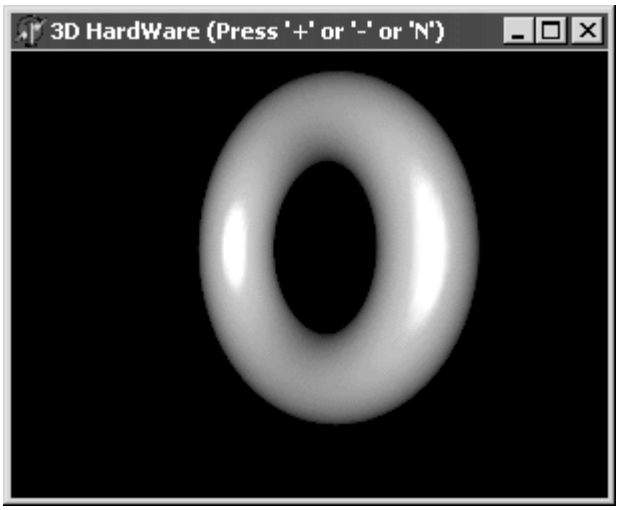
## فصل بیست و هفتم

# استفاده از امکانات ویژه کارت های گرافیکی سه بعدی




**مقدمه :**

طراحان OpenGL نیاز به توسعه آنرا در آینده پیش بینی کرده اند ، بنابراین مفهومی را به نام Extensions ، توسعه داده اند . OpenGL از Extensions پشتیبانی می کند تا سازندگان کارتهای گرافیکی سه بعدی ، توانایی های جدید محصولات خود را به سادگی و بدون منتظر تصمیم گیری گروه OpenGL ماندن ، به دنیا عرضه کنند (برخلاف Direct-X) .

**طریقه نامگذاری Extensions در OpenGL :**

GL_type_name :

در عبارت فوق بجای Type یا Ext قرار می گیرد و یا نام شرکت تولید کننده امکانات ویژه سخت افزاری و name رشته ای است متشکل از کاراکترهایی با حروف کوچک ، برای مثال :

GL_EXT_polygon_offset
GL_SGI_color_table
GL_SGIS_detail_texture
GL_MESA_window_pos

اگر یک Extension ، تابعی جدید را تعریف نماید ، بصورت زیر بکار برده می شود :

GL_EXT_polygon_offset
void glPolygonOffsetEXT( GLfloat factor, GLfloat bias ); (به فرمت زبان C)

**توضیحات بعضی از Extensions بکار گرفته شده در OpenGL :**

GL_EXT_abgr : فرمت نقطه ای GL_ABGR_EXT را به توابع glDrawPixels ، glReadPixels و glTexImage اضافه کرده است .

GL_EXT_blend_color : امکان عملیات اختلاط را با رنگ های ثابت ، فراهم آورده است .

GL_EXT_blend_logic_op : توانایی glLogicOp را به اختلاط RGB ، افزوده است .

GL_EXT_blend_minmax : عملگر های Min/Max را به اختلاط RGB اضافه می کند .

GL_EXT_blend_equation : امکان معادلات اختلاط کاهنده را فراهم می کند .

GL_EXT_convolution : پیچیدگی ۱و۲ بعدی تصاویر را میسر می سازد .

GL_EXT_copy_texture : امکان بارگذاری مستقیم تصاویر نگاشت را از بافر فراهم می آورد.

GL_EXT_packed_pixels : فرمت نقطه ای فشرده را به توابع glDrawPixels ، glReadPixels و glTexImage اضافه می کند .

GL_EXT_polygon_offset : تابع glPolygonOffsetEXT را به امکانات OpenGL می افزاید .

GL_EXT_subtexture : امکان تعویض زیرنواحی تصاویر نگاشت را فراهم می نماید .

GL_EXT_texture : نوع های داده ای فرمت فشرده بافتی را اضافه می کند .

GL_EXT_texture3D : پشتیبانی سه بعدی تصویر بافتی را که برای رندرگیری حجمی مفید است ، میسر می سازد .

GL_EXT_texture_object : کارآیی را هنگامیکه نگاشت های متعدد بافتی نیاز است ، با نامگذاری آنها ، بهبود می بخشد .

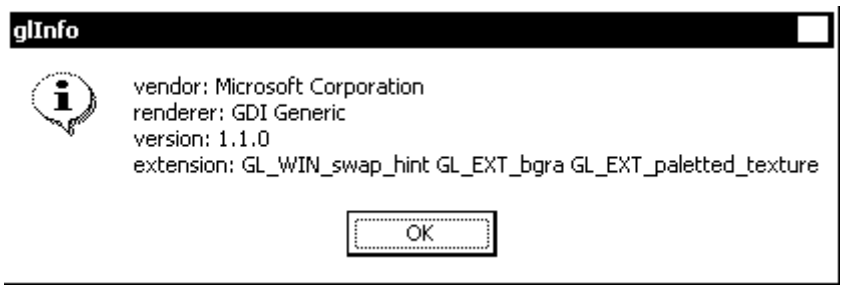
GL_EXT_vertex_array : اشکال هندسی را با آرایه ای از داده های مختصات ، مشخص می کند و راه بهتری را بجای استفاده از glVertex, glColor, glNormal, or glTexCoord ارائه می دهد .

و موارد دیگری که می توانید برای آگاهی از آنها به MSDN و یا سایت های اینترنتی سازندگان کارت های گرافیکی سه بعدی ، مراجعه نمایید .

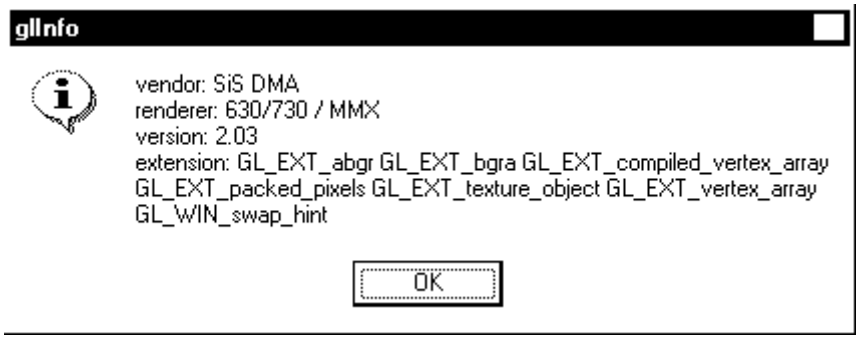
**مقایسه توانایی های چند کارت گرافیکی متداول در ایران :**

در تصاویر زیر توانایی های کارت های گرافیکی مختلف با استفاده از تابع glInfo که لیست آن در این فصل نیز تکرار می گردد ، ارائه گردیده است .

