



Infrastruktur für Programmierung

Für die Entwicklung von PDF-Anwendungen setzt PDFNet SDK einen C/C++-Compiler und Linker voraus. Dabei lässt sich unter Linux und Mac OS X die kostenfreien GNU Compiler Collection (GCC) (Versionen 3.x und 4.x) nutzen. Natürlich kann auch jede andere IDE (Interactive/Integrated Development Environment) z.B. Eclipse unter Linux oder XCode auf der Mac-Plattform eingesetzt werden. Einzige Voraussetzung ist, dass die IDE eine C/C++-Sprachversion unterstützt.

Unter Windows wird MS Visual Studio 2005 empfohlen; hier kann auf die kostenfreie Express Edition von Microsoft zurück gegriffen werden. Jedoch muss man dann einige Besonderheiten bzw. Einschränkungen der Express Edition kennen: Visual C++ beinhaltet nicht die Microsoft Foundation Classes (MFC), die man für Anwendungen mit grafischer Benutzeroberfläche (GUI) benötigt. Für die Programmierung reiner PDF-Anwendungen ohne GUI ist die MFC-Bibliothek allerdings nicht erforderlich.

Der Hersteller liefert das Produkt mit einer Vielzahl von Programmbeispielen aus, die man direkt in eine IDE importieren kann. Leider übernimmt Visual C++ Express im Unterschied zur Standard oder Professional Edition nicht alle vordefinierten Bibliotheksabhängigkeiten. Tipp: Die fehlenden Windows SDK-Libs richtet man recht einfach bei den Projekt-Eigenschaften im Bereich »Linker> Input>Additional Dependencies« ein (Bild 1).

Integrierte, abgestimmte Architektur

Die Bibliothek besitzt eine saubere Architektur, die in drei C++-Namespaces aufgeteilt ist:

- PDF: eine Menge von high-level Funktionen, um PDF-Elemente wie Lesezeichen, Grafiken, Formulare oder Seiten zu bearbeiten. Da der PDF-Namespace auf SDF aufbaut, kann man ausgehend von SDF-Datenstrukturen unmittelbar PDF-Funktionen nutzen.
- SDF: eine Menge von low-level Funktionen, um die interne PDF-Dateistruktur ansprechen zu können. Im Unterschied zum Adobe COS ist SDF tatsächlich objektorientiert aufgebaut, daher leichter verständlich und eleganter zu handhaben.
- Filter: dieser Namespace behandelt Komprimierung und Verschlüsselung, um kundenspezifische Belange abzubilden.

In der Regel reichen die Funktionen des PDF-Namensraums für die Implementierung aus. So kann man Seiten zwischen Dokumenten kopieren, grafische Elemente wie Bilder oder Text lesen/schreiben oder Formulare manipulieren. Sollte eine Anforderung nicht darüber umsetzbar sein, greift man auf den SDF-Namensraum zurück. Mittels SDF erhält man eine vollständige Kontrolle über jedes Detail der PDF-, FDF- oder PJTF-Spezifikation.

Einblick in den Funktionsumfang

Will man schnell einen Überblick zu den Einzellizenzen und deren Funktionalitäten gewinnen, so findet man auf der Website des Herstellers die Tabelle »PDFNet Feature Chart«. Diese Tabelle ordnet den sechs verschiedenen Lizenzmodellen deren Einzelfunktionen, gruppiert in dreizehn verschiedene Funktionsgruppen, zu. Dabei wird die vom Hersteller angestrebte umfassende PDF-Unterstützung für unterschiedliche Plattformen und Programmierumgebungen deutlich.

Tipp: Aus der Sammlung der Programmbeispiele sollte man sich die Applikation »PDFView« näher ansehen (Bild 2). Diese lässt sich direkt von der Homepage herunterladen, ohne die Bibliothek mit IDE installieren oder den Quellcode übersetzen zu müssen. Als eigenständige Anwendung zeigt sie nicht nur die Anzeige- und Editier-Funktionen von PDFNet auf, sondern veranschaulicht auch die Mächtigkeit der integrierten Algorithmen für Rasterizer, Antialiasing sowie Bildglättung.

Der Quellcode lässt sich unmittelbar mit eigenen Funktionen z.B. eigenhändigem Unterschreiben oder individuelle Stempel ausbauen. Achtung: Die Menüs unter »Document« sind lediglich Platzhalter – deren Implementierung findet man im Beispiel »PDFPageTest«. Auch weitere PDF-Elemente wie Formularfelder, interaktives U3D-Modell-Viewing oder Anmerkungen kann man entsprechend integrieren.

Zentrale Funktionen der Druckvorstufe

Die Fehlertoleranz der Bibliothek bei nicht vollständig PDF-konformen Dokumenten (Schriftproblemen, ungültigen Cross-Referenz-Tabellen) ist mit Adobe Reader/Acrobat identisch. Auch ein Farbmanagement (z.B. ICC) lässt sich realisieren, wobei man die von Adobe genutzten Standard-Farbprofile einsetzen kann. Allerdings fehlt bisher (mangels Vorgaben z.B. in der PDF-Spezifikation)



Bild 1 – Windows SDK-Libs nachträglich bekannt geben



eine automatische Konvertierung zwischen den Farbräumen.

