فصل سوم

مرجع توابع مرتبط سازنده OpenGL و ویندوز

مرجع wgl (Windows GL)

wgl که به صورت wiggle تلفظ می شود مانند چسبی است که OpenGL و Win32 API را به هم متصل می سازد. این توابع برای کنترل فرمت نقطه ای بکارمی روند. مواردی راکه شما می توانید انتخاب کنید : عمق رنگ ، بافر دوگانه ، بافر استریو ، استفاده از رنگ های RGB و یا پالت ها ، استفاده ازصفحات آلفا (شفاف) و غیره، می باشد. با استفاده از توابع زیر می توان فرمت نقطه ای را وابسته به سخت افزار کرد و یا خیر . البته مایکروسافت نگارشی عمومی (Generic) از OpenGL را ارائه داده است که بر روی تمام کارت های گرافیکی کار می کند. توابع wgl به شرح زیر می باشند :

توابع RC

تابع	توضیح
wglCreateContext	یک زمینه ارائه (RC) جدید ایجاد می کند.
wglMakeCurrent	ریسمان (Thread) جاری را به RC تنظیم می کند و اینکار باید قبل از اجرای هرگونه تابعی از OpenGL صورت گیرد .
wglGetCurrentContext	دستگیره (Handle) ریسمان جاری RC را برمی گرداند .
wglGetCurrentDC	دستگیره DC مربوط به ریسمان جاری RC را بر می گرداند .
wglDeleteContext	RC را حذف می کند .

توابع فونت و متن

تابع	توضیح
wglUseFontBitmap	یک مجموعه لیست نمایشی از کاراکترهای بیت مپ گونه ایجاد می کند. کاراکتر ها از DC معینی گرفته می شوند که حاوی فونت جاری است .
wglUseFontOutLines	مجموعه ای از لیست های نمایشی بر مبنای فونت OutLine از DC برای استفاده با RC جاری ایجاد می کند . لیست های نمایشی برای ترسیم کاراکترهای سه بعدی فونت های TrueType بکار می روند .

توابع مربوط به لایه های رویی ، زیری و صفحه اصلی

تابع	توضیح
wglCopyContext	گروهی از حالت های ارائه انتخاب شده را از یک RC به RC دیگر کپی می کند .
wglCreateLayerContext	یک RC جدید برای ترسیم در لایه مشخص شده بر روی DC ایجاد می کند .
wglDescribeLayerPlane	اطلاعاتی را در مورد صفحات لایه های مربوط به فرمت نقطه ای داده شده ارائه می دهد .
wglGetLayerPaletteEntries	مدخل های پالت رنگی را از یک لایه DC بر می گرداند .
wglRealizeLayerPalette	پالت لایه RGBA را آغاز می کند و یا مدخل های پالت لایه داده شده را روی پالت فیزیکی ترسیم می کند .
wglSetLayerPaletteEntries	مدخل های پالت در یک لایه داده شده را برای یک DC معین تنظیم می کند .
wglSwapLayerBuffers	بافر رویی و زیری را در صفحات رویی ، زیری و اصلی ، در صفحه معین شده با DC تعویض می کند .

توضیح :

لیست های نمایشی راهی هستند برای ذخیره کردن فرامین ارائه .

توابع متفرقه

تابع	توضیح
wglShareLists	RC را قادر می سازد تا فضای لیست نمایشی را با سایر RC ها به اشتراک گذارد .
wglGetProcAddress	آدرس توابع OpenGL را برای استفاده در RC جاری بر می گرداند.

