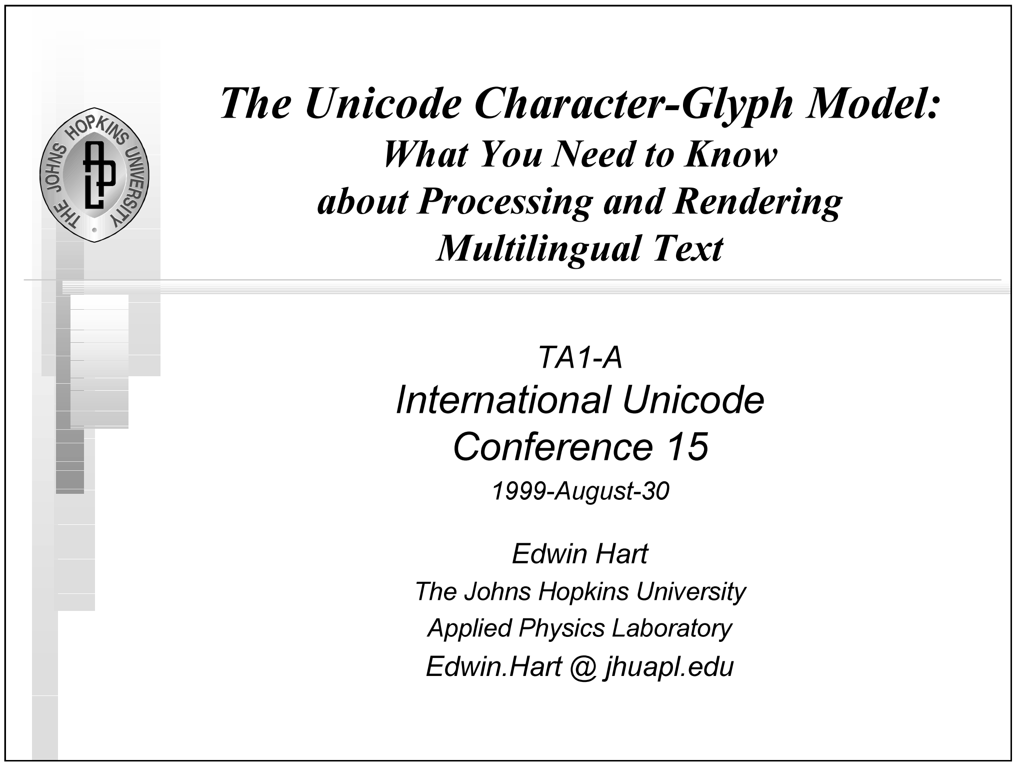


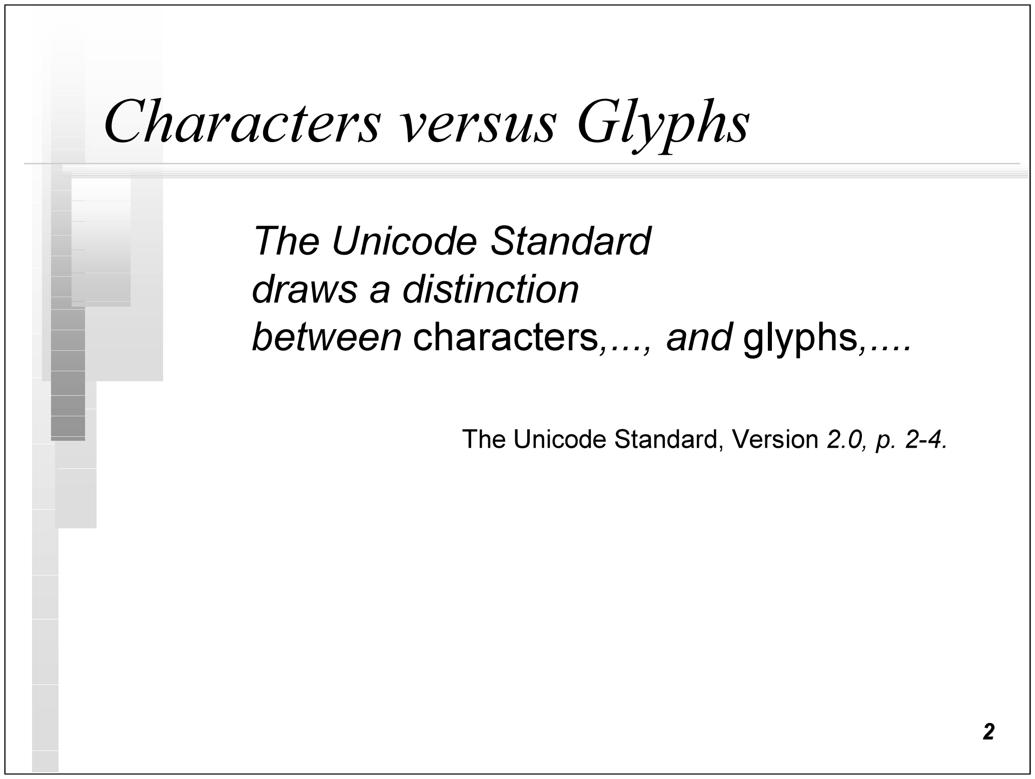
The Unicode Character-Glyph Model: What You Need to Know about Processing and Rendering Multilingual Text

TA1-A

International Unicode Conference 15

1999-August-30

Edwin Hart
The Johns Hopkins University
Applied Physics Laboratory
Edwin.Hart @ jhuapl.edu



Characters versus Glyphs

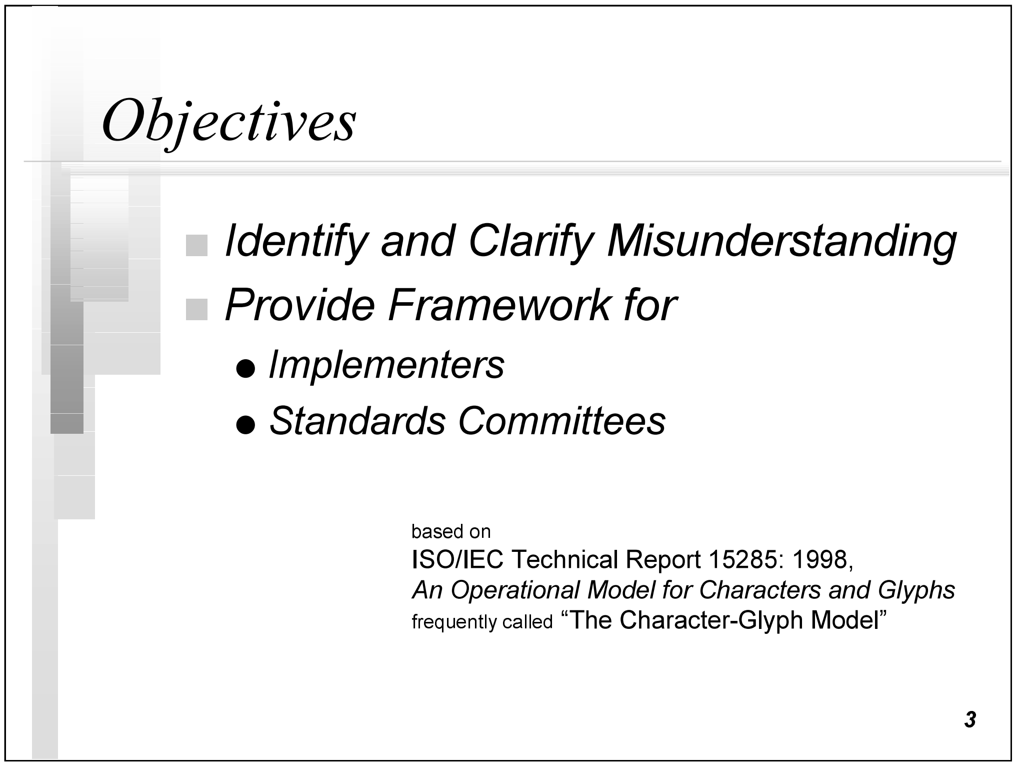
The Unicode Standard
draws a distinction
between characters*,...,* *and* glyphs*,....*

The Unicode Standard, Version *2.0, p. 2-4.*

2

- What does this mean?
- What are *characters*?
- What are *glyphs*?
- How are they related?
- How are they different?
- How are characters converted into glyphs?
- Why is this relevant to rendering multilingual text?

This presentation will answer these questions.



Objectives

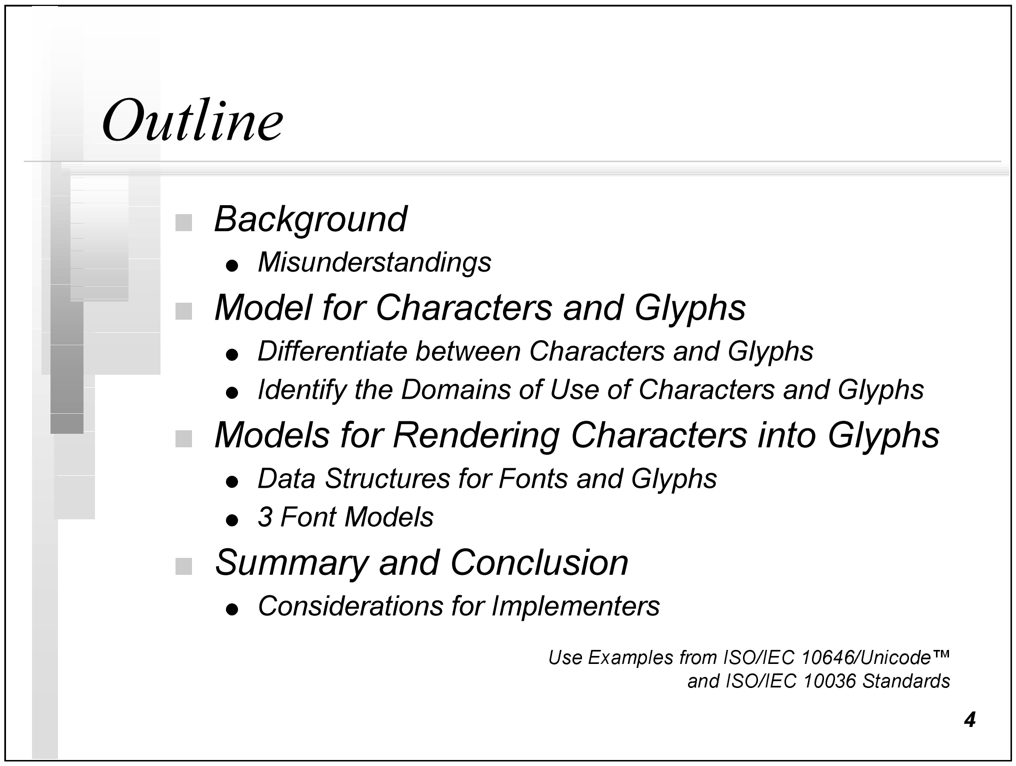
- *Identify and Clarify Misunderstanding*
- *Provide Framework for*
  - *Implementers*
  - *Standards Committees*

based on
ISO/IEC Technical Report 15285: 1998,
An Operational Model for Characters and Glyphs
frequently called "The Character-Glyph Model"

3

The reason for developing this presentation is that the concepts behind characters and glyphs were misunderstood not only by implementers, but also by people on the standards committees. Understanding these concepts is particularly important for rendering multilingual text for people to read. If you are developing multilingual software, you need to be aware of these concepts. By the conclusion of this session, you should have a much better understanding of characters, glyphs, and how to render characters into glyphs.

The information in this presentation is based on an ISO Technical Report, ISO/IEC TR 15285, *An operational model for characters and glyphs*.



Outline

- Background
  - Misunderstandings
- Model for Characters and Glyphs
  - Differentiate between Characters and Glyphs
  - Identify the Domains of Use of Characters and Glyphs
- Models for Rendering Characters into Glyphs
  - Data Structures for Fonts and Glyphs
  - 3 Font Models
- Summary and Conclusion
  - Considerations for Implementers

Use Examples from ISO/IEC 10646/Unicode™ and ISO/IEC 10036 Standards

4

This presentation has four parts:

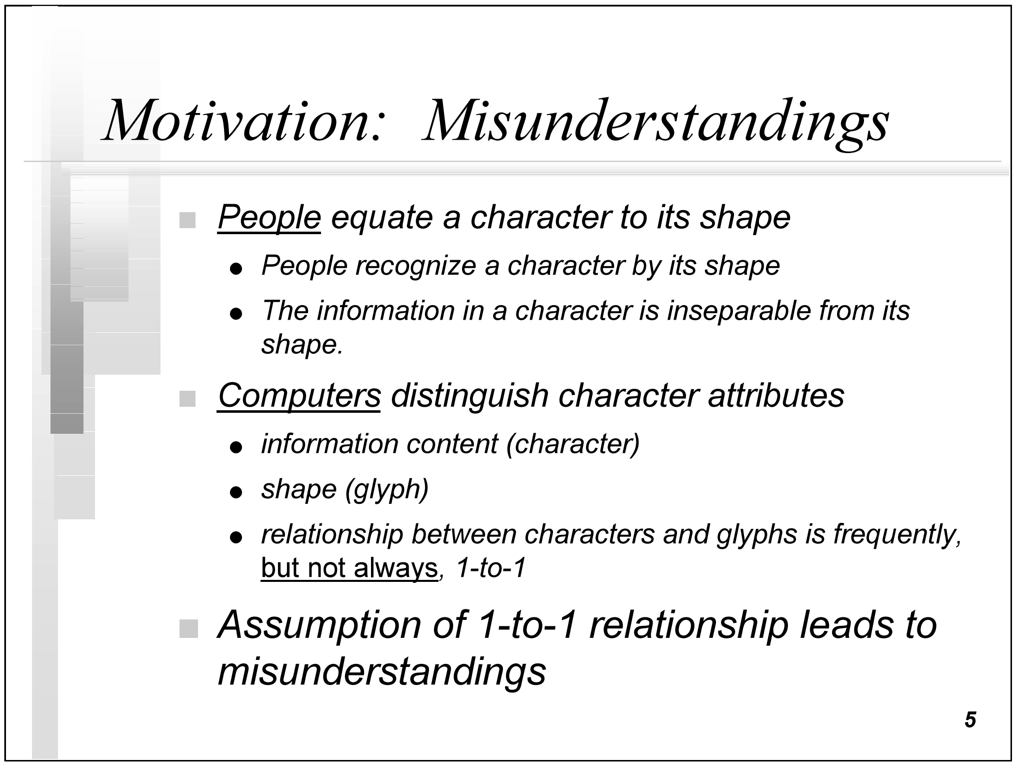
First, we describe the misunderstandings that motivated development of this session.

Second, we describe a model for characters and glyphs to differentiate between the two and identify where to use one versus the other. We also show several examples of the glyph-selection process of mapping characters into glyphs.

Third, we describe three similar models for rendering characters into glyphs. We describe the data structures required for this process and then three font models. We conclude this section with a comparison of the models and a recommendation of which ones to use for multilingual text processing.

Fourth, we conclude with a summary and a list of considerations for implementers.

This presentation uses examples from the Unicode implementation of ISO/IEC 10646-1: 1993 and the ISO/IEC 10036 standard.



Motivation: Misunderstandings

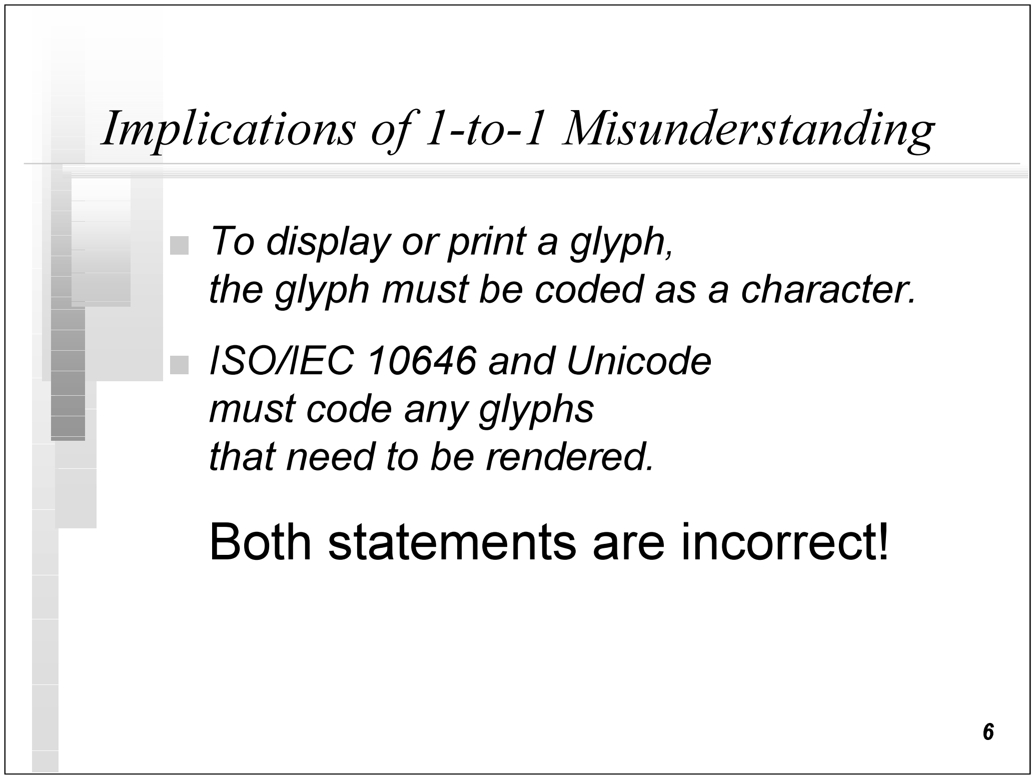
- *People equate a character to its shape*
  - *People recognize a character by its shape*
  - *The information in a character is inseparable from its shape.*
- *Computers distinguish character attributes*
  - *information content (character)*
  - *shape (glyph)*
  - *relationship between characters and glyphs is frequently, but not always, 1-to-1*
- *Assumption of 1-to-1 relationship leads to misunderstandings*

5

When you learned your letters you were taught to recognize a letter by its shape. As far as you were concerned, the letter equated to its shape and the two were inseparable.

Although this was fine for learning to read, in Information Technology we distinguish a character's attributes of information content versus its shape. We use the term *character,* to describe the information content attribute, and the term *glyph,* to describe the shape attribute. This distinction appears arbitrary because for most characters coded in Unicode, a one-to-one relationship exists between a character's information content and its shape. Notice that we said "for most characters" because, in multilingual text processing, the relationship is not always one-to-one. The incorrect assumption of a one-to-one relationship between the information content and shape attributes leads to misunderstandings.

One can liken this to a similar situation in Physics. For most of the world we perceive, Newtonian Physics works very well. However, in the late 1800s and early 1900s, Physicists identified situations where Newtonian Physics was inadequate. It took Einstein to discover Relativity theory that provided better answers. Returning to the domains of characters and glyphs, for the most part, a one-to-one is sufficient; however, in a number of situations, the relationship is more complex and a one-to-one relationship fails.



Implications of 1-to-1 Misunderstanding

- *To display or print a glyph,
  the glyph must be coded as a character.*
- *ISO/IEC 10646 and Unicode
  must code any glyphs
  that need to be rendered.*

Both statements are incorrect!