Es gibt aber Funktionen, um Auflösungen oder dpi-Werte zu bestimmen und abzuändern. Für die Umsetzung von PDF-Standards (PDF/A, PDF/E, PDF/H, PDF/UA oder eine der PDF/X-Varianten) wird im Unterschied zu anderen Bibliotheken keine Runtime-Error-Exception geworfen. Dies hat den Vorteil, dass man als Entwickler nicht ständig eventuell aufgetretene Runtime-Ausnahmen abfangen muss. Vielmehr bekommt man als Entwickler mehr Freiheiten; man kann eigenständig prüfen, ob die Bedingungen eines Standards – aber auch davon abweichende – eingehalten werden.

Der Hersteller stellt für seine COS/SDF-Serialisierung sicher, dass ein PDF-Dokument immer Standardkonform sein kann. Er will für jeden PDF-Standard Unterstützung und Hilfe bei der Umsetzung anbieten: Kunden sollen in die Lage versetzt werden, jeden beliebigen PDF-Subset realisieren zu können. Angekündigt sind auch PDF-Verification und Preflight-Utilities, die PDFNet SDK-Kunden kostenlos erhalten sollen.

Web 2.0 und Formulare

Für zukünftige Web 2.0-Anwendungen spielt die Einbindung von PDF-Dokumenten eine wichtige Rolle. Zur Realisierung kann .NET, ActiveX oder AJAX eingesetzt werden. Bei den ersten beiden Software-techniken muss ein »Custom User Control« mittels der .NET- bzw. der

C++-Version entwickelt werden. Im AJAX-Falle ist es notwendig, den PDF-Inhalt als Bild (PNG oder JPEG) zu rendern. Dafür kann auf die PDFDraw-Klasse aus den Anwendungsbeispielen zurückgegriffen werden.

Die nächste Version der Bibliothek wird eine einfachere Verarbeitung von Daten in Formularfeldern unterstützen. Greift man auf dieses neue Feature bei der Entwicklung von PDF-Formularen zurück, so kann man dem Benutzer mehr Interaktivität (z.B. Speichern ausgefüllter Formulare) bieten. Als weiterer Vorteil ist die leichtere, schnellere Programmierung und eine bessere Wiederverwendung von Anwendungscode zu nennen.

Flexibles Drucken

Leider wird das eigentliche Ausdrucken nicht über die PDF-Spezifikation sondern über zusätzliche Druckstandards abgedeckt. Im Umfeld des Druckens hat man aber häufig besondere Wünsche, die in der Regel nicht mit dem PDF-Dokument alleine realisierbar sind. Daher kann durch reines PDF nicht gewährleistet werden, dass ein bestimmtes Dokument z.B. immer auf einem vorgegebenen Formular oder immer mit einem (nur auf dem Druck sichtbaren) Wasserzeichen ausgedruckt wird.

Derartige Anforderungen können mittels Programmierung auf vielfältige Weise umgesetzt werden. So kann man optionale Inhalte vor dem eigentlichen Druck einer Dateikopie hinzufügen oder Anmerkungen im PDF verwenden. Das SDK unterstützt unmittelbar diese Realisierungstechniken. Zudem können natürlich Methoden der verschiedenen APIs; z.B. .NET oder Win32 aufgerufen werden. Tipp: Ein Quellcode-Beispiel zum Druck findet man im Beispiel »PDFPrint«.

Performance und Optimierungen

Mit dem Produkt lassen sich verschiedene Strategien zur Verbesserung des Laufzeitverhaltens verfolgen. So kann man PDF-Dokumente für schnelle Web-Anzeige (mit linearisierten Objekten und Hinweis-Tabellen) erstellen. Tipp: Beim Mischen von Dokumenten sollte man anstatt auf Kopien direkt mit den Original-Dokumenten arbeiten. Soll ein Ausschnitt eines PDF-Dokuments kopiert werden, so greift man recht effizient auf ganze Seitenbereiche mit der ImportPages()-Funktion zu.

Oft stellen Text-Inhalte den eigentlichen Flaschenhals bei der Anzeige eines PDF-Dokuments dar. Da die Bibliothek zwei Rendering-Alternativen beinhaltet, kann man zwischen dem Build-In- oder dem GDI+-Rasterizer wählen. Setzt man letzteren ein, so verbessert sich häufig signifikant die Performance. Hinweis: Bei der .NET-Version handelt es sich um keinen COM-Wrapper, womit man auch in der .NET-Welt Leistungsverbesserungen erzielen kann.

Für performante Programme ist aber die Eigenschaft »thread safe« des PDFNet SDKs von entscheidender Bedeutung. Damit kann man verschiedene Rendering-Threads auf einer Mehrkernprozessor- oder Multi-CPU-Maschine gleichzeitig ausführen. Diese Parallelität verbessert wesentlich die Performance; insbesondere auch von Hintergrund-Jobs.

Dr. Manfred und Renate Simon

Bild 2 – Beispiel-Anwendung mit interaktiver grafischer Oberfläche