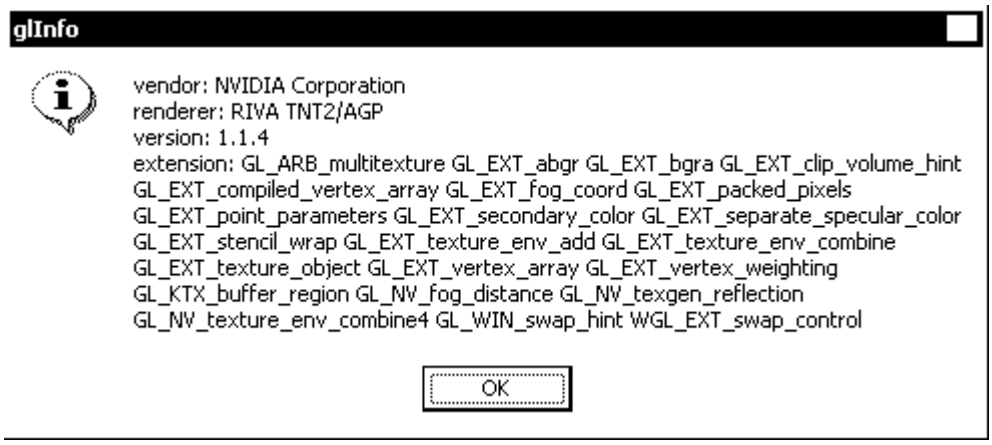
*کارت گرافیکی **Trident 9750** :*



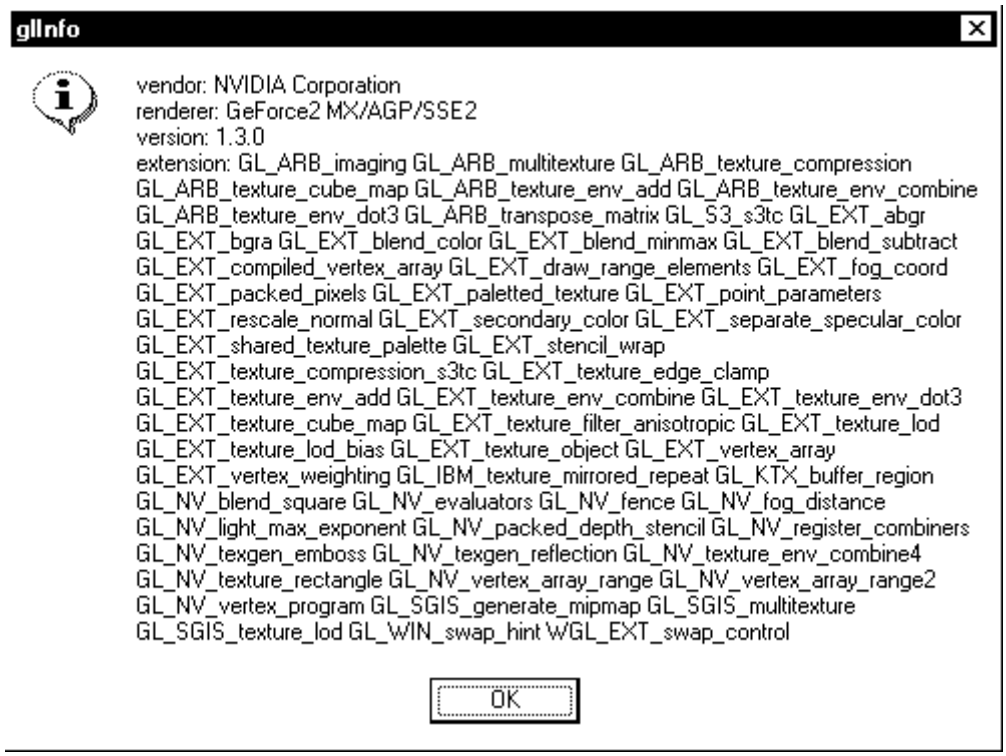

*کارت گرافیکی **SiS DMA**:*



*کارت گرافیکی **TNT2/AGP** :*




*کارت گرافیکی **GeForce2** MX/AGP/SSE2:*




احتمالا پس از مشاهده تصاویر فوق ، دلیل ارزان بودن کارت های گرافیکی OnBoard را متوجه شده اید !

## لیست کاملی از انواع Extensions شناخته شده تا امروز :

GL_3DFX_multisample
GL_3DFX_texture_compression_FXT1
GL_ARB_imaging
GL_ARB_multisample
GL_ARB_multitexture
GL_ARB_point_parameters
GL_ARB_texture_border_clamp
GL_ARB_texture_compression
GL_ARB_texture_cube_map
GL_ARB_texture_env_add
GL_ARB_texture_env_combine
GL_ARB_texture_env_crossbar
GL_ARB_texture_env_dot3
GL_ARB_texture_mirrored_repeat
GL_ARB_transpose_matrix
GL_ARB_vertex_blend
GL_ARB_window_pos
GL_ATIX_envmap_bumpmap
GL_ATIX_pn_triangles
GL_ATIX_texture_env_combine3
GL_ATIX_texture_env_dot3
GL_ATIX_texture_env_route
GL_ATIX_texture_env_subtract
GL_ATIX_vertex_shader_output_point_size
GL_ATI_element_array
GL_ATI_envmap_bumpmap
GL_ATI_fragment_shader
GL_ATI_lock_texture
GL_ATI_map_object_buffer
GL_ATI_pn_triangles
GL_ATI_texture_mirror_once
GL_ATI_vertex_array_object
GL_ATI_vertex_blend
GL_ATI_vertex_shader
GL_ATI_vertex_streams
GL_Autodesk_facet_normal
GL_Autodesk_valid_back_buffer_hint
GL_DIMD_YUV
GL_EXT_abgr
GL_EXT_bgra
GL_EXT_blend_color
GL_EXT_blend_func_separate
GL_EXT_blend_logic_op
GL_EXT_blend_minmax
GL_EXT_blend_subtract
GL_EXT_clip_volume_hint
GL_EXT_compiled_vertex_array
GL_EXT_convolution
GL_EXT_cull_vertex
GL_EXT_draw_range_elements
GL_EXT_fog_coord
GL_EXT_histogram

GL_EXT_multi_draw_arrays
GL_EXT_packed_pixels
GL_EXT_packed_pixels_12
GL_EXT_paletted_texture
GL_EXT_point_parameters
GL_EXT_polygon_offset
GL_EXT_rescale_normal
GL_EXT_secondary_color
GL_EXT_separate_specular_color
GL_EXT_shared_texture_palette
GL_EXT_stencil_wrap
GL_EXT_subtexture
GL_EXT_texgen_reflection
GL_EXT_texture3D
GL_EXT_texture_compression_s3tc
GL_EXT_texture_cube_map
GL_EXT_texture_edge_clamp
GL_EXT_texture_env_add
GL_EXT_texture_env_combine
GL_EXT_texture_env_dot3
GL_EXT_texture_filter_anisotropic
GL_EXT_texture_lod
GL_EXT_texture_lod_bias
GL_EXT_texture_object
GL_EXT_vertex_array
GL_EXT_vertex_shader
GL_EXT_vertex_weighting
GL_HP_occlusion_test
GL_IBM_cull_vertex
GL_IBM_multimode_draw_arrays
GL_IBM_multi_draw_arrays
GL_IBM_rasterpos_clip
GL_IBM_rescale_normal
GL_IBM_texture_mirrored_repeat
GL_IBM_vertex_array_lists
GL_INGR_blend_func_separate
GL_KTX_buffer_region
GL_MESA_resize_buffers
GL_MESA_window_pos
GL_NV_blend_square
GL_NV_copy_depth_to_color
GL_NV_evaluators
GL_NV_fence
GL_NV_fog_distance
GL_NV_light_max_exponent
GL_NV_multisample_filter_hint
GL_NV_occlusion_query
GL_NV_packed_depth_stencil
GL_NV_register_combiners
GL_NV_register_combiners2
GL_NV_texgen_emboss
GL_NV_texgen_reflection
GL_NV_texture_compression_vtc
GL_NV_texture_env_combine4
GL_NV_texture_rectangle
GL_NV_texture_shader
GL_NV_texture_shader2
GL_NV_texture_shader3

GL_NV_vertex_array_range
GL_NV_vertex_array_range2
GL_NV_vertex_program
GL_NV_vertex_program1_1
GL_S3_s3tc
GL_SGIS_generate_mipmap
GL_SGIS_multitexture
GL_SGIS_pixel_texture
GL_SGIS_texture_border_clamp
GL_SGIS_texture_edge_clamp
GL_SGIS_texture_lod
GL_SGIX_depth_texture
GL_SGIX_pixel_texture
GL_SGIX_shadow
GL_SGIX_shadow_ambient
GL_SGI_color_matrix
GL_SGI_color_table
GL_SGI_compiled_vertex_array
GL_SGI_cull_vertex
GL_SGI_index_array_formats
GL_SGI_index_func
GL_SGI_index_material
GL_SGI_index_texture
GL_SGI_texture_edge_clamp
GL_WIN_swap_hint
WGL_3DFX_gamma_control
WGL_EXT_swap_control
WGL_3DFX_multisample
WGL_ARB_buffer_region
WGL_ARB_extensions_string
WGL_ARB_make_current_read
WGL_ARB_multisample
WGL_ARB_pbuffer
WGL_ARB_pixel_format
WGL_ARB_render_texture
WGL_EXT_extensions_string
WGL_EXT_pixel_format
WGL_NV_render_depth_texture
WGL_NV_render_texture_rectangle

با توجه به تعداد زیاد Extensions موجود ، برای دریافت توضیحات کامل مربوط به این Extensions می توانید به سایت های اینترنتی سازندگان مربوطه مراجعه نمایید .

**طریقه بکارگیری توابع** Extensions:

بسیاری از Extensions تنها ثوابت جدیدی را برای توابع موجود در OpenGL تعریف می نمایند و استفاده از آنها بسیار ساده می باشد . اما Extension های دیگر ممکن است توابعی را نیز به

مجموعه توابع استاندارد OpenGL اضافه نمایند . برای استفاده از این توابع در ابتدا باید یک اشاره گر به تابع مربوطه تعریف نمود . برای مثال EXT_point_parameters در بازی Quake 2 بکار رفته است و از آن برای رندر کردن سیستم ذرات (Particles) استفاده می شود. توسط این Extension ، انفجارها بوسیله مجموعه عظیمی از اشیاء نقطه ای OpenGL رندر و بصورت خودکار بر مبنای فاصله ذرات از بیننده ، اندازه آنها تنظیم می گردند . این Extension دو تابع جدید به نامهای glPointParameterfEXT و glPointParameterfvEXT به OpenGL اضافه کرده است . برنامه ای که بخواهد از این توابع استفاده کند باید توسط تابع wglGetProcAddress ویندوز ، آدرس تابع را در کتابخانه یافته و سپس از طریق این آدرس تابع را بکار بگیرد .