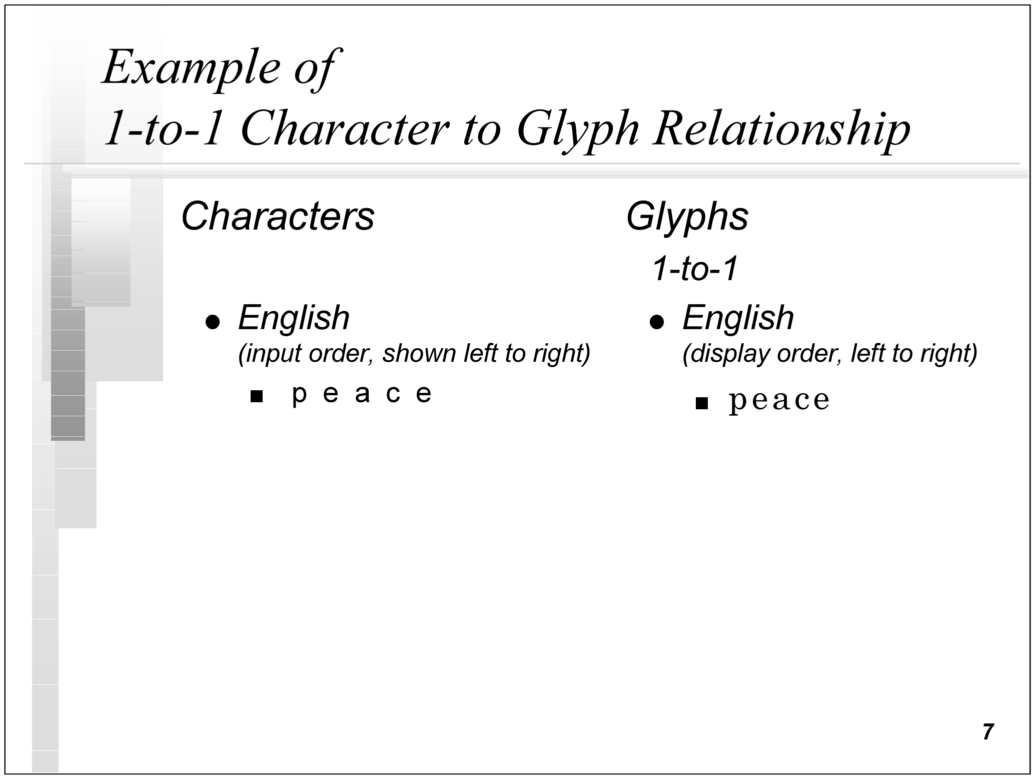
6

If you assume the 1-to-1 relationship as a requirement (and many people on the standards committee made this assumption), then you obtain two conclusions:

First, if you want to see a particular shaped character, then it must be coded as a character in a code.

Furthermore, if this is true, then ISO/IEC 10646 and Unicode must code the shapes than need to be rendered.

Let me assure you that both of these statements are false due to a false assumption.



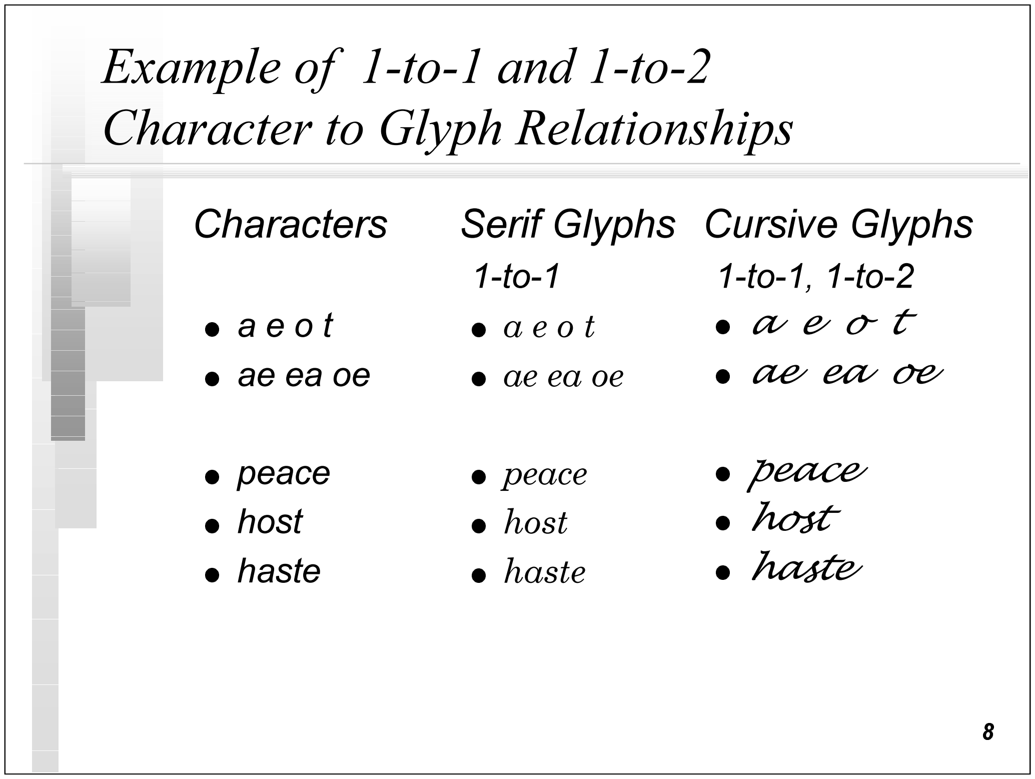
Example of 1-to-1 Character to Glyph Relationship

Characters	Glyphs
	1-to-1
• English (input order, shown left to right)	• English (display order, left to right)
▪ p e a c e	▪ peace

7

We will now look at three examples that illustrate increasing complexity in the rendering of characters into glyphs.

The first is a simple example in English. (The figure has characters on the left and glyphs on the right.) You type “p”, “e”, “a”, “c”, and then “e” to form the word “peace”. Text is stored in logical or input order. In this case, you see it in the natural left-to-right order of the English language. On the right, you see the word “peace” displayed, also in left-to-right order. Here we have a 1-to-1 mapping from the 5 English characters into 5 glyphs. This is all very straightforward.



Example of 1-to-1 and 1-to-2 Character to Glyph Relationships

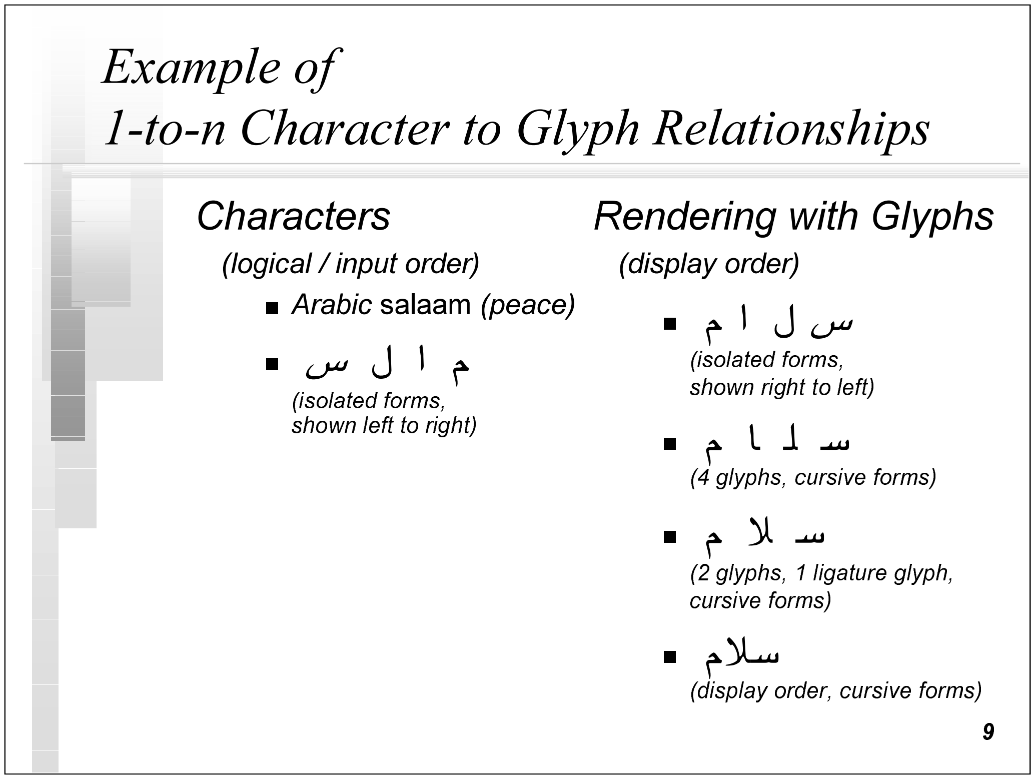
Characters	Serif Glyphs 1-to-1	Cursive Glyphs 1-to-1, 1-to-2
• a e o t	• a e o t	• a e o t
• ae ea oe	• ae ea oe	• ae ea oe
• peace	• peace	• peace
• host	• host	• host
• haste	• haste	• haste

8

Let's continue with a second English example.

The figure has three columns. The first column contains the characters. The second column show the glyphs in a Serif font (Century Schoolbook). Like the first example on the previous page, the glyphs in the second column have a 1-to-1 relationship to the characters in the first column. Nothing new is in the second column.

However, focus on the third column. Here we have glyphs from a cursive Script font (Lucida Handwriting). The cursive glyphs are connected in words to mimic cursive handwriting. Note that in this font, the lengths of the tails of the “a”, “e”, and “o” glyphs vary depending on whether the glyph is isolated or at the end of a word versus preceding another letter. For these letters, we see a 1-to-2 character to glyph mapping. Now examine the “t” glyph in the words “host” and “haste”. The “t” seems to have the same length tail regardless of what follows. For the “t”, we see a 1-to-1 character to glyph mapping. Therefore, selecting the correct glyph for this cursive font depends on the character itself and sometimes on the context of the character. Rather than being strictly 1-to-1, sometimes, the mapping is 1-to-2. Here we see some more complexity and we are still discussing rendering English text.



Example of 1-to-n Character to Glyph Relationships

Characters

(logical / input order)

- *Arabic* salaam *(peace)*
- ﺱ ﻝ ﺍ ﻡ
  (isolated forms, shown left to right)

Rendering with Glyphs

(display order)

- س ل ا م
  (isolated forms, shown right to left)
- ﺳ ﻠ ﺎ ﻡ
  (4 glyphs, cursive forms)
- ﺳ ﻼ ﻡ
  (2 glyphs, 1 ligature glyph, cursive forms)
- سلام
  (display order, cursive forms)

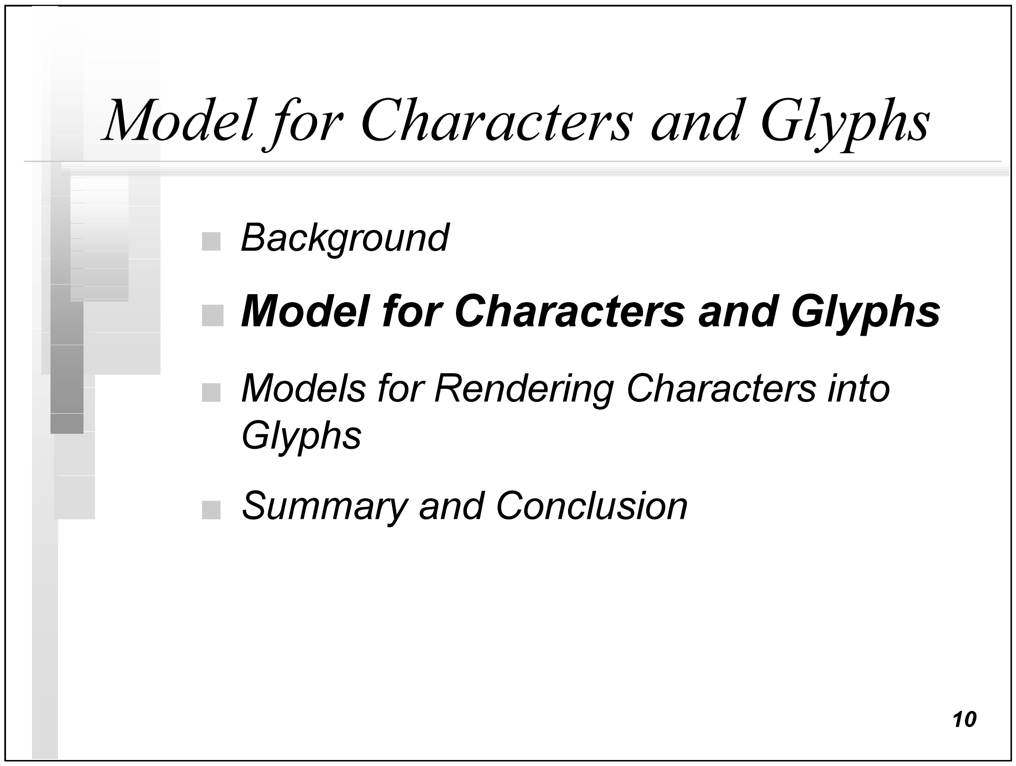
9

Now, let's turn to a more complex example. Once again, we have characters in the left column and glyphs in the right column. On the left are the letters for the Arabic word "salaam", which means "peace". Someone typing it, would type the Arabic letters "seen", "lam", "alef", and "meem". (The illustration has spaces between the glyphs so that you can more easily identify them.) The figure first shows the 4 letters in left-to-right *input* order.

Now look at the Glyph column on the right. Unlike English, Arabic letters are written from right to left. As part of the rendering process, we first order the letters in right-to-left *display* order. In addition, the Arabic script uses cursive (or joined) forms of the letters. Therefore, like the English example of the Script font, we select glyphs so that they properly connect together. However, each Arabic character can take one of up to 4 shapes depending on the context. The next bullet shows the 4 Arabic glyphs, in display order, with the proper contextual shape. However, we are not yet finished. Arabic typography also uses ligatures. Ligatures are single glyphs that represent multiple letters. In Arabic, the "lam-alef ligature" is mandatory. So we need to replace the "lam" glyph followed by the "alef" glyph with the "lam-alef ligature glyph". The next bullet shows 3 glyphs: the "seen, initial form", the mandatory "lam-alef ligature, final form", and the "meem, isolated form" glyphs. The final bullet shows a proper Arabic rendering of "salaam" without spaces.

These examples should illustrate that rendering characters into glyphs can be a complex process.

Kamal Mansour from Monotype kindly provided this Arabic example. On Tuesday, Mr. Mansour will be speaking on *Unicode and Fonts: An Overview of New Scripts in Unicode 3.0*. Following him, Thomas Milo will be discussing the complexities of rendering Arabic in *Authentic Arabic*.

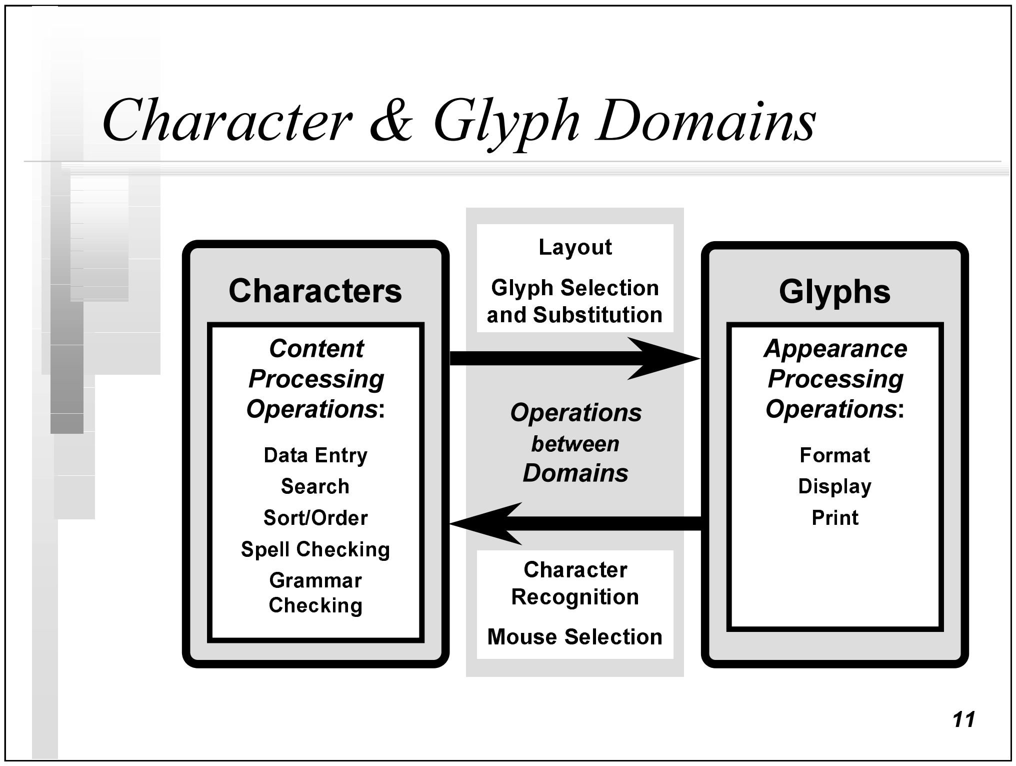


Model for Characters and Glyphs

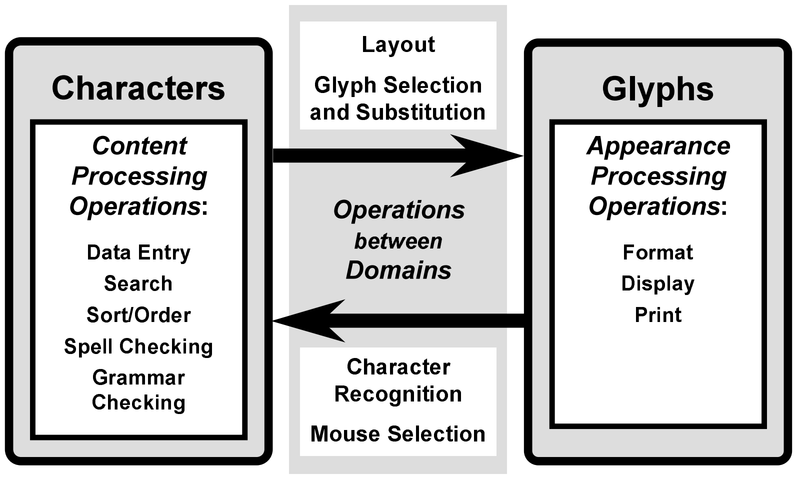
- Background
- **Model for Characters and Glyphs**
- Models for Rendering Characters into Glyphs
- Summary and Conclusion

10

Having described the misunderstandings about characters and glyphs, and given examples to illustrate some of the complexity of the rendering process, let's continue by looking at a model for describing characters and glyphs.



