مقدمه ای بر WPF

مقدمه

هنگامی که .NET برای اولین بار پا به عرصه ظهور گذاشت، تکنولوژی های جدیدی را در زمینه برنامه نویسی معرفی نمود. به عنوان مثال می توان به موارد زیر اشاره کرد:

- یک روش کاملاً جدید برای ایجاد برنامه های تحت وب (ASP.NET).
- یک روش کاملاً جدید برای اتصال به پایگاه های داده (ADO.NET).
- زبان های مدیریت شده جدید (C# و VB.NET).
- مدیریت برنامه ها در زمان اجرا (CLR).

در میان این تکنولوژی ها فرم های ویندوز، که در واقع کتابخانه ای از کلاس های موجود برای ایجاد برنامه های ویندوزی می باشد، بیشتر مورد استفاده قرار می گیرد. هر چند فرم های ویندوز، ابزاری کامل و با تمام خصوصیات برای ایجاد برنامه های ویندوزی می باشند، اما وابستگی شدیدی به اساس کار ویندوز دارند که برای بیش از یک دهه تغییرات زیادی در آن اعمال نشده است.

بیشترین وابستگی فرم های ویندوز، مربوط به استفاده از APIهای ویندوز به منظور ایجاد یک نمای بصری برای برنامه های کاربر می باشد. در این مورد می توان به APIهایی که برای ایجاد دکمه، جعبه متن و ... استفاده می شود اشاره کرد. در نتیجه نمی توان در کنترل هایی که با استفاده از این روش ایجاد می شوند، تغییرات زیادی اعمال کرد (تا قبل از پیدایش WPF اکثر کاربران از همین روش برای ایجاد کنترل های استاندارد و مورد نظرشان استفاده می نمودند). به عنوان مثالی در این مورد، اگر بخواهید یک دکمه با متن درخشان و شیک ایجاد کنید، دیگر نمی توانید از توابع API که برای ایجاد کنترل Button در نظر گرفته شده اند استفاده کنید. برای این منظور باید ابتدا یک UserControl ایجاد کنید و سپس عملیات ترسیمی مربوط به جنبه های مختلف دکمه را با استفاده از مدل طراحی سطح پایین انجام دهید.

با تکیه بر مطالبی که در همین سطور بیان گردید، دیگر نمی توانید حتی خیال ایجاد جلوه های جالبی که در اکثر برنامه های گرافیکی استفاده می شود (مانند موج دار کردن دکمه ها، منقبض شدن فرم ها و...) در برنامه ها را به ذهنتان راه دهید. زیرا همانطور که اشاره گردید می بایست تمامی جنبه های این جلوه ها را به صورت دستی ترسیم کنید.

اساس نمایشی ویندوز (WPF) برای حل این معزلات یک ساختار کاری کاملاً جدید را معرفی کرده است. البته WPF از کلیه کنترل های استانداردی که تاکنون از آنها استفاده می نمودید، پشتیبانی می کند؛ اما برای ایجاد یک کنترل کلیه ترسیمات مربوط به متن، حاشیه و پس زمینه را خودش(WPF) انجام می دهد. با این تفاصیل، WPF می تواند با ارئه ویژگی های قدرتمندتری به برنامه نویس اجازه دگرگون کردن روشی که محتوای هر قسمت از صفحه نمایش رندر می شود را بدهد. با استفاده از این ویژگی ها می توانید سبک کنترل های عمومی موجود مانند دکمه را بدون نوشتن کد دوباره طراحی کنید. به شیوه مشابه ای می توانید با استفاده از اشیاء تبدیل، هر چیزی که بر روی واسط نمایشی برنامه قرار دارد را دستخوش تغییرات کنید. این تغییرات عبارتند از:

- چرخش
- کشیدن
- بزرگ نمایی
- انحراف (کج کردن)

نکته: البته برای انجام تغییرات ذکر شده درست قبل از ارائه به کاربر می توانید از سیستم پویانمایی bake-in که از ابزارهای WPF می باشد، استفاده کنید.

ساختار ویژگی های جدید در WPF، در واقع یک ساختار جدید و قدرتمند که براساس DirectX و APIهایِ گرافیکی سریعِ سخت افزاری، که معمولاً در اکثر بازی های کامپیوتری مدرن استفاده می شوند، پایه گذاری شده است. این بدان معناست که برنامه نویس می تواند از Effectهای گرافیکی جالب، بدون توجه سربار اجرایی که به دلیل استفاده از فرم های ویندوزی ممکن است به وجود آید، استفاده کند. در حقیقت می توانید از ویژگی های پیشرفته ای مانند پشتیبانی از فایل های ویدئویی و محتویات D۳ در فرم های ویندوزی استفاده کنید.