مرجع توابع فرمت نقطه ای و پالت ها

فرمت نقطه ای بیانگر خواص زمینه(Context) OpenGL می باشد . فرمت های نقطه ای به همراه پالتها مدخلی برای زمینه مناسب برنامه ای که خلق می شود ، هستند . برپایی و تنظیم فرمت نقطه ای یکی از ظریف ترین قسمت های برنامه نویسی OpenGL در Win32 می باشد . توابع متعددی برای این منظور وجود دارند ، که به شرح زیر ارائه می گردند :

تابع	توضیح
ChoosePixelFormat	نزدیکترین فرمت نقطه ایDC ، به فرمت نقطه ای مورد نظر را برمی گزیند.
SetPixelFormat	فرمت نقطه ای DC جاری را به فرمت نقطه ای معین شده با اندیس مربوطه ، تنظیم می کند . *همواره* خروجی این تابع را بررسی کنید! زیرا شکست آن مساوی با عدم اجرای برنامه شما می باشد .
GetPixelFormat	اندیس فرمت نقطه ای DC جاری را بر می گزیند .
DescribePixelFormat	اندیس فرمت نقطه ای و DC داده شده در یک رکورد به نام PixelFormatDescriptor باخواص فرمت نقطه ای ، قرار می گیرد و در نهایت حداکثر تعداد فرمت های نقطه ای مهیا را بر می گرداند .

اکثر اوقات تابع ChoosePixelFormat برای انتخاب فرمت نقطه ای کافی می باشد ، اما هنگامیکه دقت بیشتری نیاز باشد ، سایر توابع باید بکار برده شود . روشی عالی برای انتخاب فرمت نقطه ای با خواصی معین این است که تمام فرمت های نقطه ای موجود برشمرده شوند و سپس با فرمت

نقطه ای مورد نظر مقایسه گردند و هنگامیکه یکی از آنها با تمام شرایط مطابقت داشت ، انتخاب شود .

در ادامه جزئیات رکورد PixelFormatDescriptor ارائه می شود (رکورد یا ساختار زیر با فرمت زبان C++ ارائه گردیده است) :

```
typedef struct tagPIXELFORMATDESCRIPTOR
{
WORD nSize;                 //sizeof(PIXELFORMATDESCRIPTOR)
WORD nVersion;              //1
DWORD dwFlags;
BYTE iPixelType;            //rgba or color indexed
BYTE cColorBits;            //# of color bitplanes
BYTE cRedBits;              //# red bitplanes
BYTE cRedShift;             //shift count for red bitplanes
BYTE cGreenBits;            //# green bitplanes
BYTE cGreenShift;           //shift count for green bitplanes
BYTE cBlueBits;             //# blue bitplanes
BYTE cBlueShift;            //shift count for blue bitplanes
BYTE cAlphaBits;            //not used in generic format
BYTE cAlphaShift;           //not used in generic format
BYTE cAccumBits;            //total # accum buffer bitplanes
BYTE cAccumRedBits;         //# red bitplanes in accum buffer
BYTE cAccumGreenBits;       //# green bitplanes in accum buffer
BYTE cAccumBlueBits;        //# blue bitplanes in accum buffer
BYTE cAccumAlphaBits;       //# alpha bitplanes in accum buffer
BYTE cDepthBits;            //depth of depth (z) buffer
BYTE cStencilBits;          //depth of stencil buffer
BYTE cAuxBuffers;           //not used in generic format
BYTE iLayerType;            //PFD_MAIN_PLANE only in generic format
BYTE bReserved;             //must be 0
DWORD dwLayerMask;
DWORD dwVisibleMask;
DWORD dwDamageMask;
} PIXELFORMATDESCRIPTOR;
;
```

اعضای این ساختار یا رکورد :

nSize : اندازه رکورد را معین می سازد . این مقدار باید مساوی sizeof(PIXELFORMATDESCRIPTOR) قرار گیرد .

nVersion : شماره نگارش رکورد را معین می کند که باید به ۱ تنظیم شود .

dwFlags : مجموعه ای از پرچم های بیتی که خواص بافر نقطه ای را معین می کنند و بدیهی است که می توان ترکیبی از آنها را با هم بکار برد (البته با در نظر گرفتن استثناهای ذکرشده) . این ثوابت به شرح زیر هستند :