Character & Glyph Domains



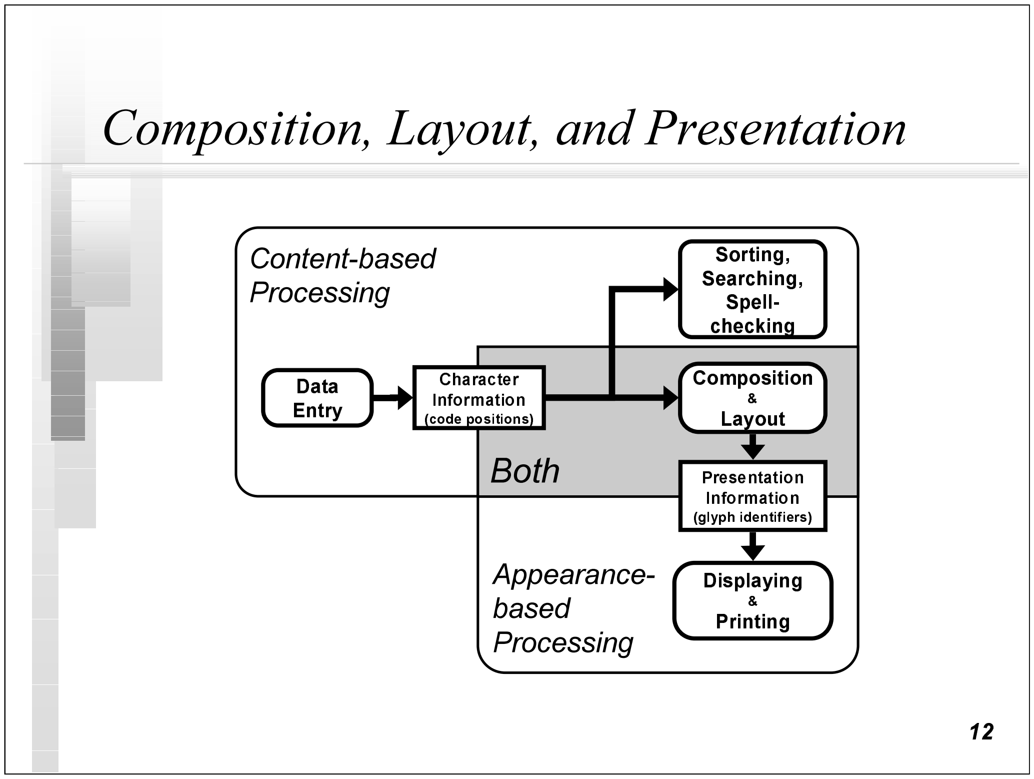
11

This figure shows two separate domains, one for characters on the left and another for glyphs on the right. It also show operations for going from one domain to the other.

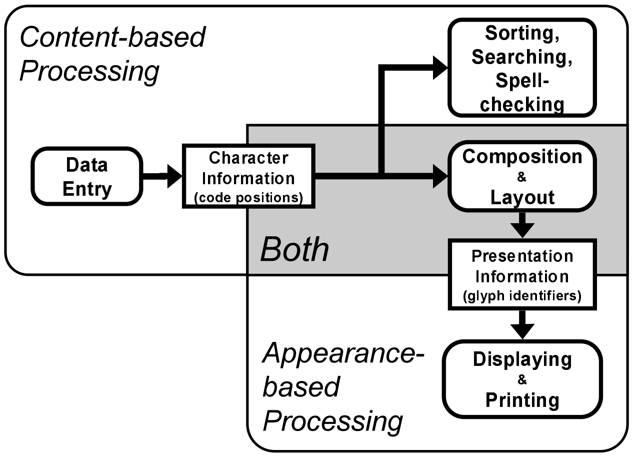
The *character domain* on the left concerns itself with processing the *information content* of text. In this domain, we see operations such as entering characters, ordering and searching text, and checking the spelling and grammar. On the other hand, the *glyph domain* on the right concerns itself with processes related to the *appearance* of the text. In this domain, we see operations to format the text, and to display and print it.

In the middle, we see the operations of glyph selection and substitution to transform characters into glyphs for presentation, and the layout operation to decide, for example, where to split the text at the end of a line. We also see operations from the glyph domain into characters. Character-recognition processes transform glyphs into characters. Mouse selection identifies the character under the mouse cursor.

Notice that the two domains complement each other and the types of processes in one domain are distinct from those in the other domain.



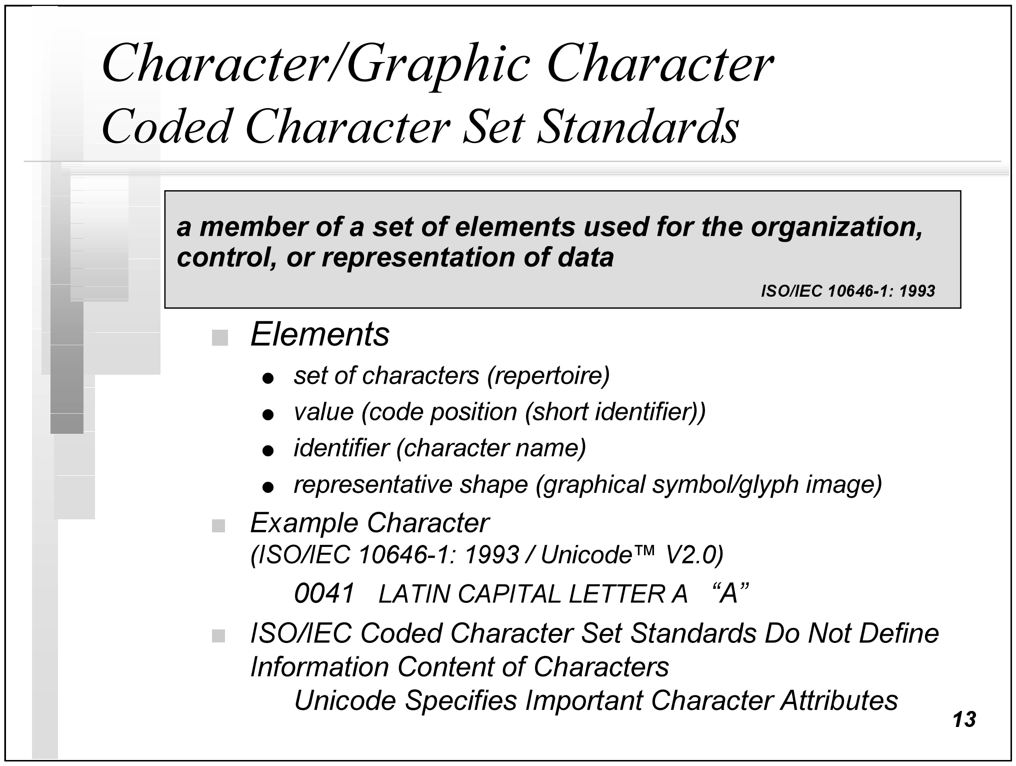
Composition, Layout, and Presentation


Content-based Processing
Sorting, Searching, Spell-checking
Data Entry
Character Information (code positions)
Composition & Layout
Both
Presentation Information (glyph identifiers)
Appearance-based Processing
Displaying & Printing

Let's examine another model for the composition, layout, and presentation functions that you will find in a wordprocessor. This model, in some ways, is an overlay of the domains of previous figure. However, it is interesting because of an area of processing that is in both the character and the glyph domains.

First, lets examine the "Content-base Processing" area of the character domain. Here, we see processes concerned with the information content. The data entry process converts keystrokes into character codes. The character codes are, in turn, input to other information processes like sorting, searching, and spell checking and grammar checking. The character information is also the input to the composition and layout process that produces glyph identifiers as presentation information. Finally, the "Appearance-based Processing" area of the glyph domain takes the glyph identifiers and displays or prints them.

What is interesting is the area labeled "Both" because processes here must be aware of both domains. The composition and layout process uses glyph metrics to know the next position on the line (to know when it reaches the end of a line) and how far down to position the start of the next line from the previous line of text. However, a hyphenation or line-splitting subprocess uses the information domain to know where the word and syllable boundaries are.



Character/Graphic Character
Coded Character Set Standards

a member of a set of elements used for the organization, control, or representation of data

ISO/IEC 10646-1: 1993

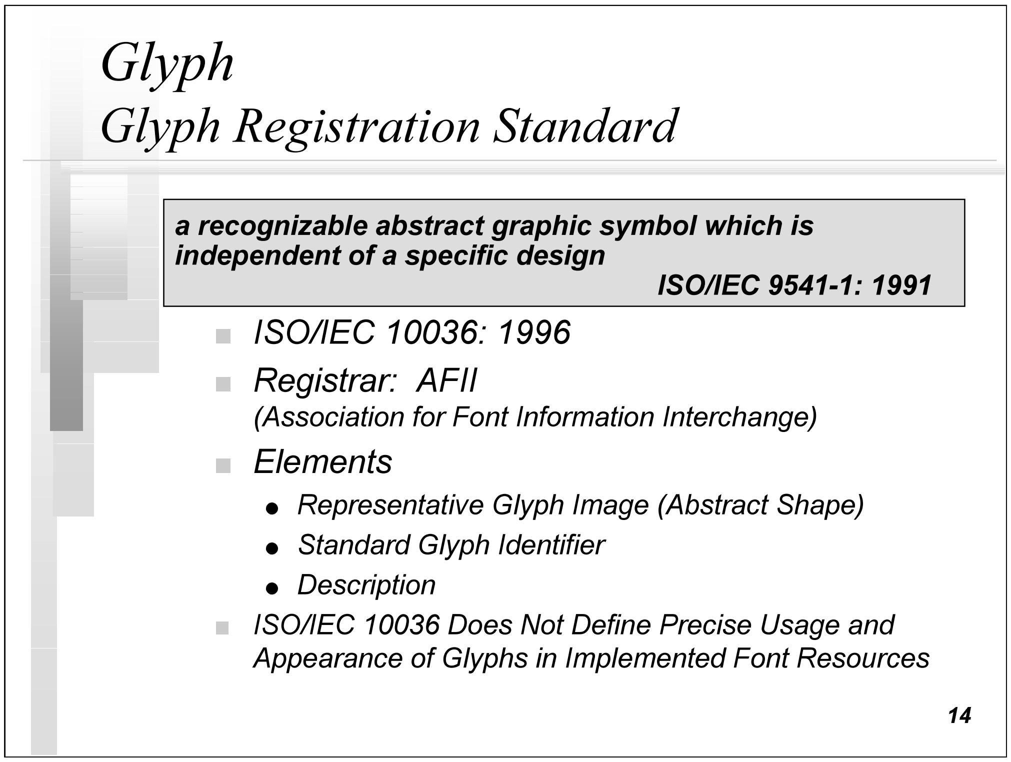
- *Elements*
  - *set of characters (repertoire)*
  - *value (code position (short identifier))*
  - *identifier (character name)*
  - *representative shape (graphical symbol/glyph image)*
- *Example Character*
  (ISO/IEC 10646-1: 1993 / Unicode™ V2.0)
  0041 LATIN CAPITAL LETTER A "A"
- *ISO/IEC Coded Character Set Standards Do Not Define Information Content of Characters*
  Unicode Specifies Important Character Attributes

13

Having seen the model for the character and glyph domains, let's turn to the relevant standards in each domain.

In the character domain, we have coded-character-set standards. ISO defines a *character* as "a member of a set of elements used for the organization, control or representation of data". However, we are concerned with graphic characters rather than control characters. Coded-character-set standards have the following elements: a set of characters (the repertoire), and for each character in the set, code or value, a name, and a representative shape (glyph). For example, in Unicode and ISO/IEC 10646-1, the code 0041 represents LATIN CAPITAL LETTER A, which has a representative shape of "A".

ISO coded-character-set standards do not define the information content of a character. However, Unicode specified important character attributes to aid implementation.



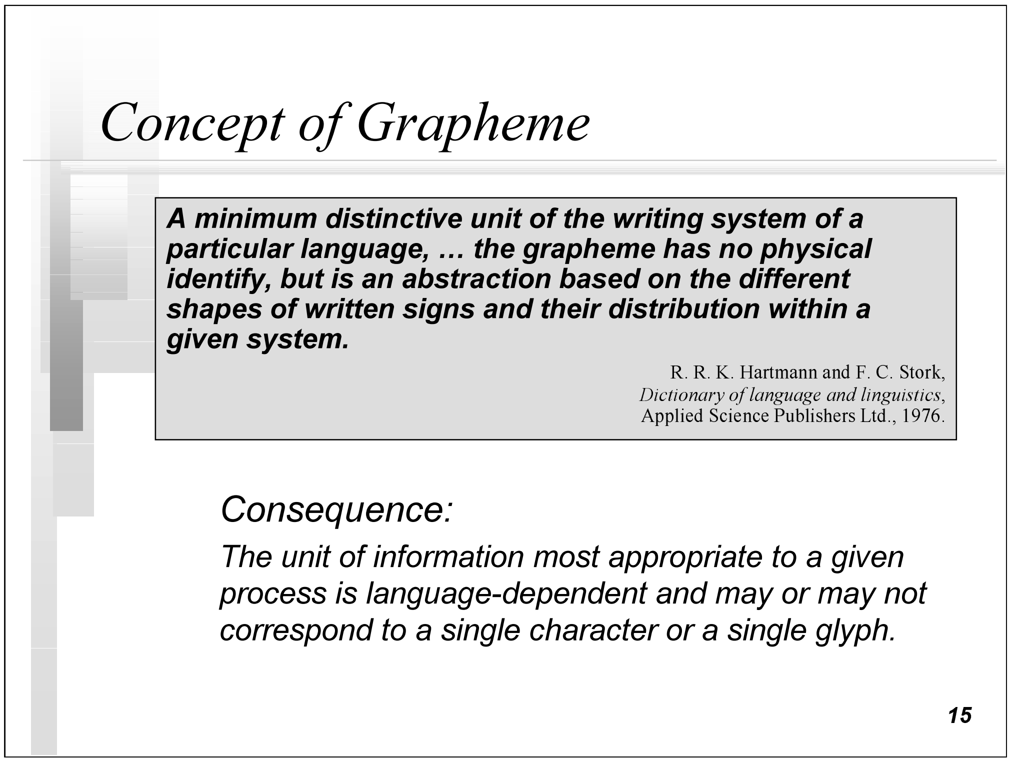
Glyph
Glyph Registration Standard

a recognizable abstract graphic symbol which is independent of a specific design
ISO/IEC 9541-1: 1991

- *ISO/IEC 10036: 1996*
- *Registrar: AFII*
  (Association for Font Information Interchange)
- *Elements*
  - *Representative Glyph Image (Abstract Shape)*
  - *Standard Glyph Identifier*
  - *Description*
- *ISO/IEC 10036 Does Not Define Precise Usage and Appearance of Glyphs in Implemented Font Resources*

14

For the glyph domain, ISO defines a *glyph* as "a recognizable abstract graphic symbol which is independent of a specific design". ISO has the 10036 standard that describes the ISO glyph registry. AFII, the Association for Font Information Interchange, maintains the glyph registry on behalf of ISO. Glyphs registered in the AFII registry have the following elements: a representative shape (which is an abstract shape), a standard glyph identifier, and a description of the glyph. In addition, the standard does not define the precise usage and appearance of glyphs that are implemented in a font (resource).



Concept of Grapheme

A minimum distinctive unit of the writing system of a particular language, … the grapheme has no physical identify, but is an abstraction based on the different shapes of written signs and their distribution within a given system.

R. R. K. Hartmann and F. C. Stork,
Dictionary of language and linguistics,
Applied Science Publishers Ltd., 1976.

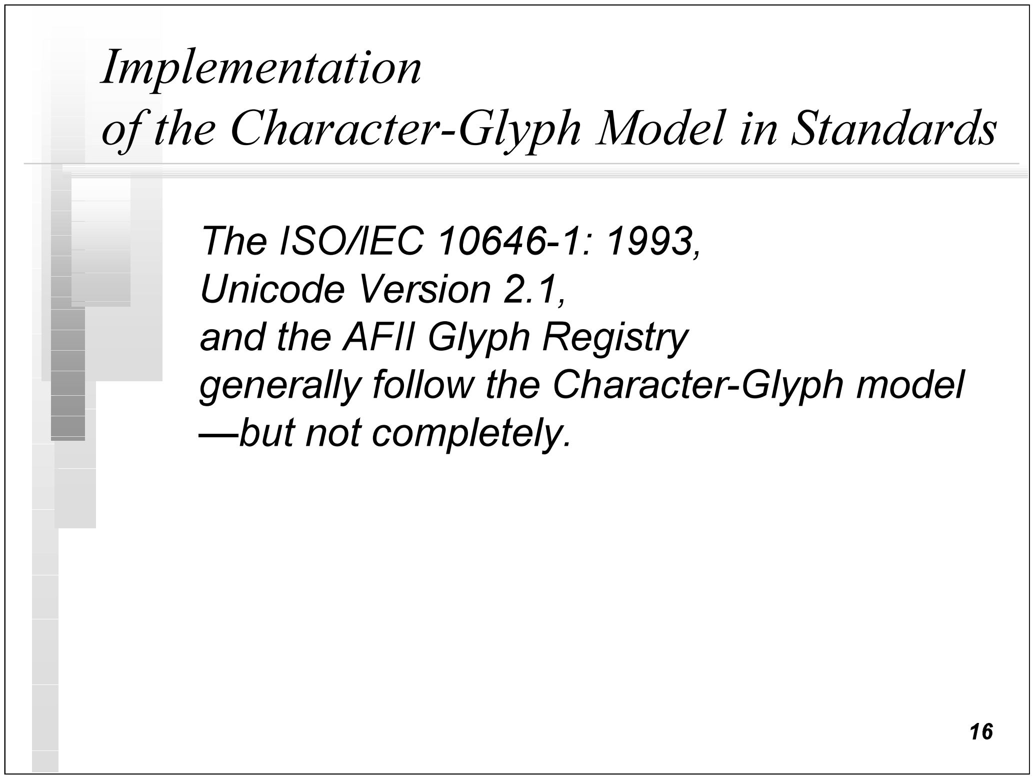
Consequence:

The unit of information most appropriate to a given process is language-dependent and may or may not correspond to a single character or a single glyph.

15

We need to introduce the linguistic concept of a grapheme. A *grapheme* is "the minimum distinctive unit of the writing system of a particular language". This concept can be extended to say that the unit of information most appropriate to a given process is language-dependent and may or may not correspond to a single character or a single glyph.

Consequently, an implementer may decide to transform character codes or glyph identifiers into a different encoding to optimize a particular process.

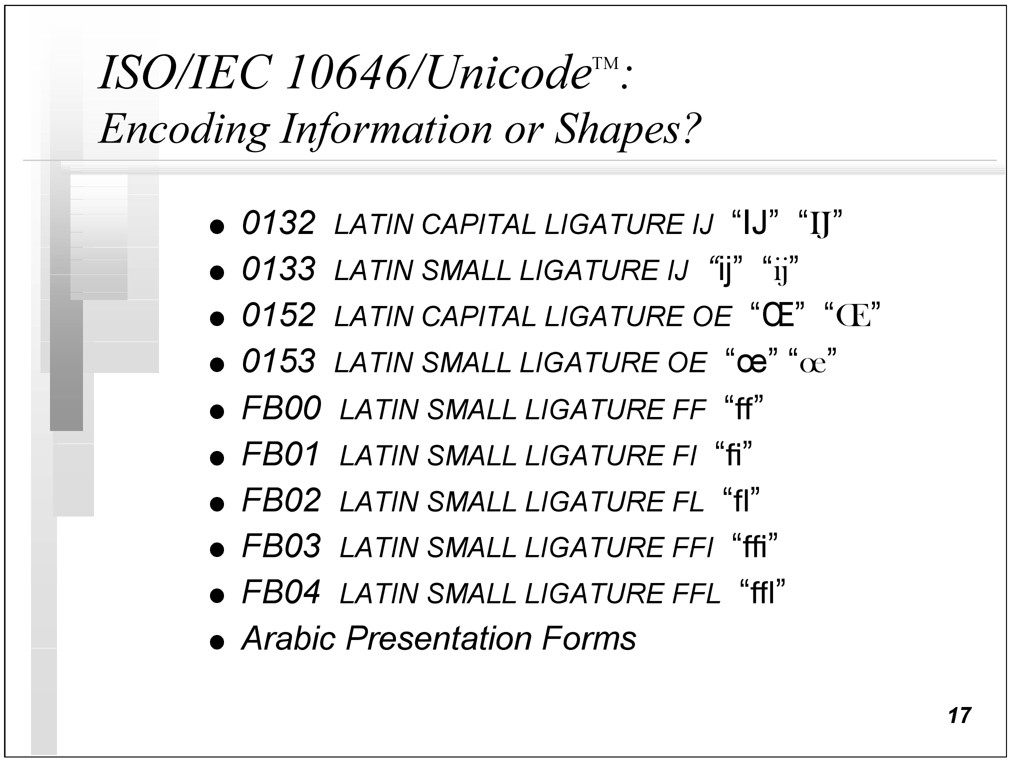


Implementation of the Character-Glyph Model in Standards

The ISO/IEC 10646-1: 1993,
Unicode Version 2.1,
and the AFII Glyph Registry
generally follow the Character-Glyph model
—but not completely.