در ادامه کد زبان C ، جهت بکارگیری این Extension ارائه می گرد و نحوه تبدیل آن به دلفی در واحد کمکی این فصل (glExt) مرور شده است .

```
#ifdef _WIN32
typedef void (APIENTRY * PFNGLPOINTPARAMETERFEXTPROC)
(GLenum pname, GLfloat param);
typedef void (APIENTRY * PFNGLPOINTPARAMETERFVEXTPROC)
(GLenum pname, const GLfloat *params);
#endif

int hasPointParams = isSupported ("GL_EXT_point_parameters");
#ifdef _WIN32
if (hasPointParams) {
glPointParameterfEXT = (PFNGLPOINTPARAMETERFEXTPROC)
wglGetProcAddress("glPointParameterfEXT");
glPointParameterfvEXT = (PFNGLPOINTPARAMETERFVEXTPROC)
wglGetProcAddress("glPointParameterfvEXT");
}
#endif

if (hasPointParams) {
static GLfloat quadratic[3] = { 0.25, 0.0, 1/60.0 };
glPointParameterfvEXT(GL_DISTANCE_ATTENUATION_EXT, quadratic);
glPointParameterfEXT(GL_POINT_FADE_THRESHOLD_SIZE_EXT, 1.0);
}
```

باید خاطرنشان کرد که قبل از فراخوانی کدهای فوق باید زمینه رندرسازی جاری OpenGL (hRC) وجود داشته باشد .

برای دریافت اطلاعات بیشتر به واحد glExt ، که در انتهای این فصل ارائه گردیده است مراجعه نمایید .

**توجه ! :**

برنامه شما باید برای حالتی که Extensions دلخواه ، فراهم نمی باشد نیز آماده بوده و در این حالت توانایی اجرای کد معادل آهسته تر را داشته باشد و از کار نیفتد !

برای مثال اگر GL_EXT_vertex_array بر روی سیستمی مهیا نبود ، برنامه باید توانایی استفاده از توابع معمول glVertex/glColor/glNormal را به صورت اتوماتیک داشته باشد .

و در بعضی از حالت ها مانند GL_EXT_texture3D ، اگر این توانایی سخت افزاری وجود نداشته باشد ، راهی بجز متوقف کردن برنامه باقی نمی ماند . اما باید کاربر را از دلیل این توقف ناخواسته ، آگاه کرد .

