Text-Rendering und Sprachunterstützung

Entwicklern von Nutzeroberflächen bietet die TFT-Technologie einen wesentlichen Vorteil: Mit ihr lassen sich Daten in einer beliebigen Sprache darstellen. Die sich daraus ergebende Vielfalt an Lokalisierungsmöglichkeiten bedeutet aber auch, dass für jeden regionalen Markt unterschiedliche Produktvarianten erstellt und gepflegt werden müssen.

Das starke Wachstum in der Unterhaltungselektronik und bei Smartphones hat die Massenproduktion von hochwertigen TFT-Displays vorangetrieben. Aufgrund der hohen Verbrauchernachfrage nach TFT-basierten Produkten bieten Erstausrüster in der Automobilindustrie zunehmend entsprechende TFT-basierte Lösungen auch im Mid-End-Segment von Fahrerinformations- und Infotainment-Systemen an, einem Bereich, den früher Segment- und Dot-Matrix-LCD-Displays beherrschten. Für OEMs eröffnen sich damit ganz neue Möglichkeiten, ihre Markenidentität flexibel über unterschiedliche Display-Themen und Sprachlokalisierung zu schärfen. Der Software-Support für solche mehrsprachigen Lösungen (Mehrsprachlösungen) in Mid-End-TFT-Applikationen stellt jedoch Herausforderungen an Produktentwickler und OEMs, unter anderem hinsichtlich Kostenoptimierung bei gleichzeitig bestmöglicher visueller Qualität, Verifizierungsprozessen und Updates bzw. Upgrades.

Mehrsprachlösungen

Mit der TFT-Technologie lassen sich Daten in einer beliebigen Sprache darstellen. Dies bedeutet aber auch, dass für jeden regionalen Markt unterschiedliche Produktvarianten erstellt und gepflegt werden müssen. Mithilfe globaler Entwicklungs- und Designstandards lassen sich Daten so in einer Display-Unit hinterlegen, dass das Modul im Sinne des allgemeinen Globalisierungstrends in jedem beliebigen Markt eingesetzt werden kann (Bild 1). Unter Betrachtung der unterschiedlichen Faktoren ergibt sich folgende Gleichung:

*Mehrsprachigkeit = ((Materialkosten (Speicherplatz, CPU, GPU) + Entwicklungskosten (Software-Reflash)) vs. (Laufzeitleistung (Rendering) + visuelle Qualität (Antialiasing))) * visuelle Effekte (Skalierung, Perspektive, gebogener oder geschwungener Schriftverlauf)*

Aus Sicht des OEM stellen sich im Wesentlichen folgende Fragen:

- Welcher Font wird dem Entwicklerteam bereitgestellt?
- Handelt es sich um ein Bitmap- oder ein Vektorfont?

Alle Bilder: Visteon

- Liegt eine einfache (single) oder mehrfache (multiple) Fontdatei vor?
- Welcher Support ist zur Umsetzung komplexer Schriftarten wie Arabisch oder Hindi verfügbar?
- Welche Upgrade-Optionen gibt es, um zu einem späteren Zeitpunkt eine neue Sprache hinzuzufügen?

Aus Zulieferersicht ergeben sich folgende Fragestellungen:

- Wie wird die Fontinformation in der Bildschirmeinheit gespeichert (Raster oder Vektor)?
- Welche Font- und Layout-Engine soll genutzt werden (Open Source, kommerzielle oder eigenentwickelte Lösung)?
- Wie lässt sich die Speichernutzung optimieren, ohne dass es zu Qualitätseinbußen kommt?
- Welche Verifizierungsmechanismen gibt es, um die Sprachdaten im fertigen Produkt zu überprüfen?

Anbieter von Schriften analysieren und erstellen ihre Fonts für eine bestimmte TFT-Anwendung äußerst sorgfältig und präzise. Dabei werden unter anderem Faktoren wie dpi (dot per inch), TFT-Größe oder die Abmessungen der Punkte betrachtet. In Zusammenarbeit mit dem Schriftanbieter prüft das Grafikdesign-Studio die verschiedenen Fontgrößen und passt die Optik für eine optimale Lesbarkeit im Display an. Der Schriftanbieter kann für eine bestimmte TFT-Anwendung mehrere Schriften für verschiedene Sprachen vorschlagen. Die entsprechende Fontinformation kann entweder als vorgerenderte Bitmap-Bilder oder idealerweise in einem skalierbaren Vektorformat wie das True Type Format (TTF) zur Verfügung gestellt werden. Im Gegensatz zum Bitmap-Format erlaubt das skalierbare Format die einfache Ergänzung

Bild 1: Geschriebene Schrift in verschiedenen Sprachen ist hochkomplex.