مقدار	معنا
PFD_DRAW_TO_WINDOW	بافر می تواند روی پنجره ترسیم شود.
PFD_DRAW_TO_BITMAP	بافر می تواند روی بیتمپی در حافظه ترسیم شود.
PFD_SUPPORT_GDI	بافر، GDI را پشتیبانی می کند . این پرچم با PFD_DOUBLEBUFFER ناسازگار است .
PFD_SUPPORT_OPENGL	بافر ، ترسیمات OpenGL را پشتیبانی می کند . (البته منظور این است که این کارها را باید انجام دهد .)
PFD_GENERIC_FORMAT	فرمت نقطه ای که توسط نگارش نرم افزاری GDI پشتیبانی می شود و به فرمت عمومی نیز معروف است . اگر بکار برده نشود و بیت مربوطه خالی باشد ، فرمت نقطه ای بوسیله سخت افزار پشتیبانی می شود .
PFD_NEED_PALETTE	بافر ، نقاط RGBA را روی وسیله ای که توسط پالت راهبری می شود ، بکار می برد . برای بدست آوردن بهترین نتایج به پالت منطقی نیاز می باشد . پالت باید قبل از بکار بردن تابع wglMakeCurrent ، ایجاد و بهینه شده باشد .
PFD_DOUBLEBUFFER	بافر ، بافر دوگانه می باشد . این پرچم و PFD_SUPPORT_GDI ناسازگار هستند .
PFD_STEREO	بافر ، استریوسکوپیک می باشد . در نگارش عمومی پشتیبانی نمی شود .
PFD_NEED_SYSTEM_PALETTE	با سیستم هایی بکار می رود که سخت افزار آنها فقط از یک پالت سخت افزاری تشکیل شده است . اگر سخت افزار شما پالتهای چندگانه سخت افزاری را پشتیبانی می کند ، لزومی به استفاده از این پرچم نخواهید داشت .
PFD_GENERIC_ACCELERATED	فرمت نقطه ای که توسط شتاب دهنده های گرافیکی پشتیبانی می شود .
PFD_SWAP_LAYER_BUFFERS	معین می سازد که آیا وسیله می تواند لایه های مختلف را تعویض کند یا خیر . در غیر اینصورت تمام لایه ها به صورت یک گروه با هم تعویض می شوند . هنگامیکه از این پرچم استفاده می شود می توان از wglSwapLayerBuffers استفاده کرد.

پرچم های زیر را نیز هنگامیکه از تابع ChoosePixelFormat استفاده می کنید ، می توانید بکار برید :

مقدار	معنا
PFD_DOUBLE_BUFFER_DONTCARE	فرمت نقطه ای می تواند بافر دوگانه و یا یگانه باشد .
PFD_STEREO_DONTCARE	فرمت نقطه ای می تواند مونوسکوپیک یا استریوسکوپیک باشد .
PFD_DEPTH_DONTCARE	فرمت نقطه ای درخواست شده می تواند عمق بافر داشته باشد و یا خیر . برای انتخاب فرمت نقطه ای بدون عمق بافر باید از این پرچم استفاده کنید . در غیر اینصورت تنها فرمتهای نقطه ای با عمق بافر در نظر گرفته می شوند .

با استفاده از تابع glAddSwapHintRectWIN دو پرچم جدید زیر را نیز می توانید استفاده کنید :

مقدار	معنا
PFD_SWAP_COPY	محتویات بافر عقبی به بافر رویی کپی می شوند .
PFD_SWAP_EXCHANGE	محتویات دو بافر عقبی و رویی یکی می شوند .

iPixelType : نوع داده نقطه ای را معین می سازد و مقادیر زیر را می تواند داشته باشد :

مقدار	معنا
PFD_TYPE_RGBA	نقاط RGBA . در این حالت هر نقطه چهار جزء خواهد داشت : قرمز ، سبز ، آبی و آلفا .
PFD_TYPE_COLORINDEX	نقاط اندیس رنگی . هر نقطه یک اندیس رنگی را بکار می برد.