## برنامه فصل :

در ادامه کد مربوط به واحد glExt ، که بکارگیری Extensions را در دلفی ساده می نماید ارائه می گردد .

```
{
Description: Contains procedure types and variables for the
             latest OpenGL extensions. Also includes a
           procedure for checking if an extension is
           supported by your video card
Installation: Put glExt in your Borland\Delphi\Lib directory
Use: Add glExt to the uses clause of your unit.
    Call isSupported(ext_name_string) to find out if an
    extension is supported.
}

unit glExt;

interface

uses OpenGL, Sysutils;

const
  // Extensions
  GL_ARB_multitexture              = 1;
  GL_EXT_abgr                    = 1;
{$IFNDEF GL_EXT_bgra}
  GL_EXT_bgra                    = 1;
{$ENDIF}
  GL_EXT_blend_color               = 1;
  GL_EXT_blend_minmax               = 1;
  GL_EXT_blend_subtract             = 1;
  GL_EXT_clip_volume_hint           = 1;
```

```
GL_EXT_compiled_vertex_array        = 1;
GL_EXT_fog_coord                    = 1;
GL_EXT_light_max_exponent           = 1;
GL_EXT_packed_pixels                = 1;
{$IFNDEF GL_EXT_paletted_texture}
  GL_EXT_paletted_texture           = 1;
{$ENDIF}
GL_EXT_point_parameters             = 1;
GL_EXT_rescale_normal               = 1;
GL_EXT_secondary_color              = 1;
GL_EXT_separate_specular_color      = 1;
GL_EXT_shared_texture_palette       = 1;
GL_EXT_stencil_wrap                 = 1;
GL_ARB_texture_cube_map             = 1;
GL_EXT_texture_env_add              = 1;
GL_EXT_texture_env_combine          = 1;
GL_EXT_texture_filter_anisotropic   = 1;
GL_EXT_texture_lod_bias             = 1;
{$IFNDEF GL_EXT_vertex_array}
  GL_EXT_vertex_array               = 1;
{$ENDIF}
GL_EXT_vertex_weighting             = 1;
GL_NV_blend_square                  = 1;
GL_NV_fog_distance                  = 1;
GL_NV_register_combiners            = 1;
GL_NV_texgen_emboss                 = 1;
GL_ARG_texgen_reflection            = 1;
GL_NV_texture_env_combine4          = 1;
GL_NV_vertex_array_range            = 1;
GL_SGIS_multitexture                = 1;
GL_SGIS_texture_lod                 = 1;
WGL_EXT_swap_control                = 1;

// EXT_abgr
GL_ABGR_EXT                         = $8000;

// EXT_blend_color
GL_CONSTANT_COLOR_EXT               = $8001;
GL_ONE_MINUS_CONSTANT_COLOR_EXT     = $8002;
GL_CONSTANT_ALPHA_EXT               = $8003;
GL_ONE_MINUS_CONSTANT_ALPHA_EXT     = $8004;
GL_BLEND_COLOR_EXT                  = $8005;

// EXT_blend_minmax
GL_FUNC_ADD_EXT                     = $8006;
GL_MIN_EXT                          = $8007;
GL_MAX_EXT                          = $8008;
GL_BLEND_EQUATION_EXT               = $8009;

// EXT_blend_subtract
GL_FUNC_SUBTRACT_EXT                = $800A;
GL_FUNC_REVERSE_SUBTRACT_EXT        = $800B;

// EXT_packed_pixels
GL_UNSIGNED_BYTE_3_3_2_EXT          = $8032;
GL_UNSIGNED_SHORT_4_4_4_4_EXT       = $8033;
GL_UNSIGNED_SHORT_5_5_5_1_EXT       = $8034;
GL_UNSIGNED_INT_8_8_8_8_EXT         = $8035;
```

```
  GL_UNSIGNED_INT_10_10_10_2_EXT      = $8036;

  // OpenGL 1.2
  {$IFNDEF GL_VERSION_1_2}
    GL_RESCALE_NORMAL               = $803A;
    GL_MAX_ELEMENTS_VERTICES        = $80E8;
    GL_MAX_ELEMENTS_INDICES         = $80E9;
    GL_CLAMP_TO_EDGE                = $812F;
    GL_TEXTURE_MIN_LOD              = $813A;
    GL_TEXTURE_MAX_LOD              = $813B;
    GL_TEXTURE_BASE_LEVEL           = $813C;
    GL_TEXTURE_MAX_LEVEL            = $813D;
    GL_SINGLE_COLOR                 = $81F9;
    GL_SEPARATE_SPECULAR_COLOR      = $81FA;
    GL_LIGHT_MODEL_COLOR_CONTROL    = $81F8;
  {$ENDIF}

  // ARB_imaging
  {IFNDEF ARB_imaging}
    GL_CONSTANT_COLOR               = $8001;
    GL_ONE_MINUS_CONSTANT_COLOR     = $8002;
    GL_CONSTANT_ALPHA               = $8003;
    GL_ONE_MINUS_CONSTANT_ALPHA     = $8004;
    GL_BLEND_COLOR                  = $8005;
    GL_FUNC_ADD                     = $8006;
    GL_MIN                          = $8007;
    GL_MAX                          = $8008;
    GL_BLEND_EQUATION               = $8009;
    GL_FUNC_SUBTRACT                = $800A;
    GL_FUNC_REVERSE_SUBTRACT        = $800B;
  {ENDIF}

  // EXT_vertex_array
  {$IFNDEF GL_VERTEX_ARRAY_EXT}
    GL_VERTEX_ARRAY_EXT             = $8074;
    GL_NORMAL_ARRAY_EXT             = $8075;
    GL_COLOR_ARRAY_EXT              = $8076;
    GL_INDEX_ARRAY_EXT              = $8077;
    GL_TEXTURE_COORD_ARRAY_EXT      = $8078;
    GL_EDGE_FLAG_ARRAY_EXT          = $8079;
    GL_VERTEX_ARRAY_SIZE_EXT        = $807A;
    GL_VERTEX_ARRAY_TYPE_EXT        = $807B;
    GL_VERTEX_ARRAY_STRIDE_EXT      = $807C;
    GL_VERTEX_ARRAY_COUNT_EXT       = $807D;
    GL_NORMAL_ARRAY_TYPE_EXT        = $807E;
    GL_NORMAL_ARRAY_STRIDE_EXT      = $807F;
    GL_NORMAL_ARRAY_COUNT_EXT       = $8080;
    GL_COLOR_ARRAY_SIZE_EXT         = $8081;
    GL_COLOR_ARRAY_TYPE_EXT         = $8082;
    GL_COLOR_ARRAY_STRIDE_EXT       = $8083;
    GL_COLOR_ARRAY_COUNT_EXT        = $8084;
    GL_INDEX_ARRAY_TYPE_EXT         = $8085;
    GL_INDEX_ARRAY_STRIDE_EXT       = $8086;
    GL_INDEX_ARRAY_COUNT_EXT        = $8087;
    GL_TEXTURE_COORD_ARRAY_SIZE_EXT   = $8088;
    GL_TEXTURE_COORD_ARRAY_TYPE_EXT   = $8089;
    GL_TEXTURE_COORD_ARRAY_STRIDE_EXT = $808A;
    GL_TEXTURE_COORD_ARRAY_COUNT_EXT  = $808B;
```

```
  GL_EDGE_FLAG_ARRAY_STRIDE_EXT       = $808C;
  GL_EDGE_FLAG_ARRAY_COUNT_EXT        = $808D;
  GL_VERTEX_ARRAY_POINTER_EXT         = $808E;
  GL_NORMAL_ARRAY_POINTER_EXT         = $808F;
  GL_COLOR_ARRAY_POINTER_EXT          = $8090;
  GL_INDEX_ARRAY_POINTER_EXT          = $8091;
  GL_TEXTURE_COORD_ARRAY_POINTER_EXT  = $8092;
  GL_EDGE_FLAG_ARRAY_POINTER_EXT      = $8093;
{$ENDIF}

// EXT_color_table
{$IFNDEF GL_COLOR_TABLE_FORMAT_EXT}
  GL_TABLE_TOO_LARGE_EXT              = $8031;
  GL_COLOR_TABLE_FORMAT_EXT           = $80D8;
  GL_COLOR_TABLE_WIDTH_EXT            = $80D9;
  GL_COLOR_TABLE_RED_SIZE_EXT         = $80DA;
  GL_COLOR_TABLE_GREEN_SIZE_EXT       = $80DB;
  GL_COLOR_TABLE_BLUE_SIZE_EXT        = $80DC;
  GL_COLOR_TABLE_ALPHA_SIZE_EXT       = $80DD;
  GL_COLOR_TABLE_LUMINANCE_SIZE_EXT   = $80DE;
  GL_COLOR_TABLE_INTENSITY_SIZE_EXT   = $80DF;
{$ENDIF}

// EXT_bgra
{$IFNDEF GL_BGR_EXT}
  GL_BGR_EXT                          = $80E0;
  GL_BGRA_EXT                         = $80E1;
{$ENDIF}

// SGIS_texture_lod
GL_TEXTURE_MIN_LOD_SGIS               = $813A;
GL_TEXTURE_MAX_LOD_SGIS               = $813B;
GL_TEXTURE_BASE_LEVEL_SGIS            = $813C;
GL_TEXTURE_MAX_LEVEL_SGIS             = $813D;

// EXT_paletted_texture
{$IFNDEF GL_COLOR_INDEX1_EXT}
  GL_COLOR_INDEX1_EXT                 = $80E2;
  GL_COLOR_INDEX2_EXT                 = $80E3;
  GL_COLOR_INDEX4_EXT                 = $80E4;
  GL_COLOR_INDEX8_EXT                 = $80E5;
  GL_COLOR_INDEX12_EXT                = $80E6;
  GL_COLOR_INDEX16_EXT                = $80E7;
{$ENDIF}

// EXT_clip_volume_hint
GL_CLIP_VOLUME_CLIPPING_HINT_EXT      = $80F0;

// EXT_point_parameters
GL_POINT_SIZE_MIN_EXT                 = $8126;
GL_POINT_SIZE_MAX_EXT                 = $8127;
GL_POINT_FADE_THRESHOLD_SIZE_EXT      = $8128;
GL_DISTANCE_ATTENUATION_EXT           = $8129;

// EXT_compiled_vertex_array
GL_ARRAY_ELEMENT_LOCK_FIRST_EXT       = $81A8;
GL_ARRAY_ELEMENT_LOCK_COUNT_EXT       = $81A9;
```

```
// EXT_shared_texture_palette
GL_SHARED_TEXTURE_PALETTE_EXT        = $81FB;

// SGIS_multitexture