با به کارگیری این ویژگی ها(به همراه یک ابزار طراحی مناسب) امکان ایجاد واسط های کاربری تحریک کننده چشم و Effectهای بصری ویژه وجود خواهد داشت. البته باید مجدداً خاطرنشان کنیم که انجام چنین کار هایی با استفاده از فرم های ویندوزی تقریباً غیرممکن است.

با توجه به اینکه ویژگی های مربوط به پویانمایی، D۳ و فایل های ویدئویی بیشترین توجه در WPF را

معطوف خود می کند، باید توجه داشته باشید که با استفاده از WPF می توانید برنامه های ویندوزی معمولی را با استفاده از خصوصیات بصری ساده (قدیمی) نیز ایجاد کنید. به عبارت دیگر استفاده از کنترل های معمولی تا هنگامی که WPF در فرم های ویندوزی کار می کند به سادگی امکان پذیر است. حتی می توان این طور بیان کرد که WPF ویژگی هایی که مستقیماً به توسعه دهندگان تجاری می شود را تسهیل بخشیده است. این ویژگی ها به طور گسترده ای شامل مدل انقیاد داده ها، یک مجموعه جدید از کلاس ها برای چاپ محتویات و مدیریت صف های چاپ و یک مشخصه متنی برای نشان دادن متن های بزرگ قالبدار می شوند.

همچنین یک روش جدید برای ایجاد برنامه های بر مبنای صفحه که به صورت پیوسته در IE اجرا می شوند و می توان از طریق یک وب سایت به آن دسترسی پیدا کرد و آن را اجرا نمود نیز در نظر گرفته شده است. در این مدل، دیگر اخطارهای امنیتی متداول و تاییدیه های رنجاننده مربوط به نصب را مشاهده نخواهید کرد.

روی هم رفته WPF، بهترین روش های قدیمی از دنیای توسعه برنامه های ویندوز را با ابداعات جدیدی که برای ایجاد واسط های کاربری مدرن و گرافیکی توانمند را با هم ترکیب کرده است. با توجه به اینکه برنامه های ویندوزی به عمر چندین ساله خود ادامه می دهند، توسعه دهندگانی که به یک پروژه جدید ویندوزی می پردازند بهتر است از WPF استفاده کنند.

WPF یک چارچوب نمایشی کاملاً جدید است که توانایی های بسیاری از چارچوب های قبلی را مانند GDI+، GDI، User و HTML با یکدیگر ترکیب کرده است و توانایی تاثیر گزاری بر روی ابزارهای ایجاد پویانمایی در وب مانند Flash را نیز دارد. البته بر روی برنامه های ویندوزی مانند Microsoft Word نیز تاثیرگذار است.

معرفی WPF

اساس نمایشی ویندوز یک سیستم نمایش گرافیکی جدید برای سیستم عامل ویندوز می باشد. این تکنولوژی برای NET. طراحی شده است و همچنین تاثیر زیادی بر تکنولوژی های نمایشی جدیدی مانند HTML و Flash داشته و باعث بهینه سازی سرعت سخت افزار می گردد. WPF شامل تغییرات بنیادینی

در زمینه واسط گرافیکی ویندوز از زمان Windows 95 می باشد.

در این مقاله سعی بر این داریم که ساختار این تکنولوژی را مورد بررسی قرار دهیم. ابتدا به نحوه کارکرد آن نگاهی خواهیم داشت و در نهایت مواردی که این تکنولوژی برای نسل های بعدی برنامه های ویندوز متعهد شده است را مورد بررسی قرار می دهیم.

درک گرافیک ویندوز

درک مزایای مهیج و زیبای WPF بدون پی بردن به این **نکته** که چگونه برنامه نویسان ویندوز از تکنولوژی های نمایشی مشابه و یکسان برای بیش از ۱۰ سال استفاده می کردند واقعاً سخت است.

یک برنامه ویندوزی جدید که با زبان C# و یا VB.NET نوشته می شود، به صورت غیر مستقیم به دو بخش اصلی سیستم عامل ویندوز برای ایجاد واسط گرافیکی اش وابسته است:

• User32: این قسمت عهده دار فراهم آوردن نمایی شبیه ویندوز برای برنامه است و با عناصری مانند فرم ها، دکمه ها، جعبه های متن و ... سر و کار دارد.

• GDI/GDI+: این قسمت پشتیبان ترسیمی برای ایجاد اشکال، متن ها و تصاویر در قبال هزینهٔ پیچیدگی های اضافی در برنامه می باشد (البته معمولا عملکرد آن بی زرق و برق است).

