Core Text and Fonts

Features and techniques

Session 226

Ned Holbrook
Typographic Engineer

What You Will Learn

- New features in Mountain Lion and iOS 6
  - Line bounds
  - Baseline alignment
- Techniques
  - Vertical text
  - Font names, fallbacks, emoji
  - Avoiding common pitfalls

CoreText Overview

- Fundamental framework for Unicode text layout and fonts
- Available via:
  - AppKit (OS X)
  - Core Animation
  - UIKit (iOS)
  - WebKit

CoreText Overview

Availability

- Leopard and iOS 3.2
- Top-level framework on iOS and Mountain Lion

CoreText Overview

Deprecated OS X Frameworks

- ATSUI (text layout)
- ATS (font management)

New Features

Line Bounds

```
CTLineGetTypographicBounds(…)
```

Line Bounds

```
CTLineGetTypographicBounds(…)
```

Line Bounds

```
CTLineGetBoundsWithOptions(…, 0)
```

“Just the facts, ma’am.”

Line Bounds

```
CTLineGetBoundsWithOptions(…, kCTLineBoundsUseOpticalBounds)
```

- “Just the facts, ma’am.”

Line Bounds

```
CTLineGetBoundsWithOptions(…, kCTLineBoundsUseHangingPunctuation)
```

Line Bounds

```
CTLineGetBoundsWithOptions(…, kCTLineBoundsUseHangingPunctuation |
                              kCTLineBoundsUseOpticalBounds)
```

“Just the facts, ma’am.”

Line Bounds

```
CTLineGetBoundsWithOptions(…, kCTLineBoundsUseGlyphPathBounds)
```

Line Bounds

`kCTParagraphStyleSpecifierLineBoundsOptions`

- Affects line edges during frame filling

Baseline Alignment

- Better handling of mixed scripts

Hello 世界

Baseline Alignment

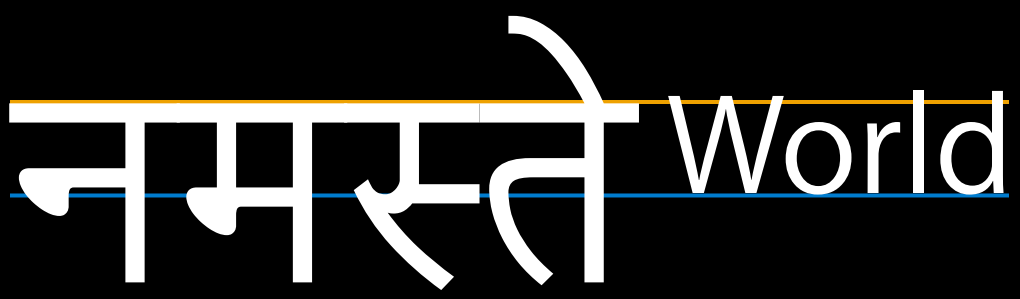
- Better handling of mixed scripts

Baseline Alignment

- Better handling of mixed scripts

Baseline Alignment

- Better handling of mixed scripts



Baseline Alignment

- Typographic effects

Drop Caps

Baseline Alignment

- Typographic effects

Baseline Alignment

Basics

- Each font has multiple baselines
- Each baseline is a vertical offset
- Each glyph has a default baseline
- Alignment uses the same baseline from two sets

Baseline Alignment

Basics, cont.

- Parameters determined by string attributes
- Reference established with `kCTBaselineReferenceInfoAttributeName`
- Baselines to be moved use `kCTBaselineInfoAttributeName`
- Baseline being aligned is `kCTBaselineClassAttributeName`

Baseline Alignment

Font Fallbacks

```
CTFontRef font;
NSMutableAttributedString *attrString;
NSRange range;

NSDictionary *referenceInfo = @{
    (id)kCTBaselineReferenceFont : (id)font
};
[attrString addAttribute:(id)kCTBaselineReferenceInfoAttributeName
                   value:referenceInfo range:range];
```

Baseline Alignment

Specifying a Baseline

```
CTFontRef font, hangingFont;
NSMutableAttributedString *attrString;
NSRange range, wordRange;

NSDictionary *referenceInfo = @{
    (id)kCTBaselineReferenceFont : (id)font
};
[attrString addAttribute:(id)kCTBaselineReferenceInfoAttributeName
                   value:referenceInfo range:range];
```

```
NSRange hangingRange = NSMakeRange(wordRange.location, 1);
[attrString addAttribute:(id)kCTBaselineClassAttributeName
                   value:(id)kCTBaselineClassHanging range:hangingRange];
[attrString addAttribute:(id)kCTFontAttributeName,
                   value:(id)hangingFont range:hangingRange];
```

Baseline Alignment

Specifying an Offset

```
CTFontRef font;
NSMutableAttributedString *attrString;
NSRange range;

NSDictionary *referenceInfo = @{
    (id)kCTBaselineReferenceFont : (id)font,
    (id)kCTBaselineClassHanging : @5.0
};
[attrString addAttribute:(id)kCTBaselineReferenceInfoAttributeName
                   value:referenceInfo range:range];
```

CTFontCollection

- Lion improvements
  - Mutable collections
  - Easier to access font family members
  - Toll-free bridged with NSFontCollection
- Preferred way to enumerate fonts

AAT and OpenType Features

- New AAT features correspond to common OT features
  - For example, to enable ‘pcap’ feature

```
CTFontRef font;

CTFontDescriptorRef descriptor = CTFontCopyFontDescriptor(font);
CTFontDescriptorRef pcapDescriptor =
    CTFontDescriptorCreateCopyWithFeature(descriptor,
        (CFNumberRef)[NSNumber numberWithInt:kLowerCaseType],
        (CFNumberRef)[NSNumber numberWithInt:
            kLowerCasePetiteCapsSelector]);

CTFontRef pcapFont = CTFontCreateWithFontDescriptor(pcapDescriptor,
                                                    0.0, NULL);
```