16

How well are the principles found in this Character-Glyph model followed in the ISO/IEC 10646-1 and the Unicode Version 2.0 standards, and in the AFII Glyph Registry for ISO/IEC 10036? For the most part, they follow the model. However, each has exceptions and the next few figures illustrate some of them.

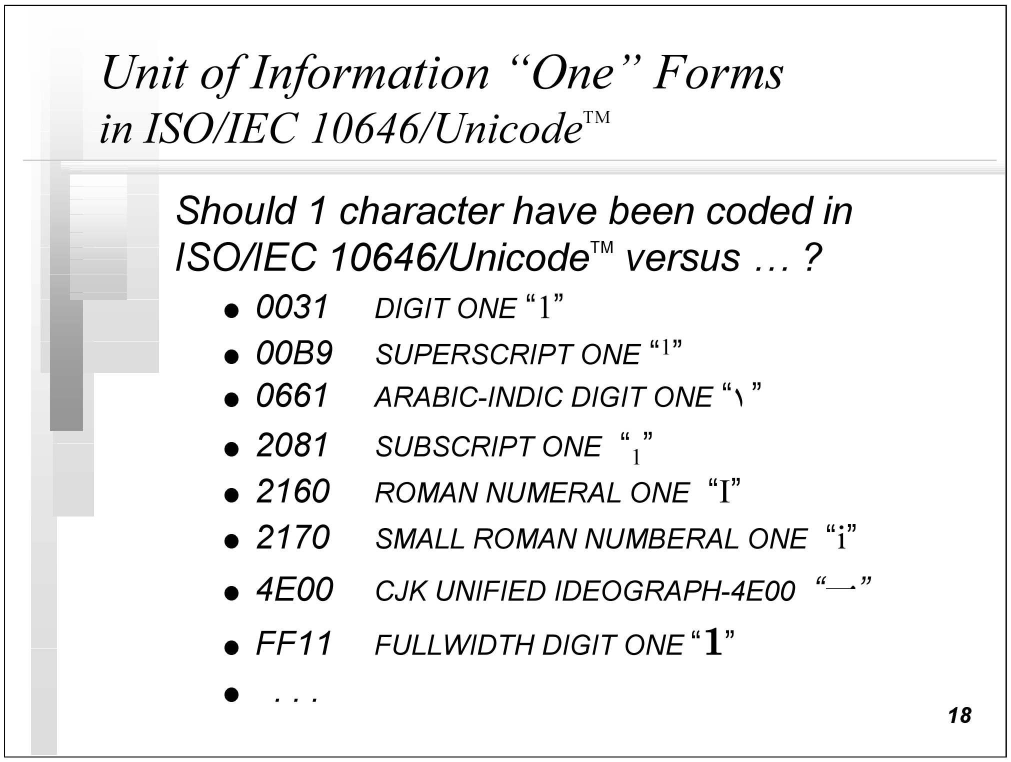


ISO/IEC 10646/Unicode™: Encoding Information or Shapes?

- *0132* *LATIN CAPITAL LIGATURE IJ* “Ĳ” “Ĳ”
- *0133* *LATIN SMALL LIGATURE IJ* “ĳ” “ĳ”
- *0152* *LATIN CAPITAL LIGATURE OE* “Œ” “Œ”
- *0153* *LATIN SMALL LIGATURE OE* “œ” “œ”
- *FB00* *LATIN SMALL LIGATURE FF* “ﬀ”
- *FB01* *LATIN SMALL LIGATURE FI* “ﬁ”
- *FB02* *LATIN SMALL LIGATURE FL* “ﬂ”
- *FB03* *LATIN SMALL LIGATURE FFI* “ﬃ”
- *FB04* *LATIN SMALL LIGATURE FFL* “ﬄ”
- *Arabic Presentation Forms*

17

First, ISO/IEC 10646 and Unicode code several ligatures for compatibility with existing coded-character-set standards. Here, you can see ligatures for “IJ”, “OE”, and the various “FF”, “FI”, and “FFI” and “FFL” ligatures. In addition, ISO/IEC 10646 and Unicode code presentation forms and an extensive set of ligatures for Arabic letters. This was done as part of the merger of 10646 and Unicode. According to the Character-Glyph Model, characters for these ligatures and presentation forms should not have been encoded because they are needed only for rendering the characters.



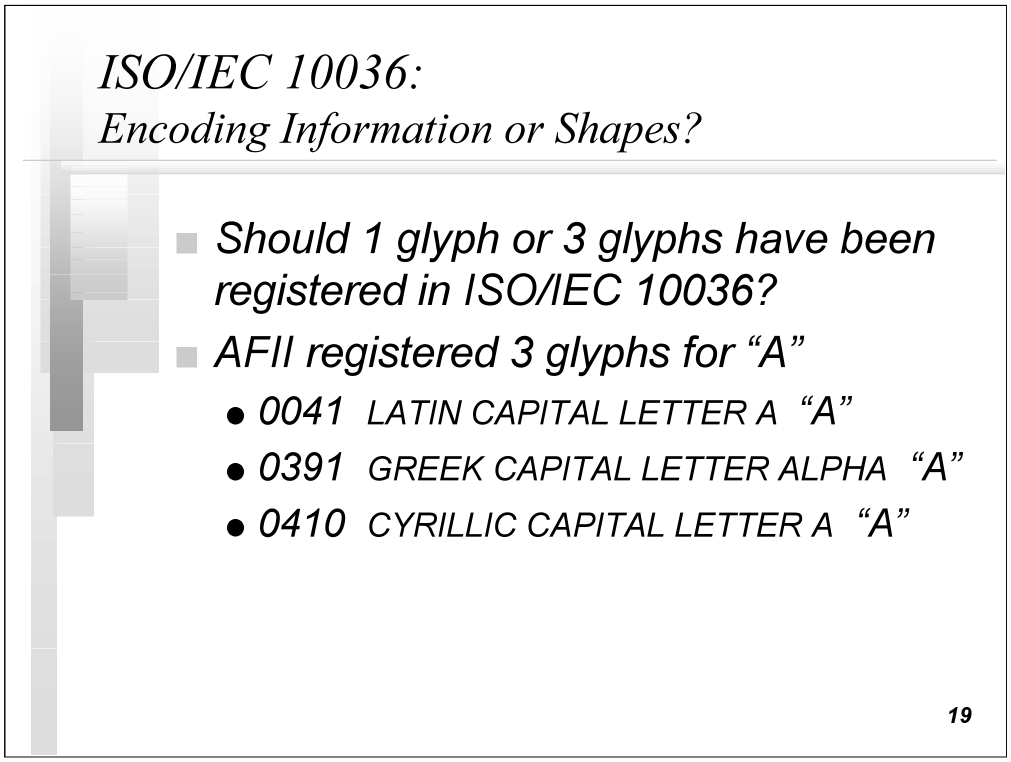
Unit of Information "One" Forms in ISO/IEC 10646/Unicode™

Should 1 character have been coded in ISO/IEC 10646/Unicode™ versus … ?

● *0031*	*DIGIT ONE*	"1"
● *00B9*	*SUPERSCRIPT ONE*	"^{1}"
● *0661*	*ARABIC-INDIC DIGIT ONE*	"١"
● *2081*	*SUBSCRIPT ONE*	"$_{1}$"
● *2160*	*ROMAN NUMERAL ONE*	"I"
● *2170*	*SMALL ROMAN NUMBERAL ONE*	"i"
● *4E00*	*CJK UNIFIED IDEOGRAPH-4E00*	"一"
● *FF11*	*FULLWIDTH DIGIT ONE*	"１"
● . . .		

18

Similarly, could coding one character for the concept of "one" or "unity" been sufficient in 10646 and Unicode? The ISO/IEC Technical Report lists about 30 different characters that are encoded to represent this concept. Some of the characters are different forms for the same script (DIGIT ONE, SUPERSCRIPT ONE, SUBSCRIPT ONE, FULLWIDTH DIGIT ONE); others are different forms for the various scripts. Don't all of these forms encode the same information?

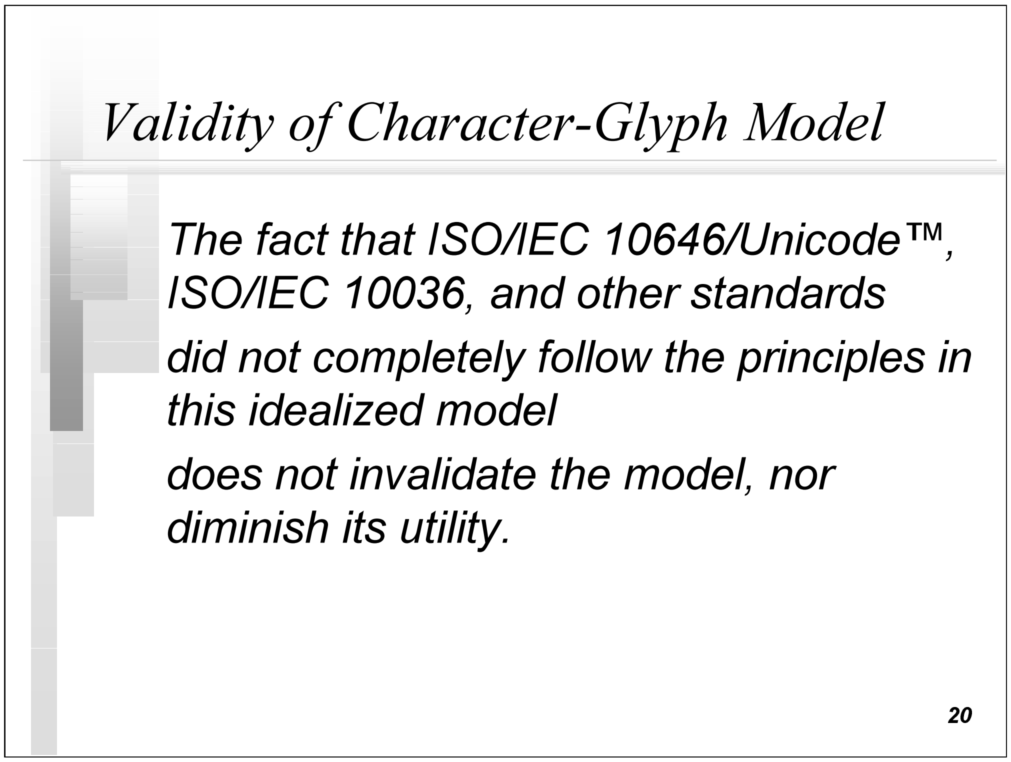


ISO/IEC 10036:
Encoding Information or Shapes?

- *Should 1 glyph or 3 glyphs have been registered in ISO/IEC 10036?*
- *AFII registered 3 glyphs for "A"*
  - *0041 LATIN CAPITAL LETTER A "A"*
  - *0391 GREEK CAPITAL LETTER ALPHA "Α"*
  - *0410 CYRILLIC CAPITAL LETTER A "А"*

19

Let's switch our focus to the AFII Glyph Registry for ISO/IEC 10036. Did AFII need to register 3 glyphs for the shape needed for the LATIN CAPITAL LETTER A, and the GREEK CAPITAL LETTER ALPHA, and the CYRILLIC CAPITAL LETTER A? Are not all of these the same abstract shape?



Validity of Character-Glyph Model

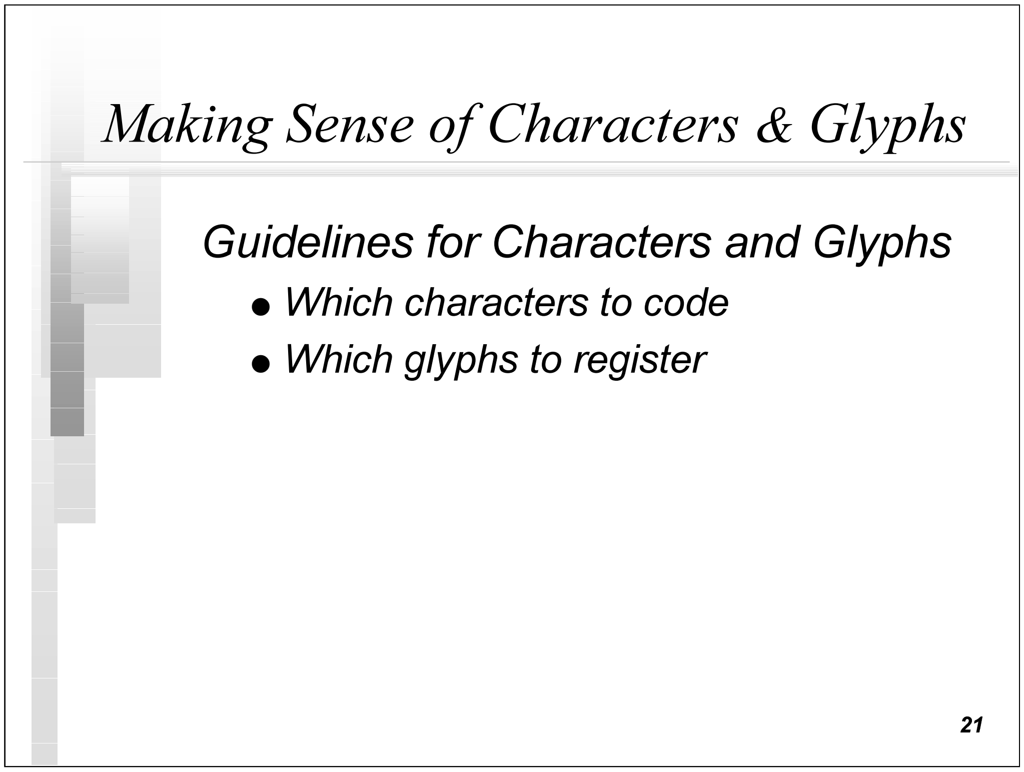
The fact that ISO/IEC 10646/Unicode™, ISO/IEC 10036, and other standards

did not completely follow the principles in this idealized model

does not invalidate the model, nor diminish its utility.

20

Even though the standards did not completely follow the principles in the idealized Character-Glyph Model, does not invalidate the model, nor diminish its utility. The model is very important to understanding the concepts of a “character” and a “glyph”, and how to use them to render characters into glyphs.



Making Sense of Characters & Glyphs

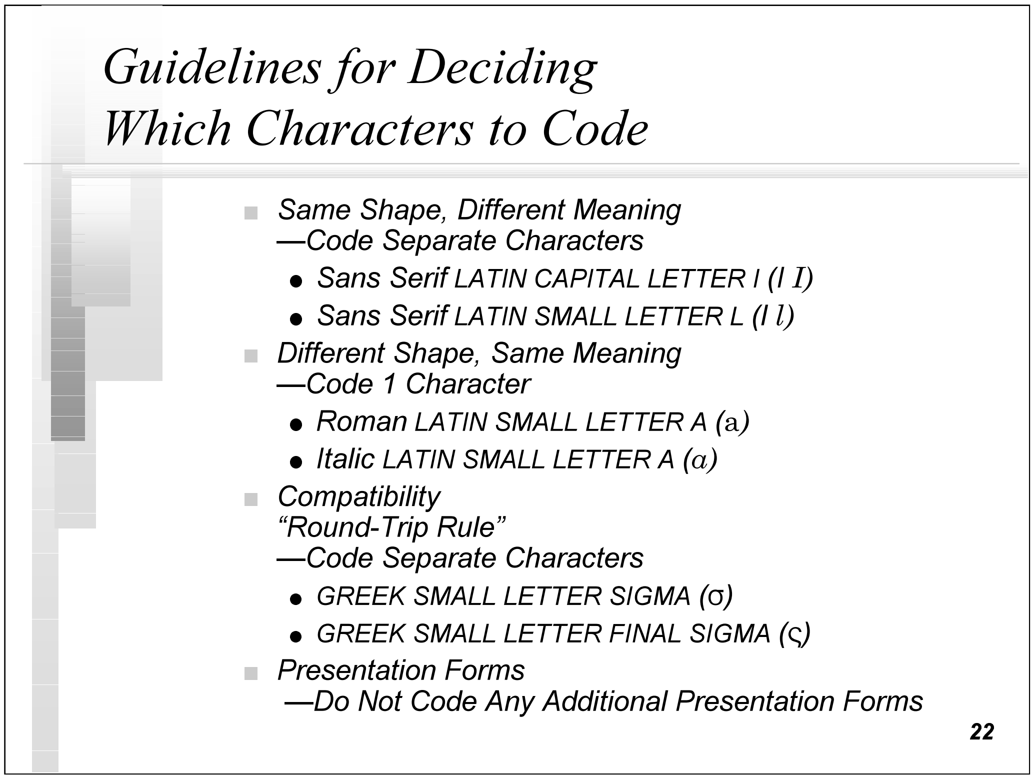
Guidelines for Characters and Glyphs

- *Which characters to code*
- *Which glyphs to register*

21

Given that ISO did not follow the model in developing ISO/IEC 10646 and AFII did not follow them for registering glyphs, can we derive any benefit from this model? In fairness to ISO, it could not be expected to follow the principles in this report when, in fact, the first draft of the report was written after ISO/IEC 10646-1 and the Unicode Standard, Version 1.0 were published.

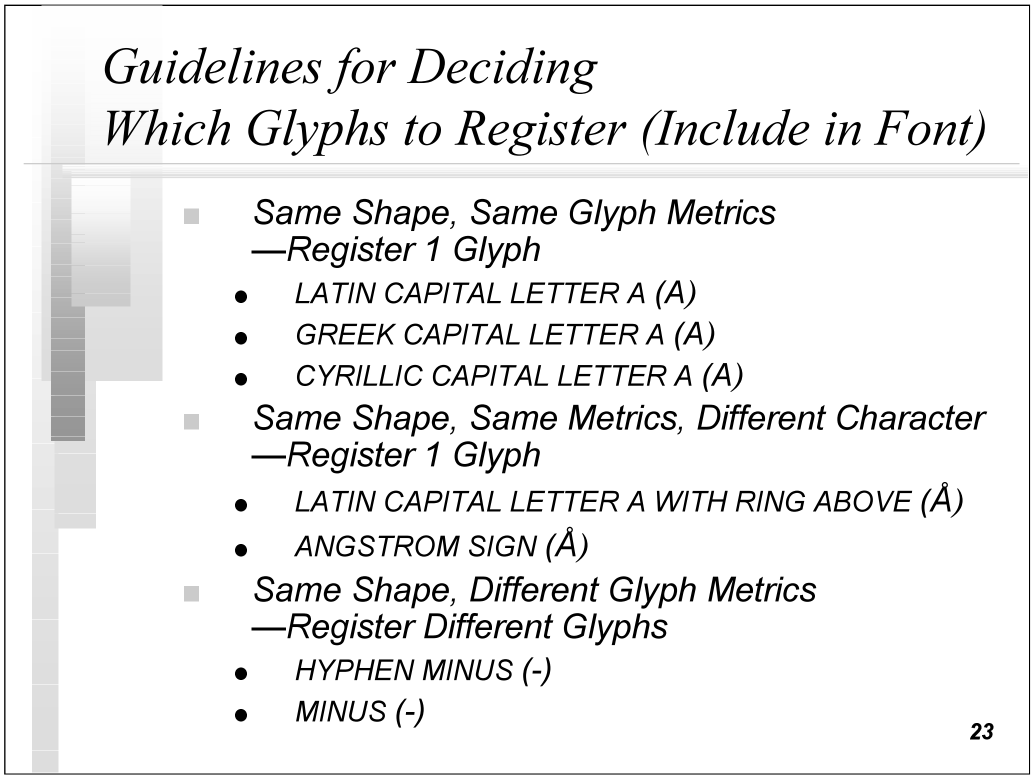
Given the existence of the model, does it provide ISO any guidelines for deciding which characters to encode, and which glyphs should be registered? The answer is that it does provide such guidance.