هر دو تکنولوژی در سال های متمادی مورد بازبینی های متعدد قرار گرفتند و APIهایی که برای تعامل برنامه نویسان با آنها در نظر گرفته شده بودند نیز تغییرات چشمگیری داشته اند. اما هنگامی که برنامه نویسان قصد داشتند در برنامه ای (چه با C# و NET 2.0 و یا زبان های قدیمی تر مانند VB6 و حتی کدهای برپایه MFC C++) اقدام به ایجاد تصاویر دستی کنند قسمتهای یکسانی از سیستم عامل ویندوز در پشت پرده در حال کار بودند. Frameworkهای جدیدتر روش های بهتری برای تعامل با User32 و همچنین GDI/GDI+ ارائه کردند. این روش ها بهبودهای زیادی در کارآیی برنامه، کاهش پیچیدگی برنامه به همراه داشتند. البته این روش ها ویژگی هایی نیز اضافه می کردند که با استفاده از آنها برنامه نویس نیازی به کدنویسی زیاد نداشت. اما این روش ها نیز نتوانستند محدودیت های اساسیِ اجزای اصلی سیستم که بیش از یک دهه از طراحی آن می گذشت را حذف کنند.

نکته: اساسی ترین تقسیم وظایف میان User32 و GDI/GDI+ حدود ۱۵ سال پیش ارائه شده بود و برای

Windows 3.0 دارای ساختاری مناسب بودند. البته در آن زمان User32 در واقع با نام User معرفی گردید زیرا هنوز سیستم های ۳۲ بیتی ایجاد نشده بودند.

DirectX: موتور گرافیکی جدید

مایکروسافت برای رهایی از محدودیت های موجود در کتابخانه های User32 و GDI/GDI+ تکنولوژی جدیدی را مورد استفاده قرار داد. این تکنولوژی در واقع همان DirectX می باشد. DirectX به عنوان یک ابزار مستعد خطا و متحد برای ایجاد بازی های کامپیوتری بر روی پلتفرم ویندوز کار خود را آغاز کرد.

هدف اصلی در طراحی DirectX افزایش سرعت بود؛ از اینرو مایکروسافت به تعامل نزدیکی با سازندگان کارت های ویدئویی به منظور فراهم کردن شتاب سخت افزاری لازم که برای الگوهای گرافیکی پیچیده لازم بودند و همچنین effectهای ویژه ای مانند شفافیت و گرافیک سه بعدی پرداخت.

این تکنولوژی در طول سال های بعد از اولین انتشارش (مدتی بعد از Windows 95) رشد های قابل ملاحظه ای داشته است و هم اکنون یکی از اجزای لاینفک در سیستم عامل ویندوز می باشد که از تمامی کارت های ویدئویی پشتیبانی می کند. اگرچه APIهای برنامه نویسیِ DirectX هنوز هم اساس کار آن را به عنوان ابزاری برای توسعه بازی معرفی می کنند، اما به علت پیچیدگی زیاد کار با آنها، از DirectX تقریباً در هیچ یک از برنامه های ویندوزی (مانند برنامه های تجاری)استفاده نمی شود.

WPF تمامی این مشکلات را برطرف کرده است. در WPF از تکنولوژی های GDI/GDI+ به عنوان پایه کار استفاده نمی شود. در عوض از DirectX استفاده می شود. در برنامه های WPF بدون توجه به اینکه برنامه نویس چه واسط کاربری را ایجاد می کند از DirectX استفاده می شود. این بدان معناست که اگر شما بخواهید یک طرح گرافیکی پیچیده و سه بعدی ایجاد کنید و یا یک دکمه به همراه متنی آشکار ایجاد کنید، تمامی کار های ترسیمی از طریق DirectX انجام می شود. بالنتیجه، می توانید در تمامی برنامه های تجاری می توانید به سادگی از effectهای غنی را بدون هیچ مشکلی به کار برید. همچنین می توانید از مزایای شتاب دهنده سخت افزاری نیز بهره مند شوید. این بدان معناست که DirectX به کارهایِ زیادی که به GPU مرتبط است دخالت نمی کند و به انجام کارهای دیگری می پردازد.

