The Pegasus Imaging Dot Net Philosophy

This document describes how Pegasus developed .NET managed components using existing Win32 object code and thus maintained the performance of the corresponding COM/ActiveX/VCL products while gaining nearly all the benefits of the .NET environment.

Pegasus Imaging Corporation provides imaging software development tools to application developers. The Pegasus product line is noted for its high-speed operations with functionality ranging from image compression and viewing to scanning, barcoding and ICR/OCR. Execution speed for each targeted delivery platform is optimized through algorithmic level and coding techniques specific to that platform. In the case of Wintel products, execution speed is optimized through significant use of assembler code utilizing the advanced MMX and SSE instructions. The significant speed advantage gained through the use of these approaches is a major differentiator over competitive products.

Microsoft's .NET initiative focuses on building, deploying and executing Web Services and applications. All applications that are built for the Microsoft .NET framework require the common language runtime (CLR) in order to run. Applications that target this framework are compiled into Microsoft Intermediate Language (MSIL) code. At runtime MSIL is just-in-time compiled into machine code and run against the CLR. Code that is executed in this manner is referred to as managed code.

Microsoft's .NET Framework thus provides a highly productive multi-language environment with the agility to solve the challenges of deployment and operation of internet-scale applications. Unfortunately an adverse effect is that execution speed of managed code running against the CLR significantly suffers when compared with execution speed of hand-optimized assembler code utilizing MMX and SSE extensions on the Wintel platform.

Pegasus customers (imaging application developers) will not be successful marketing applications that have substandard imaging performance. Yet, they need to deploy competitive .NET applications with imaging capabilities. Pegasus has taken on and solved this technical issue with extremely innovative techniques providing customers with the best of both worlds.

All of the Pegasus component-level products are available as .NET Windows Form Controls. The Pegasus .NET components appear as .NET managed controls in the Visual Studio IDE and support all .NET languages (tested in VB.NET, C#,

managed C++). They can be installed in the Global Assembly Cache or wherever the application's configuration file needs the code to be located. Current Pegasus customers will appreciate the ease of transition to these controls, implemented with a nearly identical interface. Most importantly, the Pegasus .NET components maintain all the execution speed of the corresponding COM/ActiveX/VCL components!

This begs the question: How did Pegasus create .NET imaging controls that can be deployed as managed components and still maintain native code speed and functionality? With the functionality and performance of the existing Pegasus components as the minimum requirement for a .NET version, we quickly understood that the existing code must be part of the solution. The existing code was in the form of COM objects. We developed a managed component that initializes the COM control and passes properties and methods between the managed interface and the COM layer – without requiring COM registration on the system. We then stored all code dependencies as compressed objects in the .NET component's resources and wrote a DLL loader that could access and decompress these code resources and execute the code. This allows the components to have a single file assembly for runtime deployment.

Of course, there were some trade-offs that had to be made in order to achieve these hand-optimized performance results without developing a fully managed and "safe" C# implementation of the components. The most notable from the application developers' point of view will be the means of handling exceptions within the Pegasus components. Each of the components had previously used similar methods of reporting errors that typically included an Error property queried by the application developer after each operation to validate success of that operation. In the .NET framework, exceptions are generated as typed exceptions. This capability will be available in a future version of the Pegasus components. The current version provides a PICException helper class to permit a consistent .NET coding style. The PICException class is used to simplify error handling.

.NET Remoting is allowed only for 100% managed code (with several other restraints on the operations the code actually performs) at the default security levels. Therefore remoting is possible using the PIC components only if the application can set lower security levels.

For a .NET solution NOW that provides a full feature set and the very highest-speed operation, download and test the fully featured trial version of the Pegasus .NET components. Pegasus is proud of this .NET release and we know that you will see the value and appreciate the performance of this approach.

Download our .NET components from our website at www.pegasusimaging.com. For more information, please contact the Pegasus Imaging Corporation Support Team at support@jpg.com.