Sprache	Vorgegebene Glyphen	Vorgegebener Unicode	Darstellung
Hindi	स त ा	0938, 094D, 0924, 093E	स्ता
Hindi	स त ा	0927,094D, 092F	स्ता
Arabic	स त ा	0645, 0635, 062F, 0631	स्ता

Bild 2: Beispiele für Sprachen, unterschiedliche Glyphen, der entsprechende Unicode und die tatsächliche Darstellung im Display.

einer neuen Schriftgröße oder die Feineinstellung bereits angelegter Größen. Auf den ersten Blick scheint es sinnvoll, nur eine Fontdatei zu nutzen. Kompliziert wird es allerdings dann, wenn sich Zeichencodebereiche wie beispielsweise traditionelle (Big5-Kodierung) und vereinfachte (GB1830-Kodierung) chinesische Schriftzeichen überschneiden und nicht in einer einzigen Fontdatei zusammengeführt werden können. Um diese Problematik zu lösen, sollten individuelle Fontdateien für jede Schriftart bzw. Kodierung oder auch Containerdateien wie True Type Collection (TTC) bereitgestellt werden.

Komplexen Schriften wie Arabisch, Asiatisch oder Indisch zu rendern ist nicht leicht, insbesondere bei Lösungen für Mid-End-Systeme, in denen Speicherplatz und andere Ressourcen knapp bemessen sind, ist es wichtig, von vornherein ihre Support-Möglichkeiten für komplexe Schriften zu verstehen. Einen Text für den globalen Einsatz im Display zu rendern, ist bei komplexen Schriften wie Arabisch, Asiatisch oder Indisch keine leichte Aufgabe. So lässt sich sicherstellen, dass das Display-Modul Informationen korrekt wiedergibt, wenn es an externe Geräte wie Smartphones, Navigations- oder Audioeinheiten angeschlossen ist (Bild 2).

Text-Layout-Engine

Ein weiterer wichtiger Faktor in der Darstellung von Schrift ist die Text-Layout-Engine. Für komplexe Schriften mit Ligatur-Unterstützung wie Arabisch oder Indisch muss die Layout-Engine geeignete Glyphen-Substitutionen verarbeiten und ausführen können. Die Layout-Engine sollte nicht nur mit Texten umgehen können, die wie Arabisch von rechts nach links verlaufen, sondern auch mit bidirektionalen (BiDi) Texten, die eine Mischung aus von rechts nach links und von links nach rechts laufenden Texten sind. Um mit bidirektionalen (BiDi) Texten umgehen zu können, sollte die Layout-Engine den BiDi-Algorithmus unterstützen, der durch den Unicode-Standard spezifiziert ist. (Bild 3).

Des Weiteren muss noch vor der Software-Entwicklung der in der Lösung verfügbare Support zur Überprüfung der Strings genauer betrachtet werden. Wichtig ist, dass die Lösung alle Strings korrekt im Display abbildet (Bild 4).

Die größte Herausforderung aus Sicht eines Entwicklers ist der Speicherbedarf der einzelnen Fontdateien. Die meisten asiatischen Sprachen verwenden mehrere tausend Zeichen. Bei Bitmap-Fonts verhält sich die Größe der Zeichen proportional zur TFT-Auflösung, d. h. bei zunehmender TFT-Größe werden die Zeichen größer. Vom QVGA-Format (Quarter Video Graphics Array) zum VGA-Format (Video Graphics Array) vergrößert sich der Bitmap-Speicher um das Vierfache, denn jede Bitmap muss für die Display-Auflösung auf den Faktor vier skaliert werden. Wenn eine Lösung viele verschiedene Größen unterstützen muss, ist Speicherplatz von mehreren Megabyte nötig.

Mögliche Lösungsoptionen

Bei der Unterstützung asiatischer Sprachen mit tausenden Zeichencodes hilft die Auswahl der richtigen Antialiasing-Option (Kantenglättung), den tatsächlich anfallenden Speicherplatzbedarf zu reduzieren. Bitmap-Fonts können in QVGA- oder WQVGA-Format (Wide Quarter Video Graphics Array) verwendet werden. Für größere Auflösungen wie im WVGA-Format (Wide Video Graphics Array) ist es allerdings vorteilhaft, Vektorfonts und Laufzeitrasterung zu nutzen, um Speicherplatz einzusparen.