Guidelines for Deciding Which Characters to Code

- *Same Shape, Different Meaning*
  —Code Separate Characters
  - *Sans Serif* LATIN CAPITAL LETTER I (I I)
  - *Sans Serif* LATIN SMALL LETTER L (l l)
- *Different Shape, Same Meaning*
  —Code 1 Character
  - *Roman* LATIN SMALL LETTER A (a)
  - *Italic* LATIN SMALL LETTER A (ɑ)
- *Compatibility*
  "Round-Trip Rule"
  —Code Separate Characters
  - GREEK SMALL LETTER SIGMA (σ)
  - GREEK SMALL LETTER FINAL SIGMA (ς)
- *Presentation Forms*
  —Do Not Code Any Additional Presentation Forms

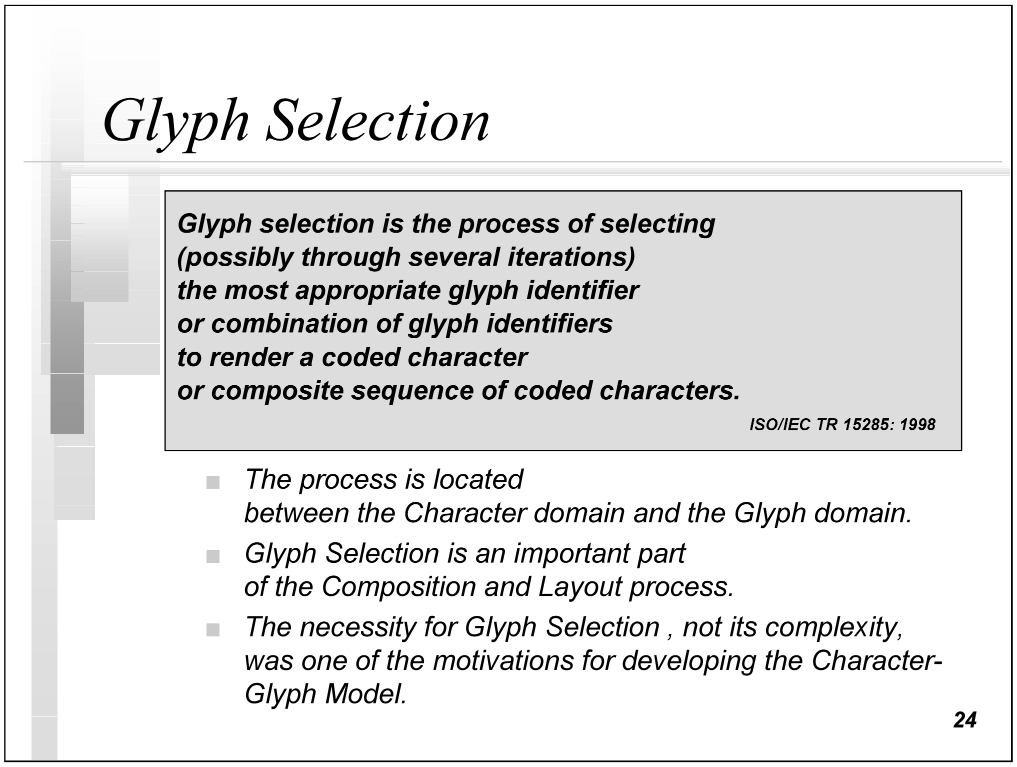
22



Guidelines for Deciding Which Glyphs to Register (Include in Font)

- *Same Shape, Same Glyph Metrics —Register 1 Glyph*
  - *LATIN CAPITAL LETTER A (A)*
  - *GREEK CAPITAL LETTER A (Α)*
  - *CYRILLIC CAPITAL LETTER A (А)*
- *Same Shape, Same Metrics, Different Character —Register 1 Glyph*
  - *LATIN CAPITAL LETTER A WITH RING ABOVE (Å)*
  - *ANGSTROM SIGN (Å)*
- *Same Shape, Different Glyph Metrics —Register Different Glyphs*
  - *HYPHEN MINUS (-)*
  - *MINUS (−)*

23



Glyph Selection

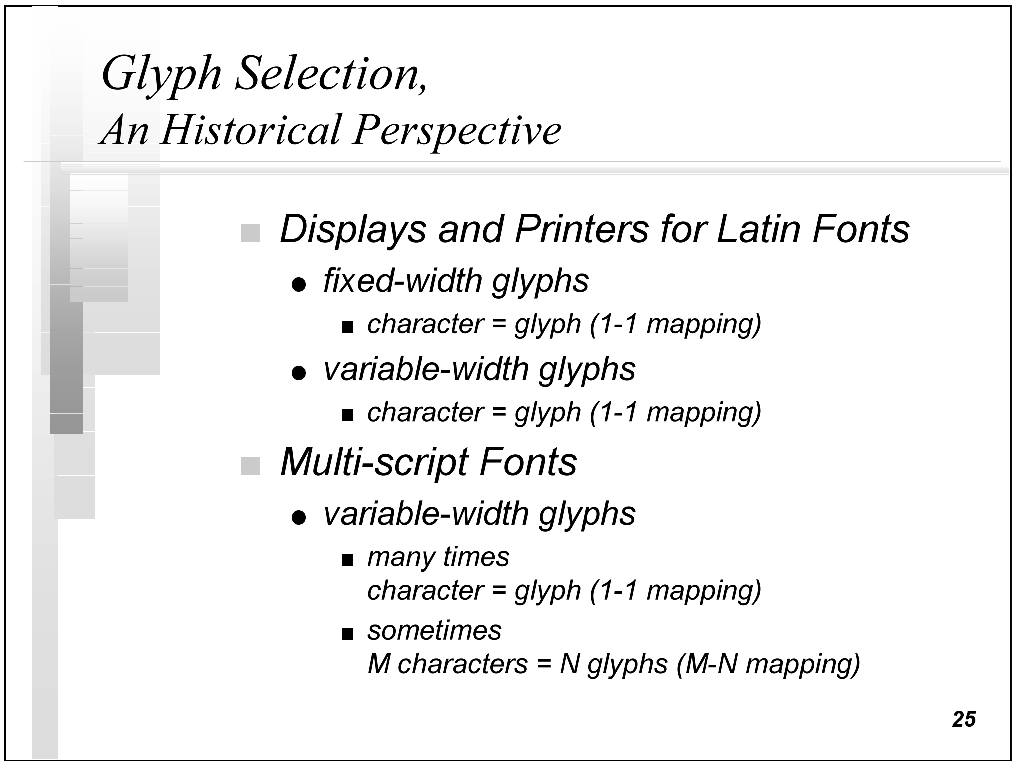
Glyph selection is the process of selecting (possibly through several iterations) the most appropriate glyph identifier or combination of glyph identifiers to render a coded character or composite sequence of coded characters.

ISO/IEC TR 15285: 1998

- *The process is located between the Character domain and the Glyph domain.*
- *Glyph Selection is an important part of the Composition and Layout process.*
- *The necessity for Glyph Selection , not its complexity, was one of the motivations for developing the Character-Glyph Model.*

24

Quoting from the Technical Report (ISO/IEC TR 15285), “Glyph selection is the process of selecting (possibly through several iterations) the most appropriate glyph identifier or combination of glyph identifiers to render a coded character or composite sequence of coded characters.” The process is located between the Character domain and the Glyph domain and converts coded character into glyph identifiers. This process is an important part of the composition and layout process. Moreover, the necessity for glyph-selection, not its complexity, was one of the motivations for developing the Character-Glyph Model.



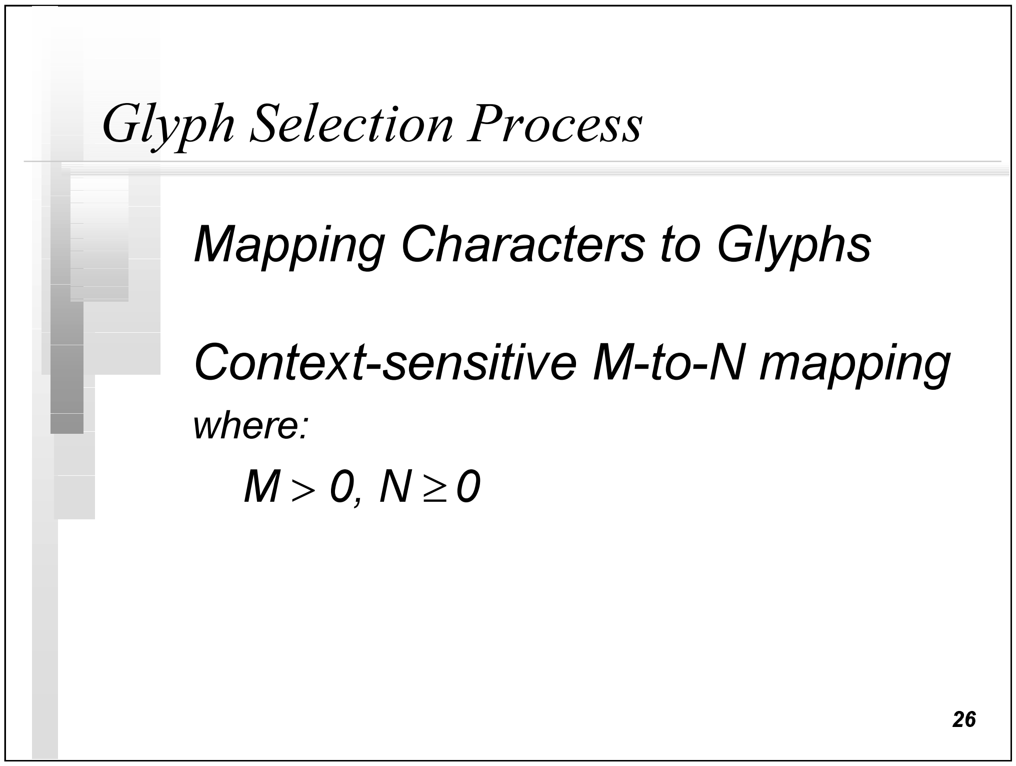
Glyph Selection, An Historical Perspective

- *Displays and Printers for Latin Fonts*
  - *fixed-width glyphs*
    - *character = glyph (1-1 mapping)*
  - *variable-width glyphs*
    - *character = glyph (1-1 mapping)*
- *Multi-script Fonts*
  - *variable-width glyphs*
    - *many times*
      character = glyph (1-1 mapping)
    - *sometimes*
      M characters = N glyphs (M-N mapping)

25

One of the reasons for the confusion is the historical development of printing from computers. In the early days of computing, no one distinguished a character from a glyph. Printers would only print digits, English letters and a few symbols. (In fact, it was rare to have a printer that would print both upper-case and lower-case characters.) At this time, the mapping between characters and glyphs was one-to-one and defined by the coding of the characters. At that time, no one conceived of any need to distinguish characters and glyphs—we thought that they were the same thing! This concept prevailed even as the vendors delivered more sophisticated printers that would print variable-width, proportionally-spaced "characters" [i.e., glyphs].

At this point, for the most part the one-to-one relationship between characters and glyphs still holds for the majority of characters in 10646 and Unicode. However, sometimes the relationship fails to render the characters correctly; sometimes something more sophisticated is needed to render the characters correctly.



Glyph Selection Process

Mapping Characters to Glyphs

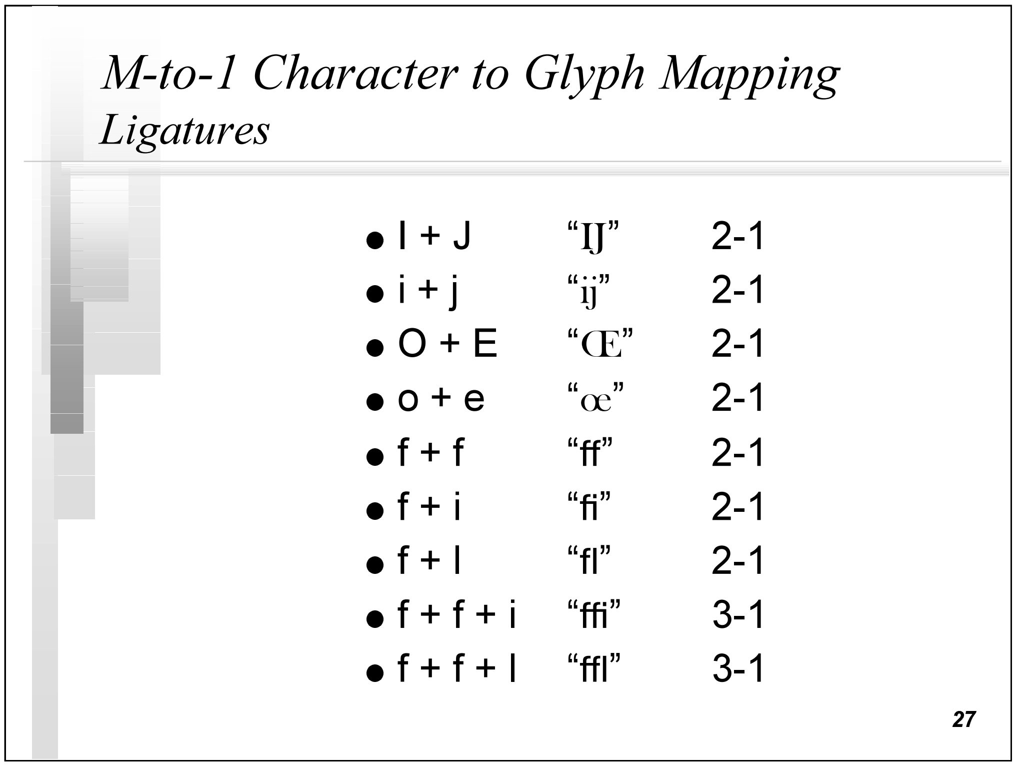
Context-sensitive M-to-N mapping

where:

$M > 0, N \geq 0$

26

Sometimes the glyph-selection process for 10646 and Unicode characters needs to be extended from a one-to-one mapping. In the general case, glyph selection is a context-sensitive mapping of *M* characters into *N* glyphs. The next few figures will show examples of the general case.

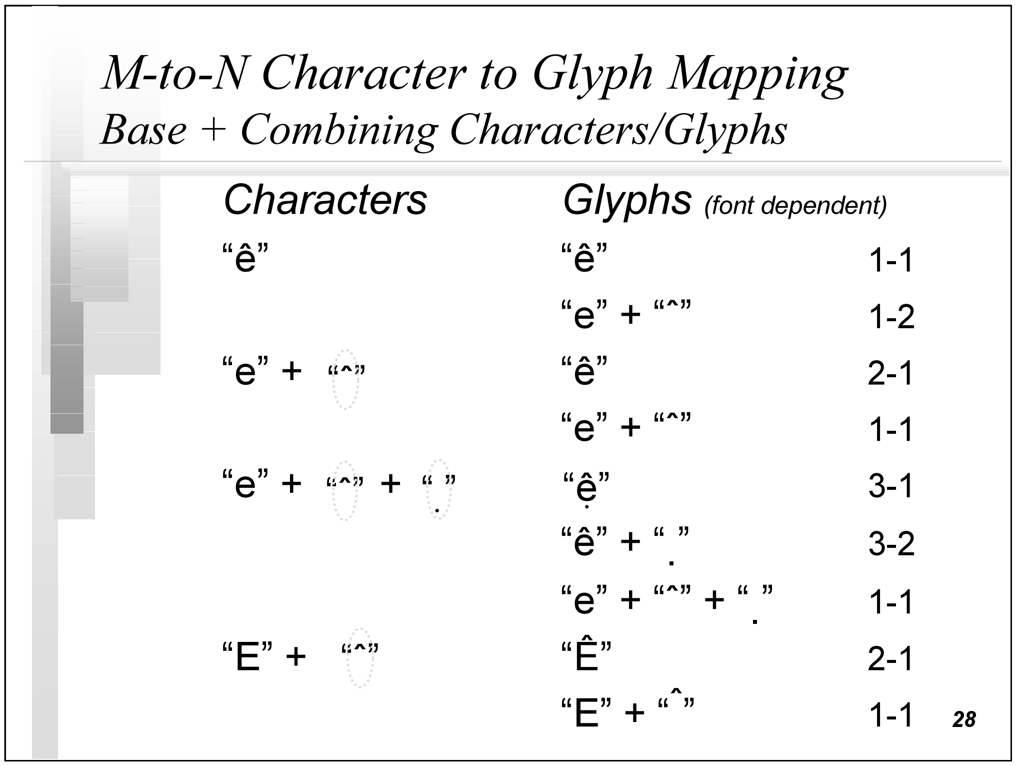


M-to-1 Character to Glyph Mapping
Ligatures

● I + J	“IJ”	2-1
● i + j	“ij”	2-1
● O + E	“Œ”	2-1
● o + e	“œ”	2-1
● f + f	“ﬀ”	2-1
● f + i	“ﬁ”	2-1
● f + l	“ﬂ”	2-1
● f + f + i	“ﬃ”	3-1
● f + f + l	“ﬄ”	3-1

27

What you see here are several examples of an M-to-1 one glyph mapping. This occurs with ligatures used in the Latin script. The “IJ” ligatures could easily be implemented with a Kerning table. (A “Kerning table” in a “font” describes the spacing required between each pair of glyphs.) If fact, I believe that the illustrated glyphs in the figure are Kerned “I” and “J” glyphs rather than an “IJ” ligature glyph. For mapping into the “OE” ligatures in French, the glyph selection process would need to know the context when the “Œ” ligature glyph is used instead of separate “O” and “E” glyphs. The last 5 ligatures illustrate various forms of the “FI” and “FL” ligatures. Note that the last two represent a 3-to-1 mapping. Also note that these ligatures occur more frequently in a serif font (like Times New Roman) rather than a sans serif font, as illustrated.