cColorBits : تعداد صفحات بیتی را در هر بافر رنگی مشخص می کند . آن برای نقاط نوع RGBA ، اندازه بافر رنگی می باشد به استثنای صفحات بیتی آلفا و برای نقاط اندیس رنگی ، اندازه بافر اندیس رنگی می باشد .

cRedBits : تعداد صفحات بیتی قرمز را در هر بافر رنگی RGBA معین می سازد.

cRedShift : تعداد شیفت های صفحات بیتی قرمز را در هر بافر رنگی RGBA معین می کند.

cGreenBits ، cGreenShift ، cBlueBits و cBluehift نیز همانند آنچه که در مورد صفحات بیتی قرمز ذکر شد می باشند .

cAlphaBits : تعداد صفحات بیتی آلفا را در هر بافر رنگی RGBA مشخص می کنند . صفحات بیتی آلفا پشتیبانی نمی شوند .

cAccumBits : تعداد کل صفحات بیتی را در بافر انباشتگی معین می سازند .

cAccumRedBits ، cAccumGreenBits ، cAccumBlueBits و cAccumAlphaBits تعداد صفحات بیتی قرمز ، سبز ، آبی و آلفا را در بافر انباشتگی مشخص می سازند .

cDepthBits : عمق بافر عمقی (محور z) را معین می کند .

cStencilBits : عمق بافر استنسیل را معلوم می کند .

cAuxBuffers : تعداد بافرهای کمکی را معین می سازد . بافرهای کمکی پشتیبانی نمی شوند .

iLayerType : در نگارش های قبلی OpenGL بکار می رفت و دیگر مصرفی ندارد .

bReserved : بکار نمی رود و باید صفر باشد .

dwLayerMask : در نگارش های قبلی OpenGL بکار می رفت و دیگر مصرفی ندارد .

dwVisibleMask : رنگ شفاف و یا اندیس صفحه زیرین را معین می کند .

dwDamageMask : در نگارش های قبلی OpenGL بکار می رفت و دیگر مصرفی ندارد .

باید خاطر نشان کرد در مواردی که جمله ((پشتیبانی نمی شود)) ذکر گردید ، بدین معنا است که این موارد در نگارشی از OpenGL که توسط مایکروسافت ارائه شده پشتیبانی نمی شوند و انواع مختلف سخت افزاری ممکن است به خوبی آنها را پشتیبانی نمایند و قسمتی از OpenGL را با تمهیدات سخت افزاری بهبود بخشند .

محدودیت های فرمت عمومی OpenGL که از طرف مایکروسافت ارائه شده است به قرار زیر هستند :

۱- برنامه نمی تواند مستقیما تصاویر OpenGL را بر روی چاپگرهای دو رنگ چاپ نماید . برنامه فقط می تواند مستقیما تصویر را به چاپگری بفرستد که حداقل ۴ بیت به ازای هر نقطه را پشتیبانی کند .

۲- OpenGL و GDI در یک پنجره با بافر دوگانه نمی توانند ترکیب شوند ؛ ولی اینکار در یک بافر یگانه می تواند صورت گیرد .

۳- به ازای هر پنجره ، پالت رنگی سخت افزاری وجود ندارد.

۴- پشتیبانی مستقیمی در مورد تخته برش ویندوز ، DDE ، MetaFiles و یا OLE وجود ندارد.

۵- کلاس Invertor 2 C++ این کتابخانه به آن ضمیمه نشده است .

۶- بعضی فرمت های نقطه ای مانند : لایه های رویی و زیری ، استریوسکوپیک ، صفحات بیتی آلفا و بافرهای کمکی پشتیبانی نمی شوند .

چند واژه جدید :

واژه های زیر بطور مکرر در این فصل بکار رفته اند که توضیحاتی در مورد آنها ضروری به نظر می رسد .

Rendering : (ارائه کردن ، رندر کردن)

پروسه ای که در آن رایانه تصاویری را از مدل ها ایجاد می کند .