GL_SELECTED_TEXTURE_SGIS            = $835C;
GL_MAX_TEXTURES_SGIS               = $835D;
GL_TEXTURE0_SGIS                  = $835E;
GL_TEXTURE1_SGIS                  = $835F;
GL_TEXTURE2_SGIS                  = $8360;
GL_TEXTURE3_SGIS                  = $8361;

// ARB_multitexture
GL_ACTIVE_TEXTURE_ARB              = $84E0;
GL_CLIENT_ACTIVE_TEXTURE_ARB        = $84E1;
GL_MAX_TEXTURE_UNITS_ARB           = $84E2;
GL_TEXTURE0_ARB                   = $84C0;
GL_TEXTURE1_ARB                   = $84C1;
GL_TEXTURE2_ARB                   = $84C2;
GL_TEXTURE3_ARB                   = $84C3;
GL_TEXTURE4_ARB                   = $84C4;
GL_TEXTURE5_ARB                   = $84C5;
GL_TEXTURE6_ARB                   = $84C6;
GL_TEXTURE7_ARB                   = $84C7;
GL_TEXTURE8_ARB                   = $84C8;
GL_TEXTURE9_ARB                   = $84C9;
GL_TEXTURE10_ARB                   = $84CA;
GL_TEXTURE11_ARB                   = $84CB;
GL_TEXTURE12_ARB                   = $84CC;
GL_TEXTURE13_ARB                   = $84CD;
GL_TEXTURE14_ARB                   = $84CE;
GL_TEXTURE15_ARB                   = $84CF;
GL_TEXTURE16_ARB                   = $84D0;
GL_TEXTURE17_ARB                   = $84D1;
GL_TEXTURE18_ARB                   = $84D2;
GL_TEXTURE19_ARB                   = $84D3;
GL_TEXTURE20_ARB                   = $84D4;
GL_TEXTURE21_ARB                   = $84D5;
GL_TEXTURE22_ARB                   = $84D6;
GL_TEXTURE23_ARB                   = $84D7;
GL_TEXTURE24_ARB                   = $84D8;
GL_TEXTURE25_ARB                   = $84D9;
GL_TEXTURE26_ARB                   = $84DA;
GL_TEXTURE27_ARB                   = $84DB;
GL_TEXTURE28_ARB                   = $84DC;
GL_TEXTURE29_ARB                   = $84DD;
GL_TEXTURE30_ARB                   = $84DE;
GL_TEXTURE31_ARB                   = $84DF;

// EXT_fog_coord
GL_FOG_COORDINATE_SOURCE_EXT         = $8450;
GL_FOG_COORDINATE_EXT               = $8451;
GL_FRAGMENT_DEPTH_EXT               = $8452;
GL_CURRENT_FOG_COORDINATE_EXT        = $8453;
GL_FOG_COORDINATE_ARRAY_TYPE_EXT     = $8454;
GL_FOG_COORDINATE_ARRAY_STRIDE_EXT    = $8455;
GL_FOG_COORDINATE_ARRAY_POINTER_EXT   = $8456;
GL_FOG_COORDINATE_ARRAY_EXT          = $8457;
```

```
// EXT_secondary_color
GL_COLOR_SUM_EXT                  = $8458;
GL_CURRENT_SECONDARY_COLOR_EXT    = $8459;
GL_SECONDARY_COLOR_ARRAY_SIZE_EXT = $845A;
GL_SECONDARY_COLOR_ARRAY_TYPE_EXT = $845B;
GL_SECONDARY_COLOR_ARRAY_STRIDE_EXT = $845C;
GL_SECONDARY_COLOR_ARRAY_POINTER_EXT = $845D;
GL_SECONDARY_COLOR_ARRAY_EXT      = $845E;

// EXT_separate_specular_color
GL_SINGLE_COLOR_EXT               = $81F9;
GL_SEPARATE_SPECULAR_COLOR_EXT    = $81FA;
GL_LIGHT_MODEL_COLOR_CONTROL_EXT  = $81F8;

// EXT_rescale_normal
GL_RESCALE_NORMAL_EXT             = $803A;

// EXT_stencil_wrap
GL_INCR_WRAP_EXT                  = $8507;
GL_DECR_WRAP_EXT                  = $8508;

// EXT_vertex_weighting
GL_MODELVIEW0_MATRIX_EXT          = GL_MODELVIEW_MATRIX;
GL_MODELVIEW1_MATRIX_EXT          = $8506;
GL_MODELVIEW0_STACK_DEPTH_EXT     = GL_MODELVIEW_STACK_DEPTH;
GL_MODELVIEW1_STACK_DEPTH_EXT     = $8502;
GL_VERTEX_WEIGHTING_EXT           = $8509;
GL_MODELVIEW0_EXT                 = GL_MODELVIEW;
GL_MODELVIEW1_EXT                 = $850A;
GL_CURRENT_VERTEX_WEIGHT_EXT      = $850B;
GL_VERTEX_WEIGHT_ARRAY_EXT        = $850C;
GL_VERTEX_WEIGHT_ARRAY_SIZE_EXT   = $850D;
GL_VERTEX_WEIGHT_ARRAY_TYPE_EXT   = $850E;
GL_VERTEX_WEIGHT_ARRAY_STRIDE_EXT = $850F;
GL_VERTEX_WEIGHT_ARRAY_POINTER_EXT = $8510;

// NV_texgen_reflection
GL_NORMAL_MAP_NV                  = $8511;
GL_REFLECTION_MAP_NV              = $8512;

// EXT_texture_cube_map
GL_NORMAL_MAP_ARB                 = $8511;
GL_REFLECTION_MAP_ARB             = $8512;
GL_TEXTURE_CUBE_MAP_ARB           = $8513;
GL_TEXTURE_BINDING_CUBE_MAP_ARB   = $8514;
GL_TEXTURE_CUBE_MAP_POSITIVE_X_ARB = $8515;
GL_TEXTURE_CUBE_MAP_NEGATIVE_X_ARB = $8516;
GL_TEXTURE_CUBE_MAP_POSITIVE_Y_ARB = $8517;
GL_TEXTURE_CUBE_MAP_NEGATIVE_Y_ARB = $8518;
GL_TEXTURE_CUBE_MAP_POSITIVE_Z_ARB = $8519;
GL_TEXTURE_CUBE_MAP_NEGATIVE_Z_ARB = $851A;
GL_PROXY_TEXTURE_CUBE_MAP_ARB     = $851B;
GL_MAX_CUBE_MAP_TEXTURE_SIZE_ARB  = $851C;

// NV_vertex_array_range
GL_VERTEX_ARRAY_RANGE_NV          = $851D;
GL_VERTEX_ARRAY_RANGE_LENGTH_NV   = $851E;
GL_VERTEX_ARRAY_RANGE_VALID_NV    = $851F;
```

```
GL_MAX_VERTEX_ARRAY_RANGE_ELEMENT_NV  = $8520;
GL_VERTEX_ARRAY_RANGE_POINTER_NV      = $8521;

// NV_register_combiners
GL_REGISTER_COMBINERS_NV          = $8522;
GL_COMBINER0_NV                   = $8550;
GL_COMBINER1_NV                   = $8551;
GL_COMBINER2_NV                   = $8552;
GL_COMBINER3_NV                   = $8553;
GL_COMBINER4_NV                   = $8554;
GL_COMBINER5_NV                   = $8555;
GL_COMBINER6_NV                   = $8556;
GL_COMBINER7_NV                   = $8557;
GL_VARIABLE_A_NV                  = $8523;
GL_VARIABLE_B_NV                  = $8524;
GL_VARIABLE_C_NV                  = $8525;
GL_VARIABLE_D_NV                  = $8526;
GL_VARIABLE_E_NV                  = $8527;
GL_VARIABLE_F_NV                  = $8528;
GL_VARIABLE_G_NV                  = $8529;
// GL_ZERO
GL_CONSTANT_COLOR0_NV             = $852A;
GL_CONSTANT_COLOR1_NV             = $852B;
// GL_FOG
GL_PRIMARY_COLOR_NV               = $852C;
GL_SECONDARY_COLOR_NV             = $852D;
GL_SPARE0_NV                      = $852E;
GL_SPARE1_NV                      = $852F;
// GL_TEXTURE0_ARB
// GL_TEXTURE1_ARB
GL_UNSIGNED_IDENTITY_NV           = $8536;
GL_UNSIGNED_INVERT_NV             = $8537;
GL_EXPAND_NORMAL_NV               = $8538;
GL_EXPAND_NEGATE_NV               = $8539;
GL_HALF_BIAS_NORMAL_NV            = $853A;
GL_HALF_BIAS_NEGATE_NV            = $853B;
GL_SIGNED_IDENTITY_NV             = $853C;
GL_SIGNED_NEGATE_NV               = $853D;
GL_E_TIMES_F_NV                   = $8531;
GL_SPARE0_PLUS_SECONDARY_COLOR_NV = $8532;
// GL_NONE
GL_SCALE_BY_TWO_NV                = $853E;
GL_SCALE_BY_FOUR_NV               = $853F;
GL_SCALE_BY_ONE_HALF_NV           = $8540;
GL_BIAS_BY_NEGATIVE_ONE_HALF_NV   = $8541;
GL_DISCARD_NV                     = $8530;
GL_COMBINER_INPUT_NV              = $8542;
GL_COMBINER_MAPPING_NV            = $8543;
GL_COMBINER_COMPONENT_USAGE_NV    = $8544;
GL_COMBINER_AB_DOT_PRODUCT_NV     = $8545;
GL_COMBINER_CD_DOT_PRODUCT_NV     = $8546;
GL_COMBINER_MUX_SUM_NV            = $8547;
GL_COMBINER_SCALE_NV              = $8548;
GL_COMBINER_BIAS_NV               = $8549;
GL_COMBINER_AB_OUTPUT_NV          = $854a;
GL_COMBINER_CD_OUTPUT_NV          = $854b;
GL_COMBINER_SUM_OUTPUT_NV         = $854c;
GL_MAX_GENERAL_COMBINERS_NV       = $854d;
```

```
  GL_NUM_GENERAL_COMBINERS_NV          = $854e;
  GL_COLOR_SUM_CLAMP_NV              = $854f;
  // NV_fog_distance
  GL_FOG_DISTANCE_MODE_NV             = $855a;
  GL_EYE_RADIAL_NV                 = $855b;
  // GL_EYE_PLANE
  GL_EYE_PLANE_ABSOLUTE_NV            = $855c;
  // NV_texgen_emboss
  GL_EMBOSS_LIGHT_NV                = $855d;
  GL_EMBOSS_CONSTANT_NV               = $855e;
  GL_EMBOSS_MAP_NV                 = $855f;
  // EXT_light_max_exponent
  GL_MAX_SHININESS_EXT              = $8504;
  GL_MAX_SPOT_EXPONENT_EXT            = $8505;
  // EXT_texture_env_combine
  GL_COMBINE_EXT                   = $8570;
  GL_COMBINE_RGB_EXT                = $8571;
  GL_COMBINE_ALPHA_EXT               = $8572;