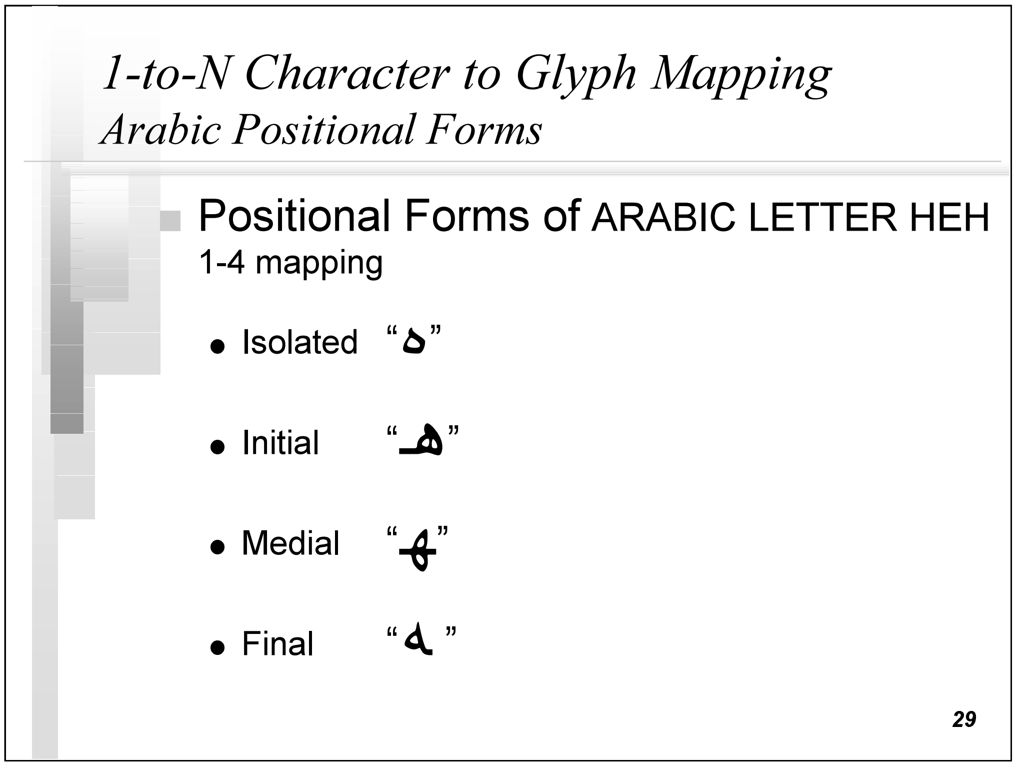


M-to-N Character to Glyph Mapping
Base + Combining Characters/Glyphs

Characters	Glyphs (font dependent)	
“ê”	“ê”	1-1
	“e” + “^”	1-2
“e” + “^”	“ê”	2-1
	“e” + “^”	1-1
“e” + “^” + “ ̣”	“ệ”	3-1
	“ê” + “ ̣”	3-2
	“e” + “^” + “ ̣”	1-1
“E” + “^”	“Ê”	2-1
	“E” + “ ̂”	1-1

28

Depending on the particular font implementation, the combination sequences of 10646 and Unicode may require more sophisticated mapping. Recall that a combining sequence consists of a base character followed by one or more combining characters. Combining sequences are frequently used to code accented characters. Recall also that many accented characters may be represented in 10646 and Unicode as either a single character or a combining sequence. The illustrated LATIN LETTER E WITH CIRCUMFLEX is one of these. Note that just as the accented characters may be encoded as a single character or a combining sequence in 10646 and Unicode, a font designer may choose to implement accented characters as either a single glyph or multiple glyphs that correspond to an encoded combining sequence. This figure illustrates several possible implementations and the corresponding mapping. Note that for the placement of the COMBINING CIRCUMFLEX ACCENT glyph needs to be higher for the LATIN CAPITAL LETTER E glyph than for the LATIN SMALL LETTER E glyph.

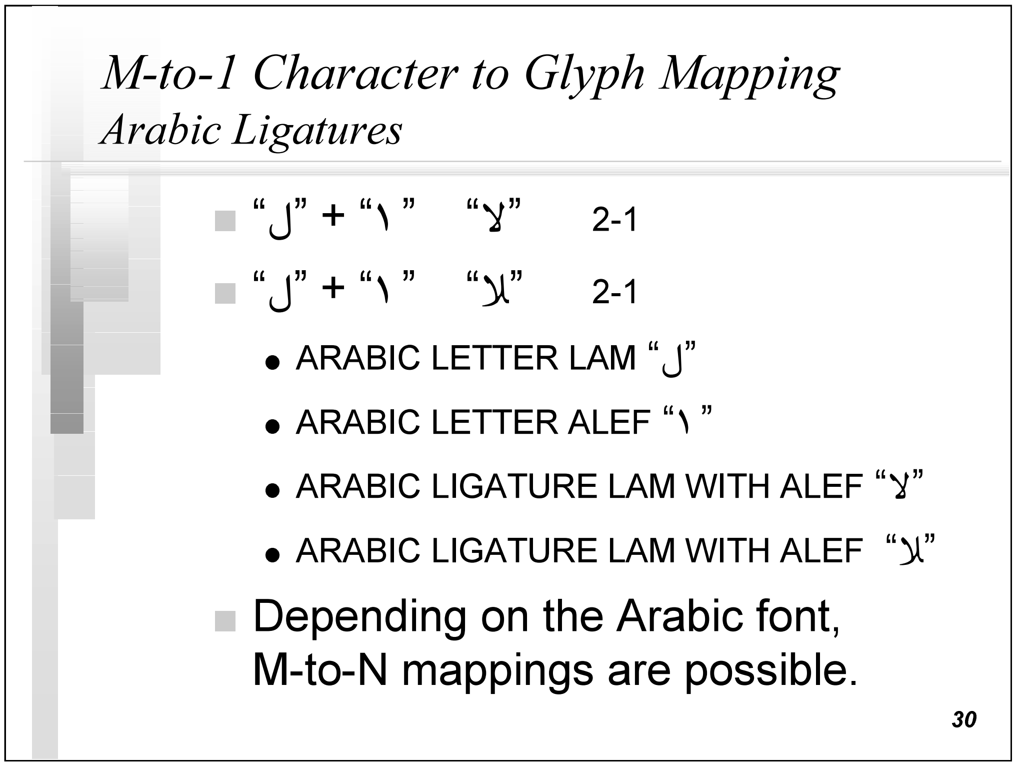


1-to-N Character to Glyph Mapping
Arabic Positional Forms

- Positional Forms of ARABIC LETTER HEH
  1-4 mapping
  - Isolated "ﻩ"
  - Initial "ﻫ"
  - Medial "ﻬ"
  - Final "ﻪ "

29

Recall that Arabic is a cursive script where the characters in words are connected. Minimum legibility of the Arabic script requires a 1-to-4 mapping to account for when the letter is alone (isolated form), at the beginning of a word (initial form), in the middle of a word (medial form), or at the end of a word (final form). This figure illustrates four forms of the ARABIC LETTER HEH. (However, Thomas Milo indicated that a fifth form, which looks like the initial form without the trailing connector, is required for enumeration to distinguish it from the ARABIC-INDIC DIGIT 5.)



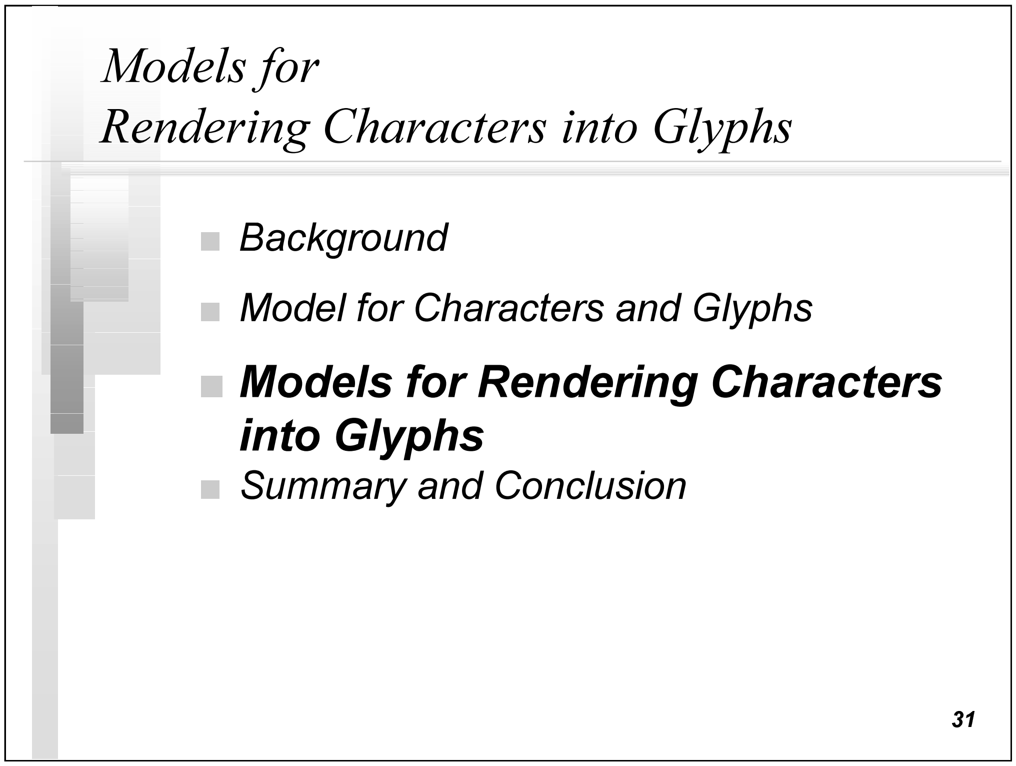
M-to-1 Character to Glyph Mapping
Arabic Ligatures

- “ل” + “ا ”   “ﻻ”   2-1
- “ل” + “ا ”   “ﻼ”   2-1
  - ARABIC LETTER LAM “ل”
  - ARABIC LETTER ALEF “ا ”
  - ARABIC LIGATURE LAM WITH ALEF “ﻻ”
  - ARABIC LIGATURE LAM WITH ALEF “ﻼ”
- Depending on the Arabic font, M-to-N mappings are possible.

30

Arabic fonts also require glyphs for the LAM WITH ALEF ligature. Note that like the Arabic letters, this ligature has forms that depend on the position of the two Arabic letters in a word.

To obtain minimum legibility for the Arabic script, a font must contain all of the presentation forms for the Arabic letters plus the LAM WITH ALEF ligature. However, this is the minimum that might be used, for example, in an Arabic newspaper. Rendering Arabic for books, requires that the fonts contain an extensive set of Arabic ligatures which, in turn, requires a much more sophisticated glyph selection process than the one required for minimum legibility of Arabic.



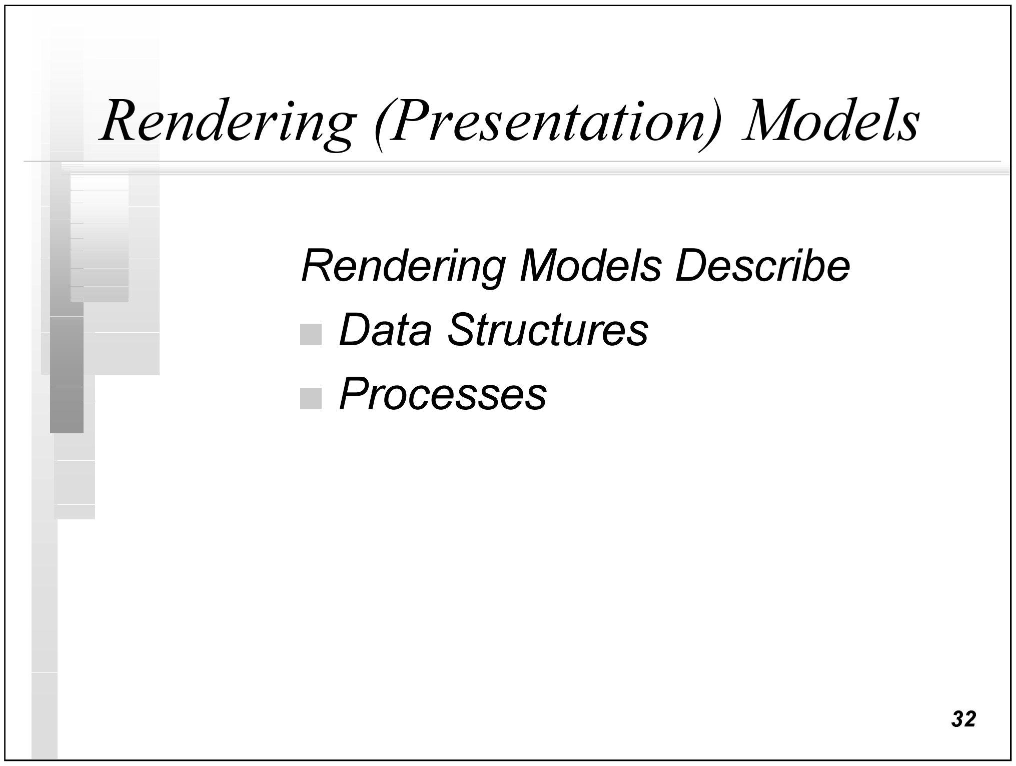
Models for Rendering Characters into Glyphs

- *Background*
- *Model for Characters and Glyphs*
- ***Models for Rendering Characters into Glyphs***
- *Summary and Conclusion*

31

We are now at the next major section of this presentation.

In the previous section, we discussed the character domain and the glyph domain. We also discussed the glyph-selection process that maps characters into glyphs for displaying or printing. In this section, we will discuss three technologies for rendering characters into glyphs. At least two of these technologies are implemented now, and the third if it has not yet be delivered in products is not far from delivery.



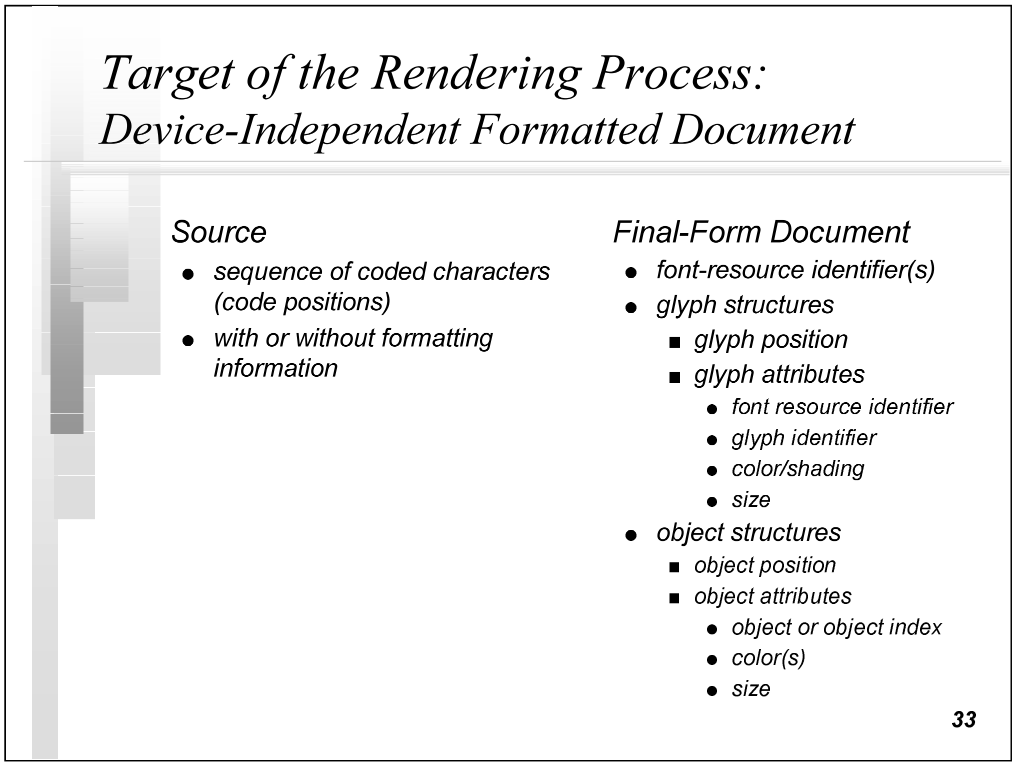
Rendering (Presentation) Models

Rendering Models Describe

- Data Structures
- Processes

32

The three technologies rely on both data structures and processes to render characters.



Target of the Rendering Process: Device-Independent Formatted Document

Source

- *sequence of coded characters (code positions)*
- *with or without formatting information*

Final-Form Document

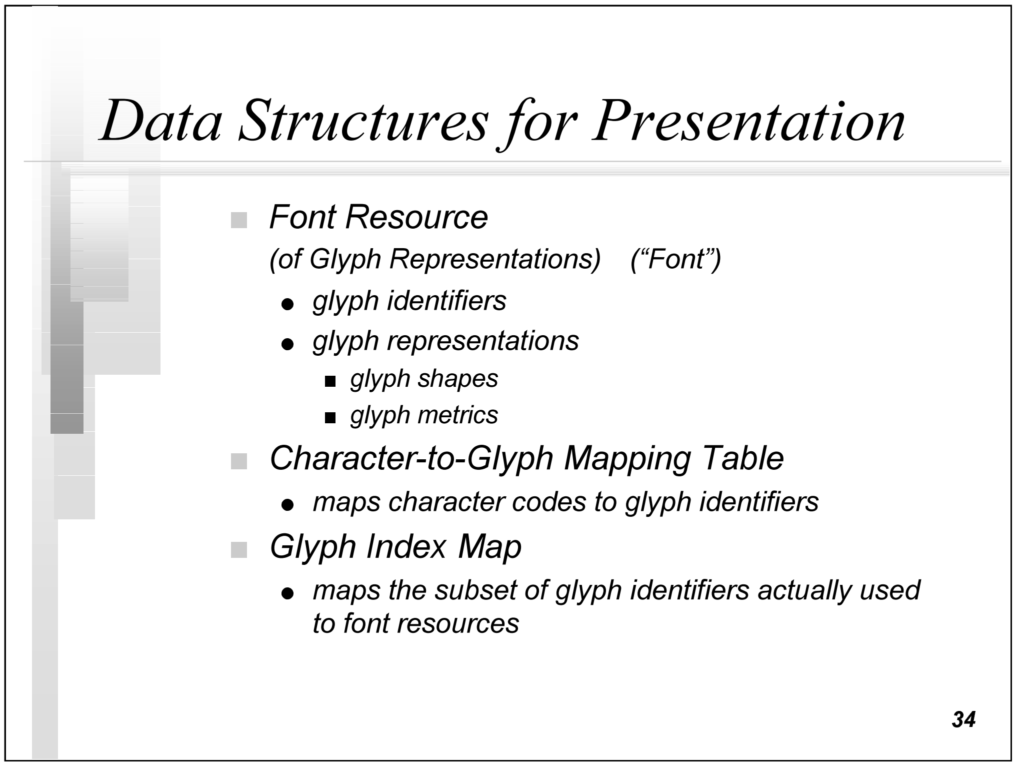
- *font-resource identifier(s)*
- *glyph structures*
  - *glyph position*
  - *glyph attributes*
    - *font resource identifier*
    - *glyph identifier*
    - *color/shading*
    - *size*
- *object structures*
  - *object position*
  - *object attributes*
    - *object or object index*
    - *color(s)*
    - *size*

33

However, before we start discussing the data structures, let’s discuss the inputs and outputs.

The input to the process is a sequence of coded characters (code positions). The input may or may not contain information for formatting the characters (bold, italic, underline, headers, paragraphs, lists, etc.).

The output of the process is what is called a “final-form document”, which is ready for printing or displaying. In general, this output is independent of the particular output device, be it a display or a printer. The final form document consists of a list of fonts used in the document (font-resource identifiers), a list of glyph n-tupples (which typically contain the glyph position and glyph attributes such as the font resource, the glyph identifier, its color or shading, its size, etc.), and object structures (which contains the object’s position and attributes).