Bidirectional Text

`kCTWritingDirectionAttributeName`

- Embedding or override

“Advanced” Topics

Vertical Text

サンフランシスコ

Vertical Text

```
kCTVerticalFormsAttributeName = @YES
```

サンフランシスコ

Vertical Text

```
kCTVerticalFormsAttributeName = @YES
```

Vertical Text

- Rotate context

Vertical Text

サンフランシスコ

- Rotate context

Vertical Text

- Use NSTextView or CTFramesetter if possible
- Not using one of those?
  - Layout process is same as with horizontal text
  - Drawing is much, much different

Font Names

- Fonts have many names
  - Example:

PostScript	TimesNewRomanPS-ItalicMT
Family	Times New Roman
Full/Display	Times New Roman Italic
Style	Italic

Font Names

- Fonts have many names
  - Example:

PostScript	TimesNewRomanPS-ItalicMT
Family	Times New Roman
Full/Display	Times New Roman Italic
Style	Italic

Font Names

- Fonts have many names
  - Example:

PostScript	TimesNewRomanPS-ItalicMT
Family	Times New Roman
Full/Display	Times New Roman Italic
Style	Italic

Font Names

Performance

`CTFontCreateWithName()` is currently too lenient

- Will stop accepting non-PostScript names in future releases

- iOS 6 adds logging to detect use of wrong name:

```
June 13 08:50:59 YourDevice YourApp[237] <Notice>: CoreText performance
note: Client requested font with PostScript name "HelveticaNeue" using name
"Helvetica Neue" instead.

June 13 08:50:59 YourDevice YourApp[237] <Notice>: CoreText performance
note: Set a breakpoint on CTLogSuboptimalRequest to debug.
```

Font Names

Specifying a non-PostScript name

```
NSDictionary *attributes = @{
    (id)kCTFontFamilyNameAttribute : @"Times New Roman"
};
CTFontDescriptorRef descriptor = CTFontDescriptorCreateWithAttributes(
    (CFDictionaryRef)attributes);

NSArray *matches =
    (NSArray *)CTFontDescriptorCreateMatchingFontDescriptors(
        descriptor, NULL);
if ([matches count] != 0) {
    // at least one match found, first one would have been returned by
    //  CTFontDescriptorCreateMatchingFontDescriptor()
}
```

Fallback Fonts

- Layout relies on fallback fonts
- Augment system cascade using kCTFontCascadeListAttribute
- Terminate fallback processing with font covering all characters
  - “LastResort”
- Composite (CFR) fonts on Mountain Lion
  - /System/Library/DTDs/SplicedFont.dtd

Embedding Fonts

- Fonts need not live in distinct files to be usable
- On Mountain Lion or iOS

```
CGDataProviderRef provider = CGDataProviderCreateWithCFData(data);
CGFontRef cgFont = CGFontCreateWithDataProvider(provider);
CFErrorRef error = NULL;
if (CTFontManagerRegisterGraphicsFont(cgFont, &error)) {
    // access font via font names

    CTFontManagerUnregisterGraphicsFont(cgFont, NULL);
}
```

- On Lion or higher
  - `CTFontManagerCreateFontDescriptorFromData()`

Emoji

Emoji

Emoji 

- Unification is a big problem

I ♥ Unicode

≠

I  Unicode

Emoji 

- Unicode 6.1 solution—variation sequences

<U+2665>	???
<U+2665 U+FE0E>	♥
<U+2665 U+FE0F>	

Drawing Color Glyphs

- Automatic with CTFrameDraw(), CTLineDraw(), and CTRunDraw()
- Or use CTFontDrawGlyphs()
  - Analogous to CGContextShowGlyphsAtPositions()

Identifying Color Glyphs

```
CTFontRef font;
CGGlyph glyph;

CTFontSymbolicTraits symTraits = CTFontGetSymbolicTraits(font);
if ((symTraits & kCTFontTraitColorGlyphs) != 0) {
    // font has color glyphs

    CGPathRef path = CTFontCreatePathForGlyph(font, glyph, NULL);
    if (path == NULL) {
        // this is a color glyph
    }
    else {
        // not a color glyph
        CFRelease(path);
    }
}
```

CGContext Drawing Parameters

- CoreText only sets what it needs to
- Most string attributes correspond to a particular parameter
- Exception to the rule—kCTForegroundColorAttributeName
  - Use kCTForegroundColorFromContextAttributeName

Synthesizing Styles

- CoreText does not synthesize font styles (bold, italic)
- Clients may choose to do something for compatibility reasons
  - Italic: Skewed font matrix
  - Bold: No best approach

More Information

Jake Behrens
UI Frameworks Evangelist
behrens@apple.com

Documentation
Mac OS X Human Interface Guidelines
http://developer.apple.com/ue

Apple Developer Forums
http://devforums.apple.com

Related Sessions

Introduction to Attributed Strings for iOS	Mission Wednesday 3:15PM
Advanced Attributed Strings for iOS	Mission Thursday 10:15AM

Labs

Core Text Framework Lab	Essentials Lab B Thursday 11:30AM


WWDC2012

The last 3 slides after the logo are intentionally left blank for all presentations.

The last 3 slides after the logo are intentionally left blank for all presentations.

The last 3 slides after the logo are intentionally left blank for all presentations.