.0,  @OpenGL_bits2);
end;

procedure RenderScene();
begin
     glClear(GL_COLOR_BUFFER_BIT);
     Draw;
     DrawFontDemo;
     SwapBuffers(f_Hdc);
end;

procedure TForm1.FormResize(Sender: TObject);
begin
  wglMakeCurrent(f_Hdc,hrc); //activate the RC
  Reshape( width, height );
  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;

end.
```