نکته: GPU یک پردازشگر مجزا است که بر روی کارت های ویدئویی نصب می شود. مخفف Graphic

Processor Unit

نکته: DirectX نسبت به GDI/GDI+ کارآیی بهتری دارد؛ زیرا درک دقیقی از عوامل سطح بالا مانند گرادیان ها (شیب) و همچنین الگوها را که می توانند مستقیماً توسط کارت ویدئویی انجام شوند را دارد اما GDI/GDI+ از این مزیت بی بهره اند. از اینرو برای انجام چنین کار هایی توسط GDI/GDI+ باید دستورات مربوطه را به دستورات پیکسل به پیکسل تبدیل کرده و سپس اجرا کنید. بدیهی است که این عمل مدت زمانی طولانی تری را به خود اختصاص می دهد و بسیار کند عمل می کند.

تنها عنصری که هنوز در صحنه حاضر است User32 (با کمی محدودیت) می باشد. این امر بدان دلیل است که WPF هنوز از User32 برای انجام سرویس های خاصی مانند اجرای دریافت ورودی ها ودسته بندی اینکه هر برنامه دربردارنده چه قسمتی از صفحه نمایش می باشد. با این حال کلیه عملیات ترسیمی توسط DirectX انجام می پذیرد.

نکته: باید توجه داشته باشید که WPF یک روپوش جدید برای GDI/GDI+ نیست؛ بلکه یک جایگزین برای آنها می باشد. (یک لایه جدید که با DirectX کار می کند)

نکته: در پشتیبانی نرم افزاری WPF یک حالت استثناء وجود دارد. به علت پشتیبانی ضعیف راه اندازها، WPF فقط هنگامی عمل خوشنماسازی را برای ترسیمات D۳ انجام می دهد که برنامه بر روی ویندوز Vista اجرا گردد (همچنین باید یک راه انداز محلی در ویندوز ویستا برای کارت ویدئویی نیز داشته باشید). این بدان معناست که اگر تصویر سه بعدی را بر روی سیستم عامل XP Windows رسم کنید، احتمالاً با لبه هایی دندانه دار در انتها تصویر مواجه خواهید شد. اما اگر همان تصویر را در Windows Vista اجرا کنید لبه های تصویر روان و سلیس خواهند بود. خوشنماسازی برای ترسیمات D۲ بدون در نظر گرفتن سیستم عامل پیاده سازی شده است.

نکته: با داشتن یک کارت ویدئویی قدرتمند، تضمینی برای داشتن سرعت و کیفیت در WPF وجود ندارد. نرم افزار ها در این مورد نیز نقش مهمی را ایفا می کنند.

نکته: هدف اصلی WPF پردازش کارهای گرافیکی توسط GPU برای بهبود کیفیت نمایش تصاویر می باشد. به جای استفاده از CPU اصلی کامپیوتر برای پردازش روتین های گرافیکی پیچیده از GPU که بر روی کارت گرافیکی به صورت مجزا وجود دارد استفاده می شود. با این کار CPU برای انجام فعالیت های

دیگر آزاد باقی می ماند و از کارت ویدئویی استفاده مناسب و حداکثر صورت خواهد گرفت.

درجه بندی های WPF

کارت های ویدئویی بسیار با یکدیگر متفاوت می باشند. هنگامی که WPF یک کارت ویدئویی را مورد دسترسی قرار می دهد، فاکتورهای مهمی را مد نظر قرار می دهد:

- میزان حافظه RAM در آن کارت.
- پشتیبانی از پیسکل های سایه دار.
- پشتیبانی از vertexهای سایه دار.
- ....

بر اساس این جزئیات به هر کارت گرافیکی یک مقدار برای درجه اجرایی داده می شود. مقادیر انتسابی به سه دسته زیر تقسیم می شوند:

- Rendering Tire 0: کارت های ویدئویی با این درجه، از هیچگونه شتاب (بهینگی) سخت افزاری پشتیبانی نمی کنند. این درجه معادل استفاده از نسخه های DirectX 7.0 به پایین می باشد.
- Rendering Tire 1: کارت های ویدئویی این دسته قسمتی از خصوصیات شتاب سخت افزاری را پشتیبانی می کنند. این درجه معادل استفاده از DirectX 7.0 تا DirectX 9.0 می باشد.
- Rendering Tire 2: کارت های این قسمت، از تمامی ویژگی های شتاب سخت افزاری حمایت می کنند. این دسته معادل استفاده از DirectX ۹.۰ به بالا می باشد.

ممکن است بخواهید درجه ای که برای کارت گرافیکی شما در نظر گرفته شده است را مشاهده کنید. برای این کار می بایست مقدار خصوصیت Tier که در کلاس System.Windows.Media.RenderCapability وجود دارد را بازیابی کنید. برای تطبیق این مقدار با درجه بندی های فوق می بایست مقدار خصوصیت Tier را ١٦ بیت به سمت راست شیفت دهید. در زیر نمونه این کد آورده شده است:

```
int renderingTier = (RenderCapability.Tier >> ١٦);
```

نکته: برای استفاده از کلاس های موجود در فضانام فوق می بایست اسمبلی PresentationCore را به Referenceهای پروژه خود اضافه کنید.