  GL_RGB_SCALE_EXT                 = $8573;
  GL_ADD_SIGNED_EXT                 = $8574;
  GL_INTERPOLATE_EXT                = $8575;
  GL_CONSTANT_EXT                  = $8576;
  GL_PRIMARY_COLOR_EXT               = $8577;
  GL_PREVIOUS_EXT                  = $8578;
  GL_SOURCE0_RGB_EXT                = $8580;
  GL_SOURCE1_RGB_EXT                = $8581;
  GL_SOURCE2_RGB_EXT                = $8582;
  GL_SOURCE0_ALPHA_EXT               = $8588;
  GL_SOURCE1_ALPHA_EXT               = $8589;
  GL_SOURCE2_ALPHA_EXT               = $858A;
  GL_OPERAND0_RGB_EXT                = $8590;
  GL_OPERAND1_RGB_EXT                = $8591;
  GL_OPERAND2_RGB_EXT                = $8592;
  GL_OPERAND0_ALPHA_EXT               = $8598;
  GL_OPERAND1_ALPHA_EXT               = $8599;
  GL_OPERAND2_ALPHA_EXT               = $859A;

  // NV_texture_env_combine4
  GL_COMBINE4_NV                   = $8503;
  GL_SOURCE3_RGB_NV                 = $8583;
  GL_SOURCE3_ALPHA_NV                = $858B;
  GL_OPERAND3_RGB_NV                 = $8593;
  GL_OPERAND3_ALPHA_NV               = $859B;

  // EXT_texture_filter_anisotropic
  GL_TEXTURE_MAX_ANISOTROPY_EXT        = $84fe;
  GL_MAX_TEXTURE_MAX_ANISOTROPY_EXT    = $84ff;

  // EXT_texture_lod_bias
  GL_MAX_TEXTURE_LOD_BIAS_EXT          = $84fd;
  GL_TEXTURE_FILTER_CONTROL_EXT        = $8500;
  GL_TEXTURE_LOD_BIAS_EXT             = $8501;

type

  PGLBoolean = ^GLBoolean;
  // EXT_vertex_array
  PFNGLARRAYELEMENTEXTPROC      = procedure(i : Glint); stdcall;
```

```
  PFNGLCOLORPOINTEREXTPROC      = procedure(size : Glint; ctype : Glenum; stride, count : Glsizei;
ptr : Pointer); stdcall;
  PFNGLDRAWARRAYSEXTPROC  = procedure(mode : Glenum; first : Glint; count : Glsizei); stdcall;
  PFNGLEDGEFLAGPOINTEREXTPROC   = procedure(stride : Glsizei; count : Glsizei; pointer :
PGLBoolean); stdcall;
  PFNGLGETPOINTERVEXTPROC       = procedure(pname : Glenum; params : Pointer); stdcall;
  PFNGLINDEXPOINTEREXTPROC      = procedure(types : Glenum; stride, count : Glsizei; ptr :
Pointer); stdcall;
  PFNGLNORMALPOINTEREXTPROC     = procedure(types : Glenum; stride, count : Glsizei; ptr :
Pointer); stdcall;
  PFNGLTEXCOORDPOINTEREXTPROC   = procedure(size : Glint; types : Glenum; stride, count :
Glsizei; ptr : Pointer); stdcall;
  PFNGLVERTEXPOINTEREXTPROC     = procedure(size : Glint; types : Glenum; stride, count : Glsizei;
ptr : Pointer); stdcall;

  // EXT_color_subtable
  PFNGLCOLORSUBTABLEEXTPROC     = procedure(target : Glenum; start, count : Glsizei; format,
types : Glenum; table : Pointer); stdcall;

  // EXT_color_table
  PFNGLCOLORTABLEEXTPROC        = procedure(target, internalformat : Glenum; width : Glsizei;
format, types : Glenum; table : Pointer); stdcall;
  PFNGLCOPYCOLORTABLEEXTPROC    = procedure(target, internalformat : Glenum; x, y : Glint; width
: Glsizei); stdcall;
  PFNGLGETCOLORTABLEEXTPROC     = procedure(target, format, types  : Glenum; table : Pointer);
stdcall;
  PFNGLGETCOLORTABLEPARAMETERFVEXTPROC = procedure(target, pname : Glenum; params :
PGLfloat); stdcall;
  PFNGLGETCOLORTABLEPARAMETERIVEXTPROC = procedure(target, pname : Glenum; params :
PGLint); stdcall;

  // EXT_compiled_vertex_array
  PFNGLLOCKARRAYSEXTPROC        = procedure(first : Glint; count : Glsizei); stdcall;
  PFNGLUNLOCKARRAYSEXTPROC      = procedure(); stdcall;

  // WIN_swap_hint
  PFNGLADDSWAPHINTRECTWINPROC   = procedure(x, y : Glint; width, height : Glsizei); stdcall;

  // EXT_point_parameter
  PFNGLPOINTPARAMETERFEXTPROC   = procedure(pname : Glenum; param : Glfloat); stdcall;
  PFNGLPOINTPARAMETERFVEXTPROC  = procedure(pname : Glenum; params : PGLfloat); stdcall;

  // ARB_multitexture
  PFNGLMULTITEXCOORD1DARBPROC   = procedure(target : Glenum; s : Gldouble); stdcall;
  PFNGLMULTITEXCOORD1DVARBPROC  = procedure(target : Glenum; v : PGLdouble); stdcall;
  PFNGLMULTITEXCOORD1FARBPROC   = procedure(target : Glenum; s : Glfloat); stdcall;
  PFNGLMULTITEXCOORD1FVARBPROC  = procedure(target : Glenum; v : PGLfloat); stdcall;
  PFNGLMULTITEXCOORD1IARBPROC   = procedure(target : Glenum; s : Glint); stdcall;
  PFNGLMULTITEXCOORD1IVARBPROC  = procedure(target : Glenum; v : PGLint); stdcall;
  PFNGLMULTITEXCOORD1SARBPROC   = procedure(target : Glenum; s : Glshort); stdcall;
  PFNGLMULTITEXCOORD1SVARBPROC  = procedure(target : Glenum; v : PGLshort); stdcall;

  PFNGLMULTITEXCOORD2DARBPROC   = procedure(target : Glenum; s, t : Gldouble); stdcall;
  PFNGLMULTITEXCOORD2DVARBPROC  = procedure(target : Glenum; v : PGLdouble); stdcall;
  PFNGLMULTITEXCOORD2FARBPROC   = procedure(target : Glenum; s, t : Glfloat); stdcall;
  PFNGLMULTITEXCOORD2FVARBPROC  = procedure(target : Glenum; v : PGLfloat); stdcall;
  PFNGLMULTITEXCOORD2IARBPROC   = procedure(target : Glenum; s, t : Glint); stdcall;
  PFNGLMULTITEXCOORD2IVARBPROC  = procedure(target : Glenum; v : PGLint); stdcall;
```

```
PFNGLMULTITEXCOORD2SARBPROC   = procedure(target : Glenum; s, t : Glshort); stdcall;
PFNGLMULTITEXCOORD2SVARBPROC  = procedure(target : Glenum; v : PGLshort); stdcall;

PFNGLMULTITEXCOORD3DARBPROC   = procedure(target : Glenum; s, t, r : Gldouble); stdcall;
PFNGLMULTITEXCOORD3DVARBPROC  = procedure(target : Glenum; v : PGLdouble); stdcall;
PFNGLMULTITEXCOORD3FARBPROC   = procedure(target : Glenum; s, t, r : Glfloat); stdcall;
PFNGLMULTITEXCOORD3FVARBPROC  = procedure(target : Glenum; v : PGLfloat); stdcall;
PFNGLMULTITEXCOORD3IARBPROC   = procedure(target : Glenum; s, t, r : Glint); stdcall;
PFNGLMULTITEXCOORD3IVARBPROC  = procedure(target : Glenum; v : PGLint); stdcall;
PFNGLMULTITEXCOORD3SARBPROC   = procedure(target : Glenum; s, t, r : Glshort); stdcall;
PFNGLMULTITEXCOORD3SVARBPROC  = procedure(target : Glenum; v : PGLshort); stdcall;

PFNGLMULTITEXCOORD4DARBPROC   = procedure(target : Glenum; s, t, r, q : Gldouble); stdcall;
PFNGLMULTITEXCOORD4DVARBPROC  = procedure(target : Glenum; v : PGLdouble); stdcall;
PFNGLMULTITEXCOORD4FARBPROC   = procedure(target : Glenum; s, t, r, q : Glfloat); stdcall;
PFNGLMULTITEXCOORD4FVARBPROC  = procedure(target : Glenum; v : PGLfloat); stdcall;
PFNGLMULTITEXCOORD4IARBPROC   = procedure(target : Glenum; s, t, r, q : Glint); stdcall;
PFNGLMULTITEXCOORD4IVARBPROC  = procedure(target : Glenum; v : PGLint); stdcall;
PFNGLMULTITEXCOORD4SARBPROC   = procedure(target : Glenum; s, t, r, q : Glshort); stdcall;
PFNGLMULTITEXCOORD4SVARBPROC  = procedure(target : Glenum; v : PGLshort); stdcall;

PFNGLACTIVETEXTUREARBPROC     = procedure(target : Glenum); stdcall;
PFNGLCLIENTACTIVETEXTUREARBPROC = procedure(target : Glenum); stdcall;

// SGIS_multitexture
PFNGLMULTITEXCOORD1DSGISPROC  = procedure(target : Glenum; s : Gldouble); stdcall;
PFNGLMULTITEXCOORD1DVSGISPROC = procedure(target : Glenum; v : PGLdouble); stdcall;
PFNGLMULTITEXCOORD1FSGISPROC  = procedure(target : Glenum; s : Glfloat); stdcall;
PFNGLMULTITEXCOORD1FVSGISPROC = procedure(target : Glenum; v : PGLfloat); stdcall;
PFNGLMULTITEXCOORD1ISGISPROC  = procedure(target : Glenum; s : Glint); stdcall;
PFNGLMULTITEXCOORD1IVSGISPROC = procedure(target : Glenum; v : PGLint); stdcall;
PFNGLMULTITEXCOORD1SSGISPROC  = procedure(target : Glenum; s : Glshort); stdcall;
PFNGLMULTITEXCOORD1SVSGISPROC = procedure(target : Glenum; v : PGLshort); stdcall;

PFNGLMULTITEXCOORD2DSGISPROC  = procedure(target : Glenum; s, t :  Gldouble); stdcall;