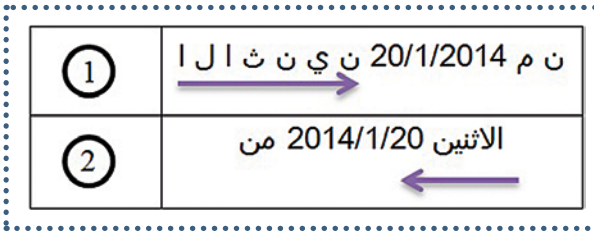

Bild 3: Das Beispiel zeigt (1) eine Eingangssequenz von Glyphen und (2) die Darstellung nach Anwendung von BiDi-Algorithmus und Glyphen-Substitution.

Vektorfonts ermöglichen zusätzliche visuelle Effekte (Skalierung, Prägung, Kontur, etc.) während der Laufzeit ohne Auswirkungen auf den Speicher. In einigen Lösungen lässt sich der gesamte asiatische Fontsatz mithilfe eines eigenen Speicherformates auf ein paar Kilobyte komprimieren.

In einem System mit knapp bemessenen Ressourcen können Laufzeitdekodierung von Fontdaten und Rasterung zu einer kostspieligen Angelegenheit werden. Das Zeichen-Rendering in einem 250-MIPS-Mid-End-System beispielsweise benötigt mit einer Vektor-Font-Engine und Laufzeitrasterung

i Visteon

Als einer der Marktführer bei Display-Lösungen für Fahrerinformations- und -Infotainmentsysteme bietet Visteon eigene Produkte für embedded Plattformen im Mid-End-Segment an und arbeitet dabei eng mit anderen Lösungsanbietern zusammen, um deren Technologien nahtlos zu integrieren. Visteon kombiniert eigenentwickelte bewährte Lösungen für Text-Layout- und Font-Rendering-Engines sowohl mit eigenen als auch mit Partnerlösungen. Zudem umfasst das Portfolio statische Analyseoptionen für Textübersetzungen, Speichersegmentierung für Updates oder Upgrades und globale Sprachunterstützung.

Sprache	String	Pixelbreite
English	Hello	34
Japanese	こんにちは	72

Bild 4: Dasselbe Wort in unterschiedlichen Sprachen ist unterschiedlich breit, auch wenn Fonttyp und Fontgröße identisch sind. Die Abbildung zeigt die Übersetzung des Strings „Hallo“ in unterschiedliche Sprachen sowie die jeweiligen Anforderungen an die Pixelbreite.

einige Millisekunden. Mit einem Bitmap-Font wären nur ein paar Mikrosekunden notwendig. Daraus können sich sichtbare Verzögerungen der Bildschirmaktualisierung ergeben, die sich negativ auf das Erscheinungsbild auswirken. Mit der Beschleunigung des Grafikprozessors (GPU) und/oder der Implementierung eines Caching-Schemas lässt sich diese Problematik beheben. Für eine effiziente Nutzung des Video-RAM ist es ratsam, einen mehrstufigen Cache zu implementieren, beispielsweise mit folgenden Stufen:

- Cache auf der ersten Stufe für Zeichenpfad-Information,
- Zweite Stufe für ein Rasterbild des Zeichens,
- Dritte Stufe für ein String-Rasterbild,
- Pre-Cache für häufig verwendete Zeichen, sodass es bei erstmaliger Aktivierung nicht zu Verzögerungen kommt.

Für das Text-Layout stehen Open-Source-Optionen wie Pango zur Verfügung, die allerdings zur Integration in eine vorgegebene Embedded-Anwendung gegebenenfalls angepasst werden müssen. Daher ist es oft ratsam, eine optimierte Text-Layout-Engine zu verwenden, die alle Anforderungen eingebetteter Systeme erfüllt. Somit lässt sich kostbare CPU-Zeit sparen.

Software Framework

Die meisten solcher Mid-End-Lösungen verfügen über keine interne Dateisystemunterstützung. Ob das System ein Update oder Upgrade unterstützt, ist abhängig vom Software-Framework. Eine Möglichkeit ist, die Grafikdaten in einzelne kleinere Segmente zu unterteilen und diese bestimmten Speichersegmenten zuzuweisen. Dadurch sind individuelle Updates in den jeweiligen Segmenten möglich, und neue Fonts oder Sprachen lassen sich problemlos zu einem späteren Zeitpunkt hinzufügen. ■ (oe)

» **www.visteon.com**

Manoj Edayilamuriyil ist Graphics Rendering Technical Professional bei der Visteon Corporation.