WPF: یک API با سطح بالاتر

اگر شتاب سخت افزاری با استفاده از DirectX را به عنوان تنها پیشنهاد WPF در نظر بگیریم، آنگاه این تکنولوژی چیزی جز یک بهبود در عمل کامپایل نخواهد بود و نمی توان آن را به عنوان یک حرکت انقلابی در زمینه کارهای ترسیماتی به حساب آورد. WPF برای ایجاد یک حرکت دگرگون ساز سرویس های سطح بالایی را برای برنامه نویسان فراهم آورده است. این سرویس ها عبارتند از:

یک مدل آرایشی تارمانند: WPF به جای ارائه کنترل هایی ثابت و با گوشه های مشخص، بر روی یک صفحه آرایی انعطاف پذیر که در آن می توان کنترل ها را بر اساس محتوایشان تنظیم کرد، تاکید فراوان دارد. نتیجه این تاکید ایجاد یک واسط کاربری که می توان آن را برای نمایش محتویات پویا سازگار کرد می باشد.

یک مدل ترسیماتی غنی: در WPF به جای رسم کردن پیکسل ها با چند مورد از اشکال پایه ای و ابتدایی، بلوک های متنی و عناصر گرافیکی دیگر سروکار خواهیم داشت. البته در میان مزایای فوق نباید پشتیبانی ذاتی از ترسیمات D۳ را از یاد برد.

نکته: پشتیبانی از ترسمیات از ترسمیات D-۳ در WPF قدرت برابری با OpenGL و یا Direct3D را ندارد. اگر قصد دارید که برنامه ای که ترسیمات سه بعدی سنگینی را در بردارد پیاده سازی کنید (مانند بازی های امروزی) WPF 1.0 احتمالاً جوابگوی کار شما نخواهد بود.

یک مدل متنی غنی: سال ها پس از نشان دادن متن ها در کنترل های ضعیفی مانند Labelها، WPF برای برنامه های ویندوز قابلیت نشان دادن متن ها با سلیقه های مختلف در هر نقطه از واسط کاربری را به ارمغان آورده است.

پویانمایی به عنوان بهترین مفهوم برنامه نویسی: تاکنون می تواستید با استفاده از زمان سنج فرم برنامه را مجبور به دوباره سازی خودش کنید. اما در WPF، پویانمایی یک جز طبیعی و ذاتی در Framework می باشد. برای این منظور ابتدا انیمیشن ها را با استفاده از تگ های اعلانی تعیین می کنیم و سپس WPF آنها را در دستور کار قرار می دهد.

پشتیبانی از رسانه های صوتی و تصویری: ابزاری های ایجاد واسط کاربری که قبلاً مورد استفاده قرار می

گرفتند مانند فرم های ویندوزی برای کار با رسانه های صوتی و تصویری به صورت چشمگیری محدود بدوند. اما WPF این مشکل را برای برنامه نویسان رفع کرده است.

سبک ها و قالب ها: با استفاده از قالب ها و سبک ها می توانید نحوه رندر شدن عناصر برنامه و ... را به صورت دلخواه تغییر دهید.

دستورات: برای بسیاری از کاربران انجام یک کار چه از طریق منوها با نوار ابزار فرقی ندارد؛ زیرا نتیجه هر دو یکسان است. به همین منظور با استفاده از WPF می توان دستورات را در جایی قرار داد و به آنها اشاره کرد.

واسط کاربری اعلانی: هر چند برنامه نویسان می توانند پنجره های WPF را با استفاده کدنویسی ایجاد کنند، اما VS.NET از روش دیگری استفاده می کند. VS.NET هر عنصر در پنجره را به صورت تگ های XML در XAML ایجاد می کند. مزیت این کار جداسازی کدهای اصلی برنامه از کدهای مربوط به واسط کاربری می باشد. همچنین update کردن واسط با استفاده از ابزار هایی که برای این منظور در نظر گرفته شده است بسیار راحت تر خواهد بود.

برنامه های براساس صفحه: با استفاده از WPF، می توانید برنامه هایی مانند مرورگر ها ایجاد کنید. این برنامه ها به شما اجازه می دهند در مجموعه صفحات حرکت کنید، اطلاعات را کامل کنید و با استفاده از دکمه ها forward/backward به صفحه های بعد و قبل حرکت کنید.