Pixel : (نقطه)

کوچکترین عنصر مرئی که سخت افزار نمایشی می تواند آنرا نمایش دهد .

Bitplanes : (صفحات بیتی)

اطلاعات مربوط به نقاط (برای مثال رنگ آنها) در حافظه در صفحات بیتی سازمان می یابد. صفحه بیتی ناحیه ای از حافظه است که یک بیت اطلاعاتی را برای هر نقطه روی صفحه نگه داری می کند .

Framebuffer : (بافر چارچوب)

صفحات بیتی خودشان نیز در یک بافر چارچوب سازماندهی می شوند که آن تمام اطلاعات مورد نیاز برای کنترل نمایش رنگ و شدت هر نقطه روی صفحه را نگه داری می کند .

ایجاد واحد SPF :

قبل از شروع فصل بعد ، لازم است برای جلوگیری از تکرار مکررات ، واحدی (Unit) جدید بنام SPF را ایجاد کنیم تا زیر روال های برپایی فرمت نقطه ای را درون آن قرار دهیم . از این پس با فراخوانی واحد SPF در قسمت uses برنامه می توان به زیر روال های آن دسترسی داشت *و این امر در تمام فصول بعدی رعایت خواهد شد .*

```
unit SPF; // setup pixel format

interface
uses    { uses clause }
  Windows ;
var
  hrc: HGLRC;  // Permanent Rendering Context
```

```
procedure CleanUp(Handle: HDC); //Properly Kill The Window
procedure SetDCPixelFormat(Handle: HDC;ColorBits,DepthBufferBits:integer);

implementation

procedure CleanUp(Handle: HDC); //Properly Kill The Window
begin
 if hrc<> 0 then       //Is There A Rendering Context?
   begin
     //Are We Able To Release Dc and Rc contexts?
     if (not wglMakeCurrent(handle,0)) then
       MessageBox(0,'Release of DC and RC failed.'
                ,' Shutdown Error',MB_OK or MB_ICONERROR);
    //Are We Able To Delete The Rc?
     if (not wglDeleteContext(hRc)) then
      begin
       MessageBox(0,'Release of Rendering Context failed.',
                ' Shutdown Error',MB_OK or MB_ICONERROR);
       hRc:=0;   //Set Rc To Null
      end;
   end;
end;

procedure SetDCPixelFormat(Handle: HDC;ColorBits,DepthBufferBits:integer);
var
 pfd: TPixelFormatDescriptor;
 nPixelFormat: Integer;

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;        // 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);
 //Did We Find A Matching Pixelformat?
 if (nPixelFormat=0) then
   begin
   CleanUp(handle);                    //Reset The Display
    MessageBox(0,'Cant''t Find A Suitable PixelFormat.'
               ,'Error',MB_OK or MB_ICONEXCLAMATION);
    Halt(1); { Halt right here! }
   end;

 //Are We Able To Set The Pixelformat?
 if (not SetPixelFormat(Handle, nPixelFormat, @pfd)) then
   begin
    CleanUp(handle);                   //Reset The Display
    MessageBox(0,'Cant''t set PixelFormat.'
```

```
          ,'Error',MB_OK or MB_ICONEXCLAMATION);
    Halt(1); { Halt right here! }
   end;

 hrc := wglCreateContext(Handle);
 if (hRc=0) then
   begin
   CleanUp(handle);                    //Reset The Display
    MessageBox(0,'Cant''t create a GL rendering context.'
                ,'Error',MB_OK or MB_ICONEXCLAMATION);
    Halt(1); { Halt right here! }
   end;

 //Are We Able To Activate The Rendering Context?
 if (not wglMakeCurrent(Handle, hrc)) then
   begin
   CleanUp(handle);                   //Reset The Display
   MessageBox(0,'Cant''t activate the GL rendering context.'
                ,'Error',MB_OK or MB_ICONEXCLAMATION);
    Halt(1); { Halt right here! }
   end;

end;

end.
```