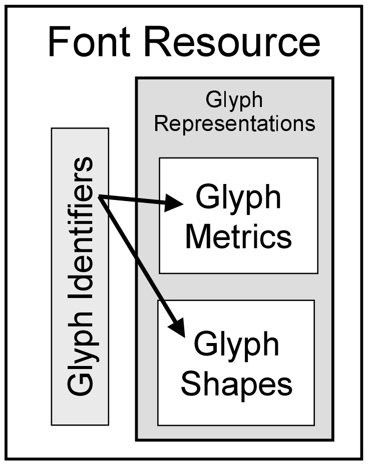
Data Structures for Presentation

- *Font Resource*
  (of Glyph Representations) ("Font")
  - *glyph identifiers*
  - *glyph representations*
    - *glyph shapes*
    - *glyph metrics*
- *Character-to-Glyph Mapping Table*
  - *maps character codes to glyph identifiers*
- *Glyph Index Map*
  - *maps the subset of glyph identifiers actually used to font resources*

34

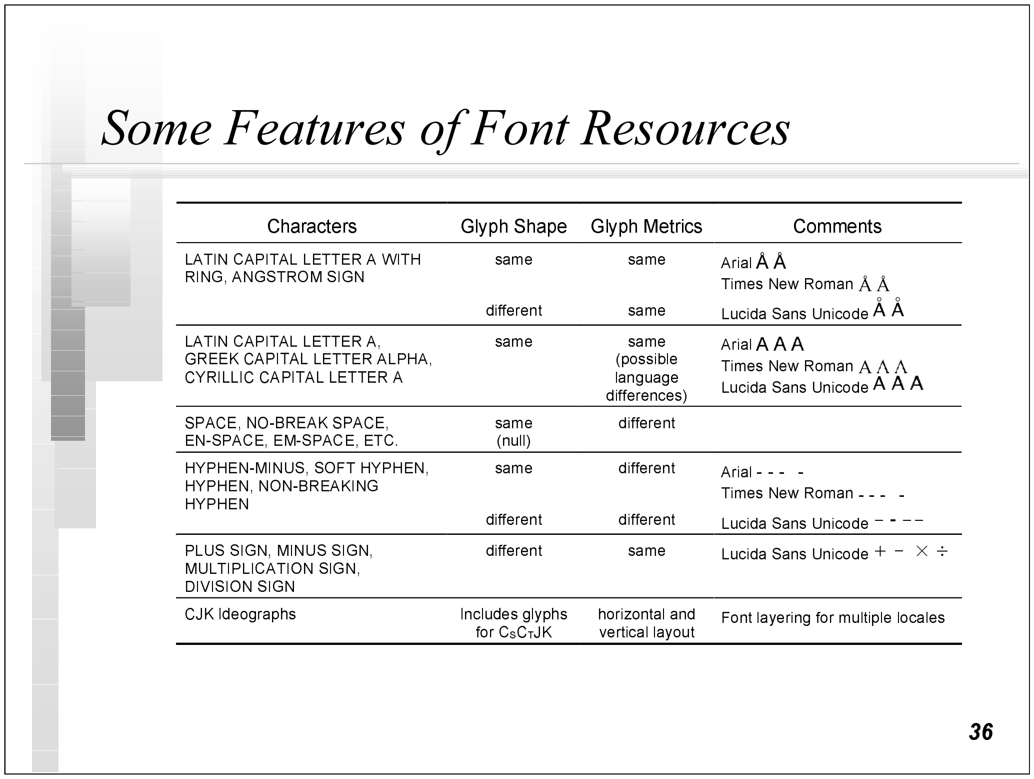
We will be discussing three data structures: the "Font Resource" typically called a "Font", the Character-to-Glyph Mapping Table, and a Glyph Index Map. Note that these are not the only data structures used, they are merely the most common ones.

Data Structures:
Font Resource (“Font”) of Glyph Representations



35

A Font Resource contains information to describe the set of glyphs in the font. It contains “Glyph Metrics” (which describe a glyph’s height, width, placement relative to the baseline, and space from the previous character), and “Glyph Shapes” (which may be either a bitmap or the outline with some optional hinting). The Font Resource also includes a table to map each glyph identifier into its metric and shape information. This is the minimum information included in a Font Resource. However, the font designer may choose to include additional information.



Some Features of Font Resources

Characters	Glyph Shape	Glyph Metrics	Comments
LATIN CAPITAL LETTER A WITH RING, ANGSTROM SIGN	same	same	Arial Å Å Times New Roman Å Å
	different	same	Lucida Sans Unicode Å Å
LATIN CAPITAL LETTER A, GREEK CAPITAL LETTER ALPHA, CYRILLIC CAPITAL LETTER A	same	same (possible language differences)	Arial A Α А Times New Roman A Α А Lucida Sans Unicode A Α А
SPACE, NO-BREAK SPACE, EN-SPACE, EM-SPACE, ETC.	same (null)	different	
HYPHEN-MINUS, SOFT HYPHEN, HYPHEN, NON-BREAKING HYPHEN	same	different	Arial - - - - Times New Roman - - - -
	different	different	Lucida Sans Unicode − - − −
PLUS SIGN, MINUS SIGN, MULTIPLICATION SIGN, DIVISION SIGN	different	same	Lucida Sans Unicode + − × ÷
CJK Ideographs	Includes glyphs for C_SC_TJK	horizontal and vertical layout	Font layering for multiple locales

36

This table illustrates some of the ways that the font resource designers implement fonts to reuse common data.

In the first row, the Arial and Times New Roman fonts use the same glyph shape for the LATIN CAPITAL LETTER A WITH RING, and the ANGSTROM SIGN. However, although it is not clear in this table, the Lucida San Unicode font includes different glyphs for each character.

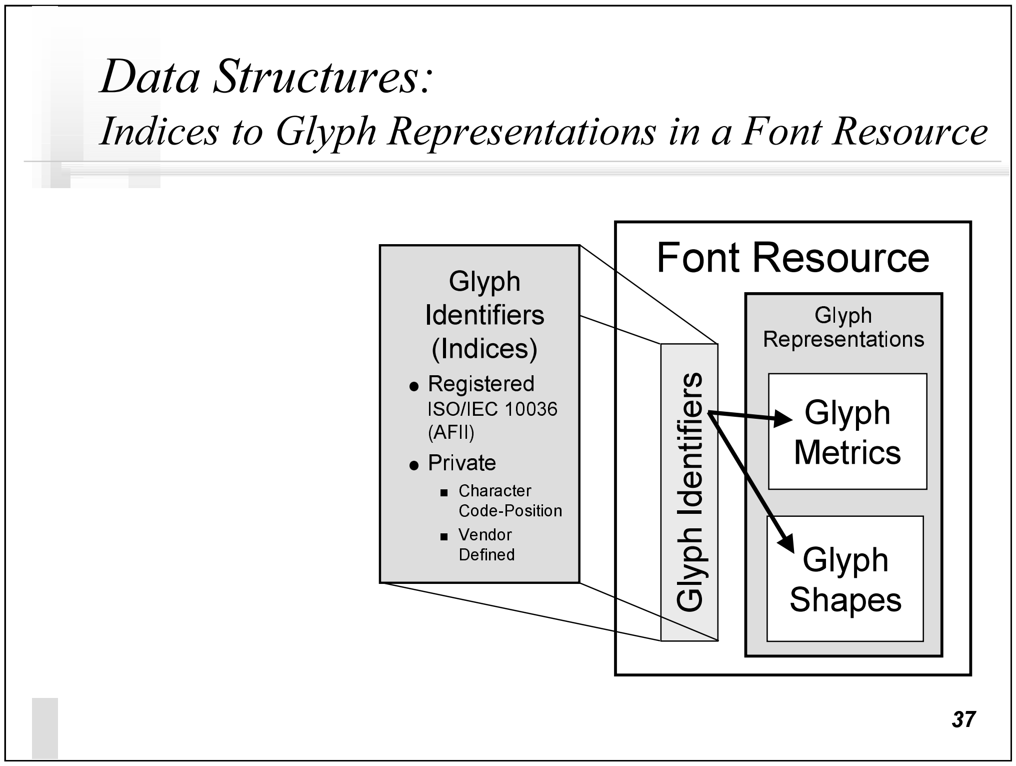
In the second row, the three fonts use the same shape and the same glyph metrics for the LATIN CAPITAL LETTER A, the GREEK CAPITAL LETTER ALPHA, and the CYRILLIC CAPITAL LETTER A. However, in the general case, the font may have different metrics that depend on the language or script.

For the third row, the various SPACE characters have the same (null) shape, but different metrics.

The fourth row illustrates different implementations of HYPHEN characters. Note that Lucida San Unicode uses a different shape.

The fifth row shows that spacing for the four basic arithmetic operators is the same.

The sixth and last row is a relatively new development where a single font will contain traditional and simplified Chinese, Japanese and Korean glyphs for the unified ideographic characters plus metrics for both horizontal and vertical layout.

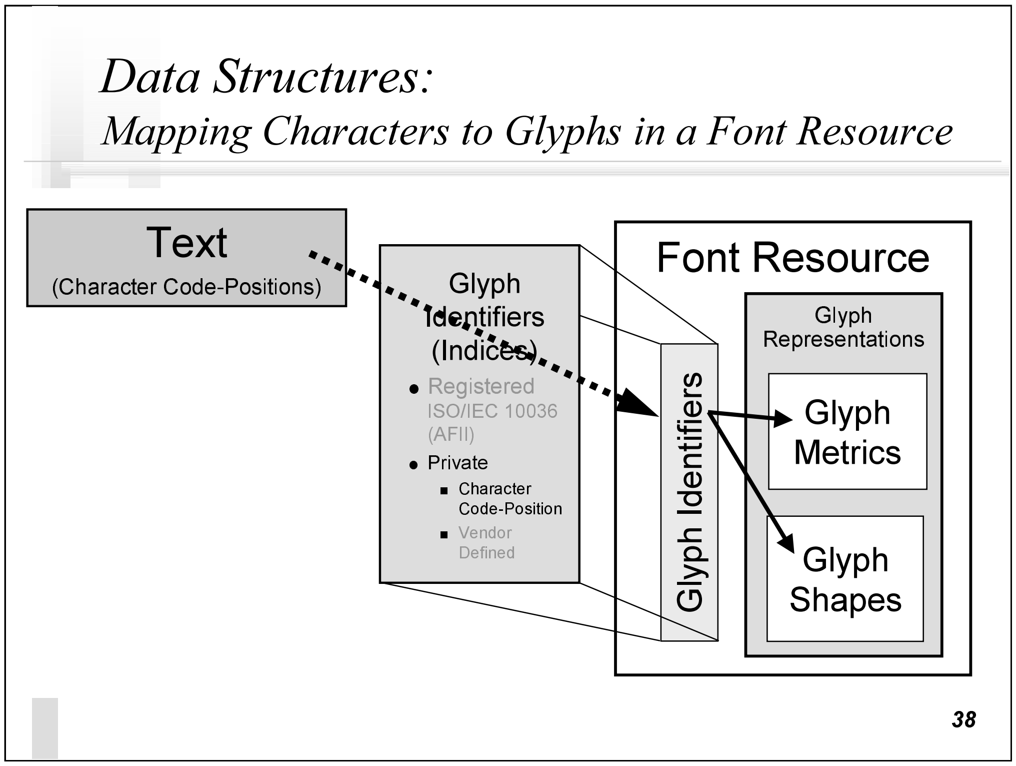


Data Structures:
Indices to Glyph Representations in a Font Resource

Glyph Identifiers (Indices)

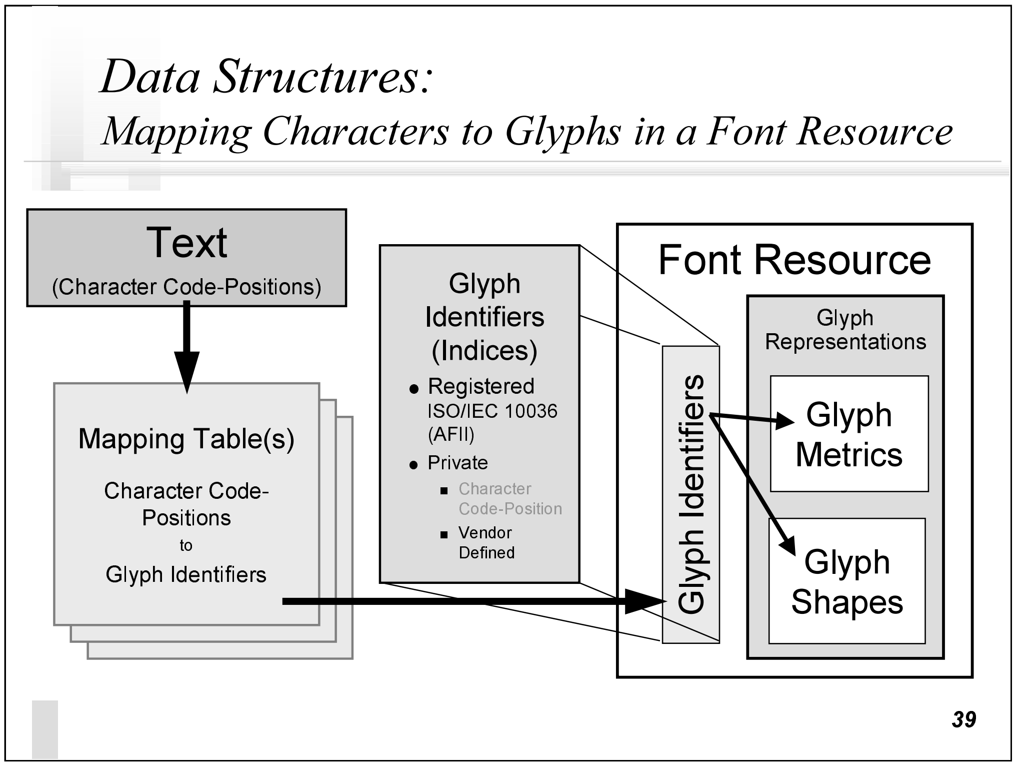
- Registered
  ISO/IEC 10036 (AFII)
- Private
  - Character Code-Position
  - Vendor Defined

This figure illustrates different types of glyph identifiers used to index into the Font Resource. A font designer can use the registered glyph identifiers from the AFII Registry or private glyph identifiers. Two types of private glyph identifiers are the character code position and vendor defined indices.



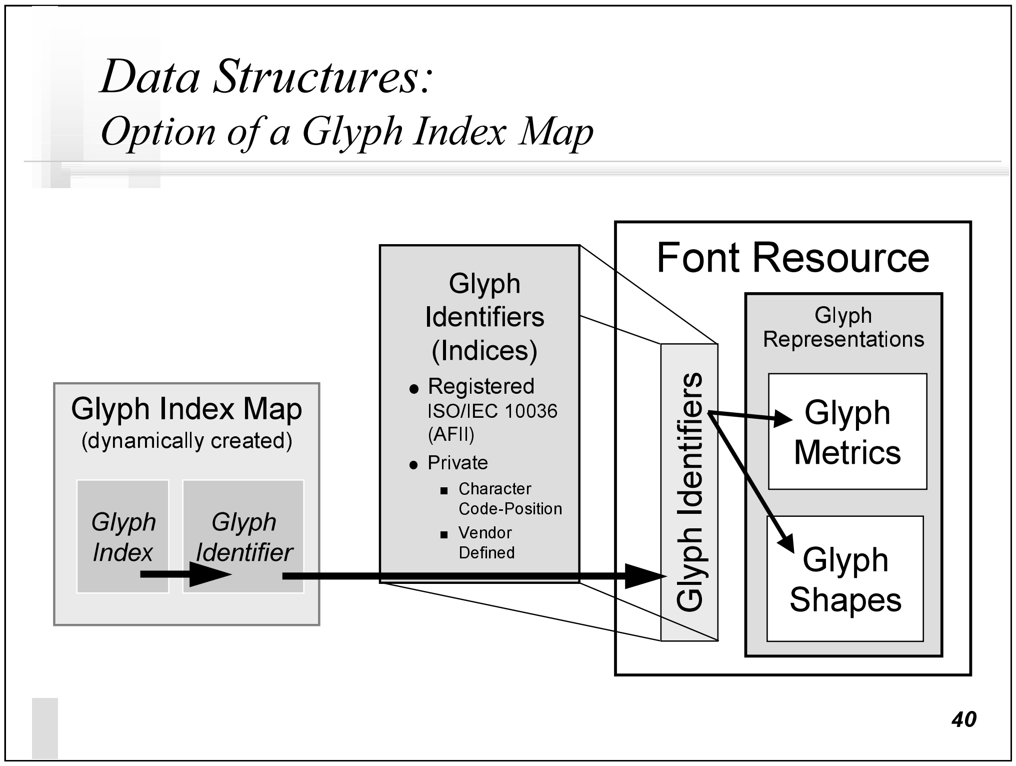
Data Structures:
Mapping Characters to Glyphs in a Font Resource

If the private, character code position is used for the glyph identifiers, glyph selection is one-to-one and consists of using the code position for a character to index into the font resource to obtain the glyph metrics and glyph shapes.



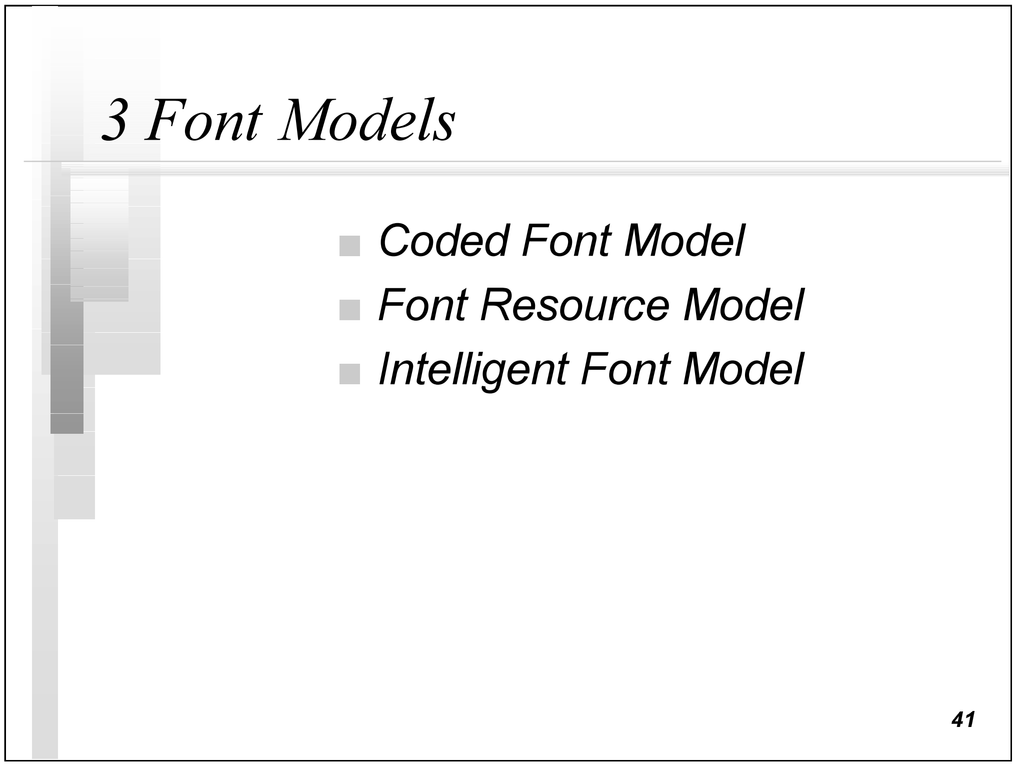
Data Structures:
Mapping Characters to Glyphs in a Font Resource

If glyph identifiers, either registered or vendor defined, are used, glyph selection requires tables to map from the character code position into the glyph identifier used to index into the font resource to obtain the glyph metrics and glyph shapes. The mapping tables may implement one-to-one mappings or more sophisticated *M*-to-*N* mappings.