PFNGLMULTITEXCOORD2DVSGISPROC = procedure(target : Glenum; v : PGLdouble); stdcall;
PFNGLMULTITEXCOORD2FSGISPROC  = procedure(target : Glenum; s, t : Glfloat); stdcall;
PFNGLMULTITEXCOORD2FVSGISPROC = procedure(target : Glenum; v : PGLfloat); stdcall;
PFNGLMULTITEXCOORD2ISGISPROC  = procedure(target : Glenum; s, t : Glint); stdcall;
PFNGLMULTITEXCOORD2IVSGISPROC = procedure(target : Glenum; v : PGLint); stdcall;
PFNGLMULTITEXCOORD2SSGISPROC  = procedure(target : Glenum; s, t : Glshort); stdcall;
PFNGLMULTITEXCOORD2SVSGISPROC = procedure(target : Glenum; v : PGLshort); stdcall;

PFNGLMULTITEXCOORD3DSGISPROC  = procedure(target : Glenum; s, t, r : Gldouble); stdcall;
PFNGLMULTITEXCOORD3DVSGISPROC = procedure(target : Glenum; v : PGLdouble); stdcall;
PFNGLMULTITEXCOORD3FSGISPROC  = procedure(target : Glenum; s, t, r : Glfloat); stdcall;
PFNGLMULTITEXCOORD3FVSGISPROC = procedure(target : Glenum; v : PGLfloat); stdcall;
PFNGLMULTITEXCOORD3ISGISPROC  = procedure(target : Glenum; s, t, r : Glint); stdcall;
PFNGLMULTITEXCOORD3IVSGISPROC = procedure(target : Glenum; v : PGLint); stdcall;
PFNGLMULTITEXCOORD3SSGISPROC  = procedure(target : Glenum; s, t, r : Glshort); stdcall;
PFNGLMULTITEXCOORD3SVSGISPROC = procedure(target : Glenum; v : PGLshort); stdcall;

PFNGLMULTITEXCOORD4DSGISPROC  = procedure(target : Glenum; s, t, r, q : Gldouble); stdcall;
PFNGLMULTITEXCOORD4DVSGISPROC = procedure(target : Glenum; v : PGLdouble); stdcall;
PFNGLMULTITEXCOORD4FSGISPROC  = procedure(target : Glenum; s, t, r, q : Glfloat); stdcall;
PFNGLMULTITEXCOORD4FVSGISPROC = procedure(target : Glenum; v : PGLfloat); stdcall;
PFNGLMULTITEXCOORD4ISGISPROC  = procedure(target : Glenum; s, t, r, q : Glint); stdcall;
PFNGLMULTITEXCOORD4IVSGISPROC = procedure(target : Glenum; v : PGLint); stdcall;
```

```
  PFNGLMULTITEXCOORD4SSGISPROC  = procedure(target : Glenum; s, t, r, q : Glshort); stdcall;
  PFNGLMULTITEXCOORD4SVSGISPROC = procedure(target : Glenum; v : PGLshort); stdcall;

  PFNGLMULTITEXCOORDPOINTERSGISPROC = procedure(target : Glenum; size : Glint; types :
Glenum; stride : Glsizei; ptr : Pointer); stdcall;
  PFNGLSELECTTEXTURESGISPROC    = procedure(target : Glenum); stdcall;
  PFNGLSELECTTEXTURECOORDSETSGISPROC = procedure(target : Glenum); stdcall;

  // EXT_vertex_weighting
  PFNGLVERTEXWEIGHTFEXTPROC     = procedure(weight : Glfloat); stdcall;
  PFNGLVERTEXWEIGHTFVEXTPROC    = procedure(weight : PGLfloat); stdcall;
  PFNGLVERTEXWEIGHTPOINTEREXTPROC = procedure(size : Glsizei; types : Glenum; stride : Glsizei;
ptr : Pointer); stdcall;

  // EXT_blend_color
  PFNGLBLENDCOLOREXTPROC        = procedure(red, green, blue, alpha : Glclampf); stdcall;

  // EXT_blend_minmax
  PFNGLBLENDEQUATIONEXTPROC     = procedure(mode : Glenum); stdcall;

  // EXT_fog_coord
  PFNGLFOGCOORDDEXTPROC         = procedure(fog : Gldouble); stdcall;
  PFNGLFOGCOORDDVEXTPROC        = procedure(fog : PGLdouble); stdcall;
  PFNGLFOGCOORDFEXTPROC         = procedure(fog : Glfloat); stdcall;
  PFNGLFOGCOORDFVEXTPROC        = procedure(fog : PGLfloat); stdcall;
  PFNGLFOGCOORDPOINTEREXTPROC   = procedure(size : Glsizei; types : Glenum; stride : Glsizei; ptr
: Pointer); stdcall;

  // EXT_secondary_color
  PFNGLSECONDARYCOLOR3BEXTPROC  = procedure(red, green, blue : Glbyte); stdcall;
  PFNGLSECONDARYCOLOR3BVEXTPROC = procedure(v : PGLbyte); stdcall;
  PFNGLSECONDARYCOLOR3DEXTPROC  = procedure(red, green, blue : Gldouble); stdcall;
  PFNGLSECONDARYCOLOR3DVEXTPROC = procedure(v : PGLdouble); stdcall;
  PFNGLSECONDARYCOLOR3FEXTPROC  = procedure(red, green, blue : Glfloat); stdcall;
  PFNGLSECONDARYCOLOR3FVEXTPROC = procedure(v : PGLfloat); stdcall;
  PFNGLSECONDARYCOLOR3IEXTPROC  = procedure(red, green, blue : Glint); stdcall;
  PFNGLSECONDARYCOLOR3IVEXTPROC = procedure(v : PGLint); stdcall;
  PFNGLSECONDARYCOLOR3SEXTPROC  = procedure(red, green, blue : Glshort); stdcall;
  PFNGLSECONDARYCOLOR3SVEXTPROC = procedure(v : PGLshort); stdcall;
  PFNGLSECONDARYCOLOR3UBEXTPROC = procedure(red, green, blue : Glubyte); stdcall;
  PFNGLSECONDARYCOLOR3UBVEXTPROC = procedure(v : PGLubyte); stdcall;
  PFNGLSECONDARYCOLOR3UIEXTPROC = procedure(red, green, blue : Gluint); stdcall;
  PFNGLSECONDARYCOLOR3UIVEXTPROC = procedure(v : PGLuint); stdcall;
  PFNGLSECONDARYCOLOR3USEXTPROC = procedure(red, green, blue : Glushort); stdcall;
  PFNGLSECONDARYCOLOR3USVEXTPROC = procedure(v : PGLushort); stdcall;
  PFNGLSECONDARYCOLORPOINTEREXTPROC = procedure(size : Glint; types : Glenum; stride :
Glsizei; ptr : Pointer); stdcall;

  // NV_vertex_array_range
  PFNGLFLUSHVERTEXARRAYRANGENVPROC = procedure(); stdcall;
  PFNGLVERTEXARRAYRANGENVPROC   = procedure(size : Glsizei; ptr : Pointer); stdcall;
  PFNWGLALLOCATEMEMORYNVPROC    = function(size : Glsizei; readfreq, writefreq, priority : Glfloat)
: Pointer; stdcall;
  PFNWGLFREEMEMORYNVPROC        = procedure(ptr : Pointer); stdcall;

  // NV_register_combiners
  PFNGLCOMBINERPARAMETERFVNVPROC = procedure(pname : Glenum; params : PGLfloat); stdcall;
  PFNGLCOMBINERPARAMETERFNVPROC = procedure(pname : Glenum; param : Glfloat); stdcall;
```

```
  PFNGLCOMBINERPARAMETERIVNVPROC = procedure(pname : Glenum; params : PGLint); stdcall;
  PFNGLCOMBINERPARAMETERINVPROC = procedure(pname : Glenum; param : Glint); stdcall;
  PFNGLCOMBINERINPUTNVPROC = procedure(stage, portion, variable, input, mapping,
componentUsage : Glenum); stdcall;
  PFNGLCOMBINEROUTPUTNVPROC = procedure(stage, portion, abOutput, cdOutput, sumOutput,
scale, bias : Glenum; abDotProduct, cdDotProduct, muxSum : Glboolean); stdcall;
  PFNGLFINALCOMBINERINPUTNVPROC = procedure(variable, input, mapping, componentUsage :
Glenum); stdcall;
  PFNGLGETCOMBINERINPUTPARAMETERFVNVPROC = procedure(stage, portion, variable, pname :
Glenum; params : PGLfloat); stdcall;
  PFNGLGETCOMBINERINPUTPARAMETERIVNVPROC = procedure(stage, portion, variable, pname :
Glenum; params : PGLint); stdcall;
  PFNGLGETCOMBINEROUTPUTPARAMETERFVNVPROC = procedure(stage, portion, pname : Glenum;
params : PGLfloat); stdcall;
  PFNGLGETCOMBINEROUTPUTPARAMETERIVNVPROC = procedure(stage, portion, pname : Glenum;
params : PGLint); stdcall;
  PFNGLGETFINALCOMBINERINPUTPARAMETERFVNVPROC = procedure(variable, pname : Glenum;
params : PGLfloat); stdcall;
  PFNGLGETFINALCOMBINERINPUTPARAMETERIVNVPROC = procedure(variable, pname : Glenum;
params : PGLint); stdcall;

  // WGL_EXT_swap_control
  PFNWGLSWAPINTERVALEXTPROC     = function(interval : Integer) : Integer; stdcall;
  PFNWGLGETSWAPINTERVALEXTPROC  = function() : Integer; stdcall;

  function isSupported(extension : String) : Boolean;

implementation

function isSupported(extension : String) : Boolean;
var
  extensions : Pchar;
  where : Pchar;
  start, terminator : Pchar;

  hasSpaces : Integer;
begin
  hasSpaces := Pos(Pchar(extension), ' ');
  if (hasSpaces <> 0) or (extension = '') then begin
    Result := False;
    Exit;
  end;

  extensions := glGetString(GL_EXTENSIONS);
  start := extensions;
  while (True) do begin
    where := StrPos(start, Pchar(extension));
    if (where = nil) then begin
      Result := False;
      Exit;
    end;

    terminator := where + Length(extension);
    if (where = start) or ((where-1)^ = ' ') then
      if (terminator^ = ' ') or (terminator^ = #0) then begin
        Result := True;
        Exit;
      end;
```

```
    start := terminator;
  end;
  Result := False;
end;


end.
```

**برنامه فصل** :

```
unit ch27;
interface

uses
  Windows, Messages, Classes, Graphics, Controls,
  Forms,Dialogs, OpenGL ,SPF , glExt ;

type
  TForm1 = class(TForm)
    procedure FormCreate(Sender: TObject);
    procedure FormResize(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure FormPaint(Sender: TObject);
    procedure FormKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
  private
    { Private declarations }

       procedure Idle(Sender: TObject; var Done: Boolean);

  public
    { Public declarations }
  end;

var
  Form1: TForm1;
implementation

{$R *.dfm}
var

  ElapsedTime : Integer;      // Elapsed time between frames

  // Other vaiables
  TorusDL : glUint;
  SeperateSpecular : Boolean; // if Seperate Specular is Enabled
  Segments : Integer;         // quality of torus

  // Lights
  LightPos : Array [0..3] of GLfloat = (0.0, 0.0, 0.0, 1.0);
  MatDiffuse : Array [0..3] of GLfloat = (0.8, 0.8, 0.8, 1.0);
  MatSpecular: Array [0..3] of GLfloat = (1.0, 1.0, 1.0, 1.0);
  MatShine   : GLfloat = 50.0;

  f_Hdc : longInt;
```

```
  DemoStart, LastTime : Integer;

{------------------------------------------------------------------}
{  Function to convert int to string. (No sysutils = smaller EXE)  }
{------------------------------------------------------------------}
// using SysUtils increase file size by 100K
function IntToStr(Num : Integer) : String;
begin
  Str(Num, result);
end;

{------------------------------------------------------------------}
{  Function to draw a torus                                        }
{------------------------------------------------------------------}
procedure CreateTorus(TubeRadius, Radius : GLfloat;
                      Sides, Rings : Integer);
var I, J : Integer;
    theta, phi, theta1 : GLfloat;
    cosTheta, sinTheta : GLfloat;
    cosTheta1, sinTheta1 : GLfloat;
    ringDelta, sideDelta : GLfloat;
    cosPhi, sinPhi, dist : GLfloat;
begin
  sideDelta := 2.0 * Pi / Sides;
  ringDelta := 2.0 * Pi / rings;
  theta := 0.0;
  cosTheta := 1.0;
  sinTheta := 0.0;

  TorusDL :=glGenLists(1);
  glNewList(TorusDL, GL_COMPILE);
    for i := rings - 1 downto 0 do
    begin
      theta1 := theta + ringDelta;
      cosTheta1 := cos(theta1);
      sinTheta1 := sin(theta1);
      glBegin(GL_QUAD_STRIP);
        phi := 0.0;
        for j := Sides downto 0 do
        begin
          phi := phi + sideDelta;
          cosPhi := cos(phi);
          sinPhi := sin(phi);
          dist := Radius + (TubeRadius * cosPhi);

          glNormal3f(cosTheta1 * cosPhi,
                     -sinTheta1 * cosPhi, sinPhi);
          glVertex3f(cosTheta1 * dist, -sinTheta1 * dist,
                     TubeRadius * sinPhi);

          glNormal3f(cosTheta * cosPhi,
                     -sinTheta * cosPhi, sinPhi);
          glVertex3f(cosTheta * dist,
                     -sinTheta * dist, TubeRadius * sinPhi);
        end;
      glEnd();
      theta := theta1;
```

```
      cosTheta := cosTheta1;
      sinTheta := sinTheta1;
    end;
  glEndList();
end;

{------------------------------------------------------------------}
{  Function to draw the actual scene                               }
{------------------------------------------------------------------}
procedure glDraw();
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,-14);

  glRotatef(ElapsedTime/20, 1, 0, 0);
  glRotatef(ElapsedTime/16, 0, 1, 0);

  glCallList(TorusDL);

  SwapBuffers(f_Hdc);
end;


{------------------------------------------------------------------}
{  Initialise OpenGL                                               }
{------------------------------------------------------------------}
procedure glInit();
begin
  glClearColor(0.0, 0.0, 0.0, 0.0);   // Black Background
  glShadeModel(GL_SMOOTH);            // Enables Smooth Color Shading
  glClearDepth(1.0);                  // Depth Buffer Setup
  glEnable(GL_DEPTH_TEST);            // Enable Depth Testing
  glDepthFunc(GL_LESS);                        // The Type Of Depth Test To Do

  //Realy Nice perspective calculations
  glHint(GL_PERSPECTIVE_CORRECTION_HINT, GL_NICEST);

  // Turn on OpenGL lighting
  glEnable(GL_LIGHTING);

  // Create a light
  glLightfv(GL_LIGHT0, GL_POSITION, @LightPos);
  glEnable(GL_LIGHT0);

  glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, @MatDiffuse);
  glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, @MatSpecular);
  glMaterialfv(GL_FRONT_AND_BACK, GL_SHININESS, @MatShine);

  SeperateSpecular :=FALSE;
  Segments :=64;
  CreateTorus(1, 3, Segments, Segments);
end;

{------------------------------------------------------------------}
{  Handle window resize                                            }
{------------------------------------------------------------------}
```

```
procedure glResizeWnd(Width, Height : Integer);
begin
  if (Height = 0) then     // prevent divide by zero exception
    Height := 1;
  // Set the viewport for the OpenGL window
  glViewport(0, 0, Width, Height);
  glMatrixMode(GL_PROJECTION);      // Change Matrix Mode to Projection
  glLoadIdentity();                 // Reset View
  // Do the perspective calculations. Last value = max clipping depth
  gluPerspective(45.0, Width/Height, 1.0, 100.0);

  glMatrixMode(GL_MODELVIEW);         // Return to the modelview matrix
  glLoadIdentity();                   // Reset View
end;

procedure determine_hardware_acceleration(
               Handle: HDC;ColorBits,DepthBufferBits:integer);

var
  generic_format : integer;
  generic_accelerated : integer;

  pfd: TPixelFormatDescriptor;
  nPixelFormat: Integer;
  pfd_new : PIXELFORMATDESCRIPTOR ;
begin

  FillChar(pfd, SizeOf(pfd), 0);

  with pfd do begin
    nSize     := sizeof(pfd);            // Size of this structure
    nVersion  := 1;                      // Version number
    dwFlags   := PFD_SUPPORT_OPENGL Or PFD_DRAW_TO_WINDOW
              Or  PFD_TYPE_RGBA
              or PFD_DOUBLEBUFFER {to avoid flickering} ; // Flags
    iPixelType:= PFD_TYPE_RGBA;          // RGBA pixel values
    cColorBits:= ColorBits;              // 24-bit color
    cDepthBits:= DepthBufferBits;        // 32-bit depth buffer
    iLayerType:= PFD_MAIN_PLANE;         // Layer type
  end;

  nPixelFormat := ChoosePixelFormat(Handle, @pfd);

  DescribePixelFormat (Handle, npixelFormat,
                  sizeof(PIXELFORMATDESCRIPTOR), pfd_new);

    generic_format :=
              pfd_new.dwFlags and PFD_GENERIC_FORMAT;

    generic_accelerated :=
              pfd_new.dwFlags and PFD_GENERIC_ACCELERATED;

    if (generic_format=1 and not(generic_accelerated)) then
    begin
     MessageBox(0,'softdware','3DInfo',
                    MB_OK+MB_ICONINFORMATION);
    end;
```

```
    if (generic_format=1 and generic_accelerated) then
    begin
     MessageBox(0,'hardware - MCD','3DInfo',
                    MB_OK+MB_ICONINFORMATION);
    end;

    if ( not(generic_format)=1 and not(generic_accelerated)) then
    begin
      MessageBox(0,'hardware - ICD','3DInfo',
                    MB_OK+MB_ICONINFORMATION);
    end;
end;


procedure glInfo();
var
  vendor,renderer,version,extension : pchar;
begin
         vendor := pchar (glGetString(GL_VENDOR));
         renderer := pchar ( glGetString(GL_RENDERER));
         version := pchar (glGetString(GL_VERSION));
         extension := pchar (glGetString(GL_EXTENSIONS));
      MessageBox(0,pchar('vendor: '+vendor+#13#10+
                'renderer: '+renderer+#13#10+
                'version: '+version+#13#10+
                'extension: '+extension),'glInfo',
                MB_OK+MB_ICONINFORMATION);
end;

procedure TForm1.FormCreate(Sender: TObject);
begin

 DemoStart := GetTickCount();   // Get Time when demo started
 f_Hdc := GetDC( handle );
 SetDCPixelFormat(f_Hdc,16,16);// Create a rendering context.
 glInit();
 Application.OnIdle := Idle;
 determine_hardware_acceleration(f_Hdc,16,16);
 glInfo();

end;

procedure TForm1.FormResize(Sender: TObject);
begin

   wglMakeCurrent(f_Hdc,hrc); //activate the RC
     glResizeWnd(width,height);
   InvalidateRect(Handle, nil, False);// Draw the scene.

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
  glDraw() // Draw the scene.
end;

procedure TForm1.Idle(Sender: TObject; var Done: Boolean);
begin

  LastTime :=ElapsedTime;

   // Calculate Elapsed Time
  ElapsedTime :=GetTickCount() - DemoStart;

  // Average it out for smoother movement
  ElapsedTime := (LastTime + ElapsedTime) DIV 2;

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

end;

procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin

  // Turn on specular highlights if supported
  if (key = ord('N')) then
  begin
    if SeperateSpecular then  // If its enabled, then disable
      glLightModel(GL_LIGHT_MODEL_COLOR_CONTROL, GL_SINGLE_COLOR)
    else
    begin
      if isSupported('GL_EXT_separate_specular_color') then
        glLightModel(GL_LIGHT_MODEL_COLOR_CONTROL,
                GL_SEPARATE_SPECULAR_COLOR)
      else
        glLightModel(GL_LIGHT_MODEL_COLOR_CONTROL,
                GL_SINGLE_COLOR);
    end;
    SeperateSpecular :=NOT(SeperateSpecular);
  end;
  if key=VK_ADD then
  begin
    Segments :=Segments*2;
    CreateTorus(1, 3, Segments, Segments);
  end;
  if key=VK_SUBTRACT then
  begin
    Segments :=Segments DIV 2;
    if Segments < 4 then
      Segments :=4;
    CreateTorus(1, 3, Segments, Segments);
  end;
     InvalidateRect(Handle, nil, False);//  Draw the scene.
end;

end.
```