Data Structures:
Option of a Glyph Index Map

An optional data structure is the "Glyph Index Map". This table is specific to a document and consists of entries for only those glyphs used in a particular document. The glyph index is used to obtain the glyph identifier (be it registered or private) to index into the Font Resource. The advantage of using this data structure is for large fonts, such as those that implement glyphs for large subsets of 10646 and Unicode. By using a Glyph Index Map, only the subset of glyph information for the glyphs actually used in the document need to be downloaded to render the document. This is valuable for printers with limited memory or for downloading pages from the WWW over slow telephone lines.

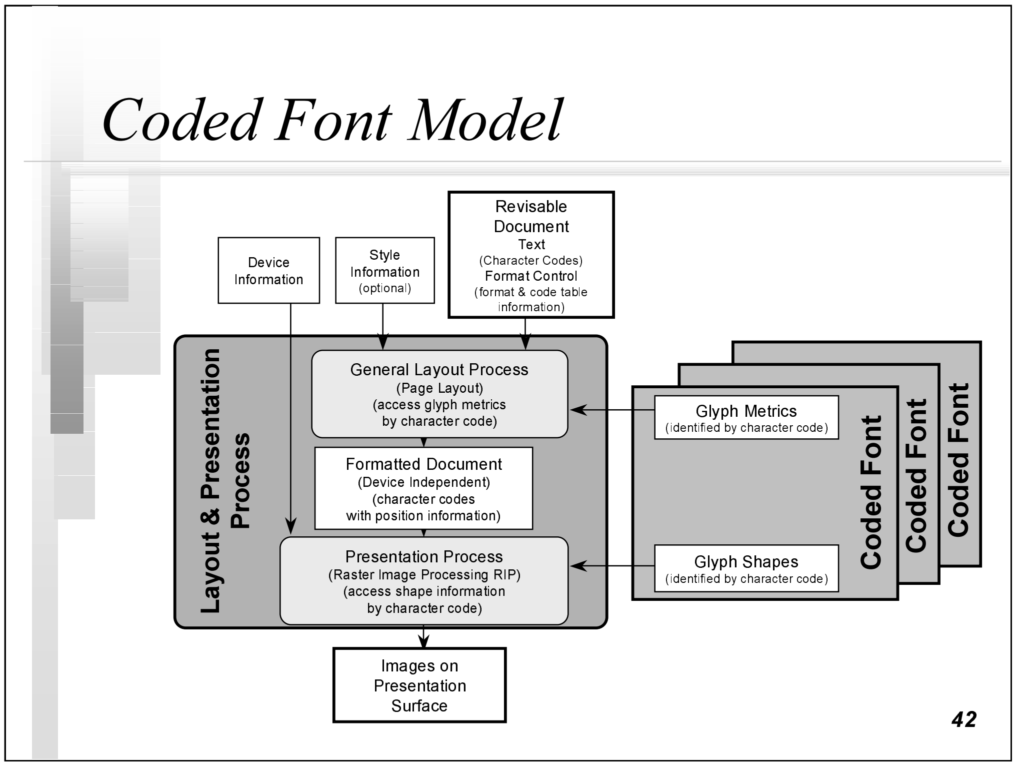


3 Font Models

- *Coded Font Model*
- *Font Resource Model*
- *Intelligent Font Model*

41

We will now turn to examine three font technologies to see how they use the various data structures to render characters into glyphs for display and printing.



Coded Font Model

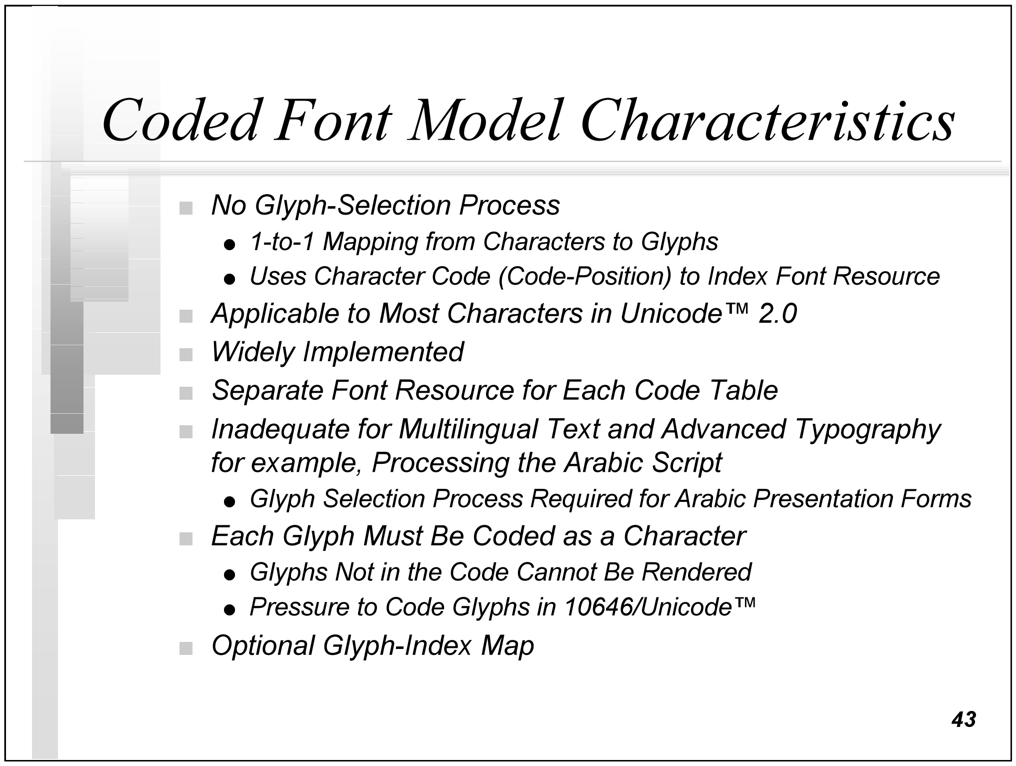
Revisable Document
Text
(Character Codes)
Format Control
(format & code table information)
Device Information
Style Information
(optional)
Layout & Presentation Process
General Layout Process
(Page Layout)
(access glyph metrics by character code)
Formatted Document
(Device Independent)
(character codes with position information)
Presentation Process
(Raster Image Processing RIP)
(access shape information by character code)
Images on Presentation Surface
Glyph Metrics
(identified by character code)
Glyph Shapes
(identified by character code)
Coded Font
Coded Font
Coded Font
42

A coded font (or a character-coded font) is a data structure in which character codes are used to index the glyph metric and glyph shape information contained in the font. The data structure depends on a one-to-one mapping of character codes to the glyphs in a font. Note that each code supported requires a separate coded font even if the glyphs are the same.

This font model is the historic presentation model for data processing. In this model, each character code encountered by the layout process is used to locate a corresponding glyph in the coded font. The glyph metric information for that character code is used to determine positioning of the glyph, along with line and page breaks. The formatted document may be interchanged to another location for presentation processing or transmitted to a local presentation process. The presentation process would use the character codes contained in the formatted document to locate a corresponding glyph in the coded font and use the associated glyph shape information to image the glyph on the presentation surface at the position indicated by the layout process.

With the coded font model, if a desired glyph is not associated with a character in a coded character set, then the glyph cannot be displayed or printed. This fact and the widespread implementation of the coded font model have resulted in pressure to include some glyphs in coded character sets. The other two font models do not require that all the glyphs in a font resource be coded as characters in the coded character set to print or display the glyphs.

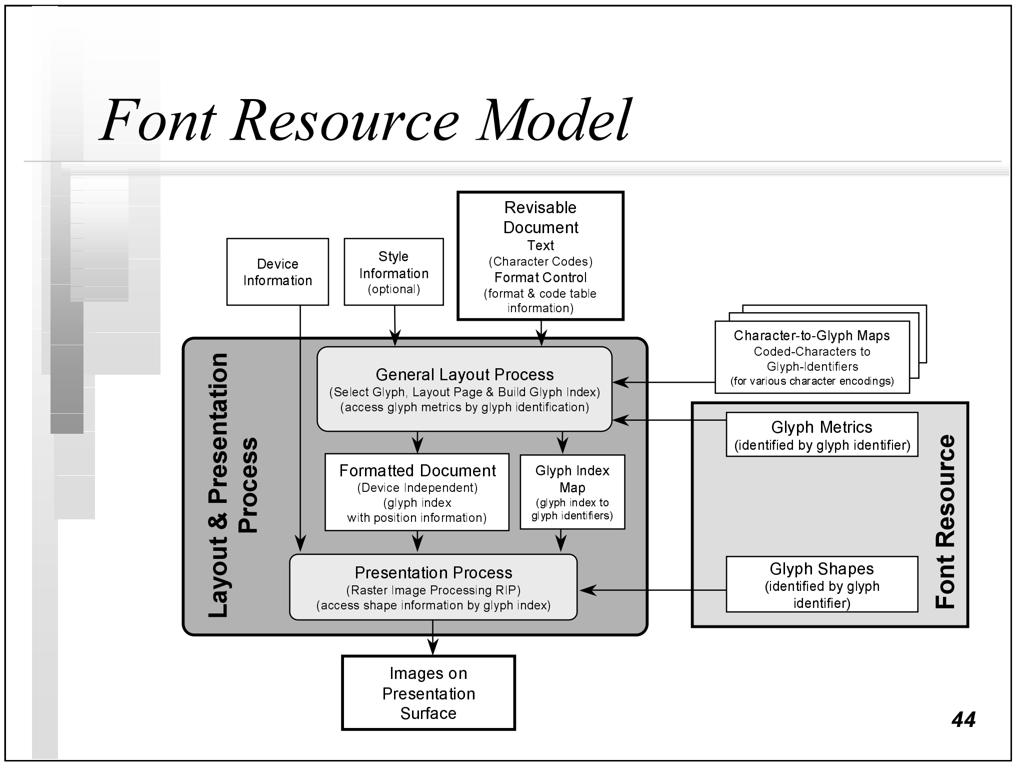
The coded font model is less suitable than the other two for the more complex glyph-selection requirements of printing and publishing. For example, the Arabic script requires special processing in the coded font model. If the input to the general layout process includes Arabic characters, the process also needs to convert the Arabic characters to the correct Arabic presentation forms, which is not normally part of this model.



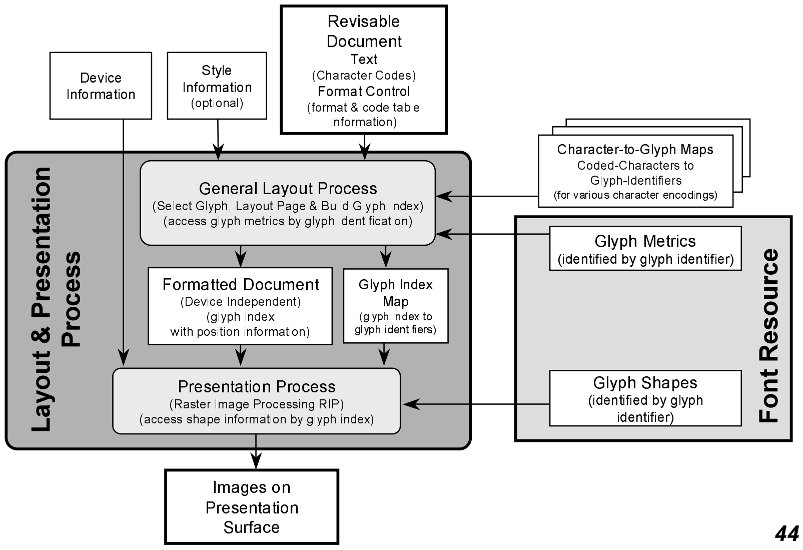
Coded Font Model Characteristics

- *No Glyph-Selection Process*
  - *1-to-1 Mapping from Characters to Glyphs*
  - *Uses Character Code (Code-Position) to Index Font Resource*
- *Applicable to Most Characters in Unicode™ 2.0*
- *Widely Implemented*
- *Separate Font Resource for Each Code Table*
- *Inadequate for Multilingual Text and Advanced Typography for example, Processing the Arabic Script*
  - *Glyph Selection Process Required for Arabic Presentation Forms*
- *Each Glyph Must Be Coded as a Character*
  - *Glyphs Not in the Code Cannot Be Rendered*
  - *Pressure to Code Glyphs in 10646/Unicode™*
- *Optional Glyph-Index Map*

43



Font Resource Model

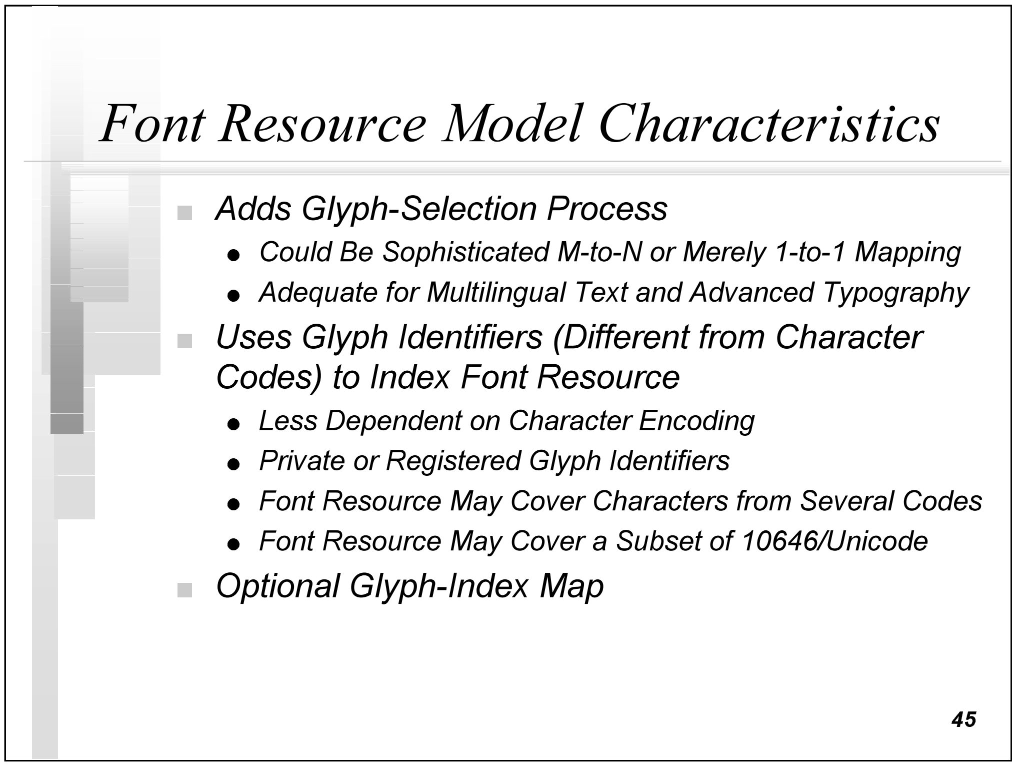

Revisable Document
Text (Character Codes)
Format Control (format & code table information)
Device Information
Style Information (optional)
Character-to-Glyph Maps
Coded-Characters to Glyph-Identifiers (for various character encodings)
Layout & Presentation Process
General Layout Process
(Select Glyph, Layout Page & Build Glyph Index)
(access glyph metrics by glyph identification)
Formatted Document (Device Independent) (glyph index with position information)
Glyph Index Map (glyph index to glyph identifiers)
Presentation Process
(Raster Image Processing RIP)
(access shape information by glyph index)
Font Resource
Glyph Metrics (identified by glyph identifier)
Glyph Shapes (identified by glyph identifier)
Images on Presentation Surface

44

The font resource model permits definition of font resources that are less dependent on any single coded character set or document-processing model. This model is more suited to the document printing and publishing environment. Glyph identifiers index the glyph metrics and glyph shape representations in the font resource. In this model, the layout process uses predefined character-to-glyph maps to determine the mapping of character codes to presentation glyphs and replaces the character codes in the formatted document with glyph index values.

Optionally, a composition and layout process may generate a glyph index map that accesses only and exactly those glyphs of a large font resource that are needed to image the output of the process.

In the font resource model, the relationship between the character repertoire and the glyph collection may involve a one-to-one mapping but may also involve a one-to-many or many-to-one mapping. It is essential for successful presentation that the set of glyphs in the glyph collection be mappable to the repertoire of characters used in the text string.

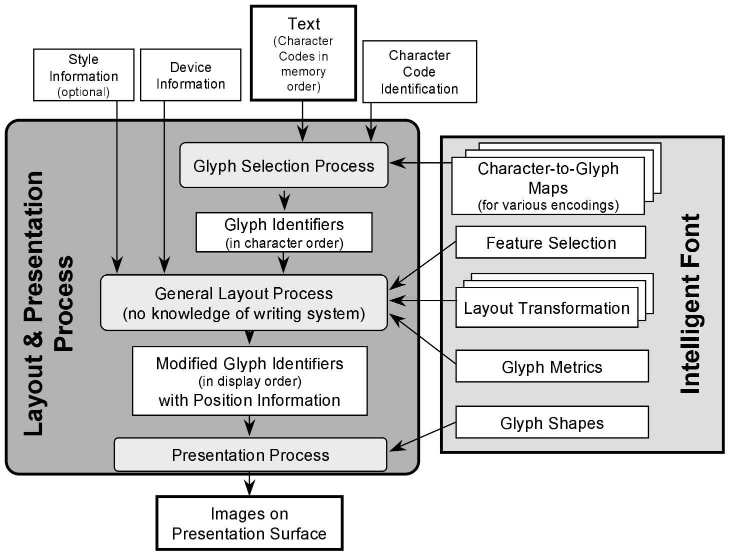


Font Resource Model Characteristics

- *Adds Glyph-Selection Process*
  - *Could Be Sophisticated M-to-N or Merely 1-to-1 Mapping*
  - *Adequate for Multilingual Text and Advanced Typography*
- *Uses Glyph Identifiers (Different from Character Codes) to Index Font Resource*
  - *Less Dependent on Character Encoding*
  - *Private or Registered Glyph Identifiers*
  - *Font Resource May Cover Characters from Several Codes*
  - *Font Resource May Cover a Subset of 10646/Unicode*
- *Optional Glyph-Index Map*

45

Intelligent Font Model


Text (Character Codes in memory order)
Style Information (optional)
Device Information
Character Code Identification
Layout & Presentation Process
Glyph Selection Process
Glyph Identifiers (in character order)
General Layout Process (no knowledge of writing system)
Modified Glyph Identifiers (in display order) with Position Information
Presentation Process
Images on Presentation Surface
Intelligent Font
Character-to-Glyph Maps (for various encodings)
Feature Selection
Layout Transformation
Glyph Metrics
Glyph Shapes

46

An intelligent font is a data structure that augments a font resource with additional information describing

- how a sequence of coded characters is transformed into a sequence of glyph identifiers, with associated position information
- how the transformation of coded characters to glyph identifiers is affected by style information

The first set of additional information typically includes several mappings from various coded character sets to private (font-specific) glyph identifiers. Subsequent transformations use the glyph identifiers. The subsequent transformations may be complex and may result in changes to the number and ordering of the glyph identifiers. The second type of additional information permits, for example, substitution of glyph subsets (for example, swash variants, vertical substitution) based on style information.

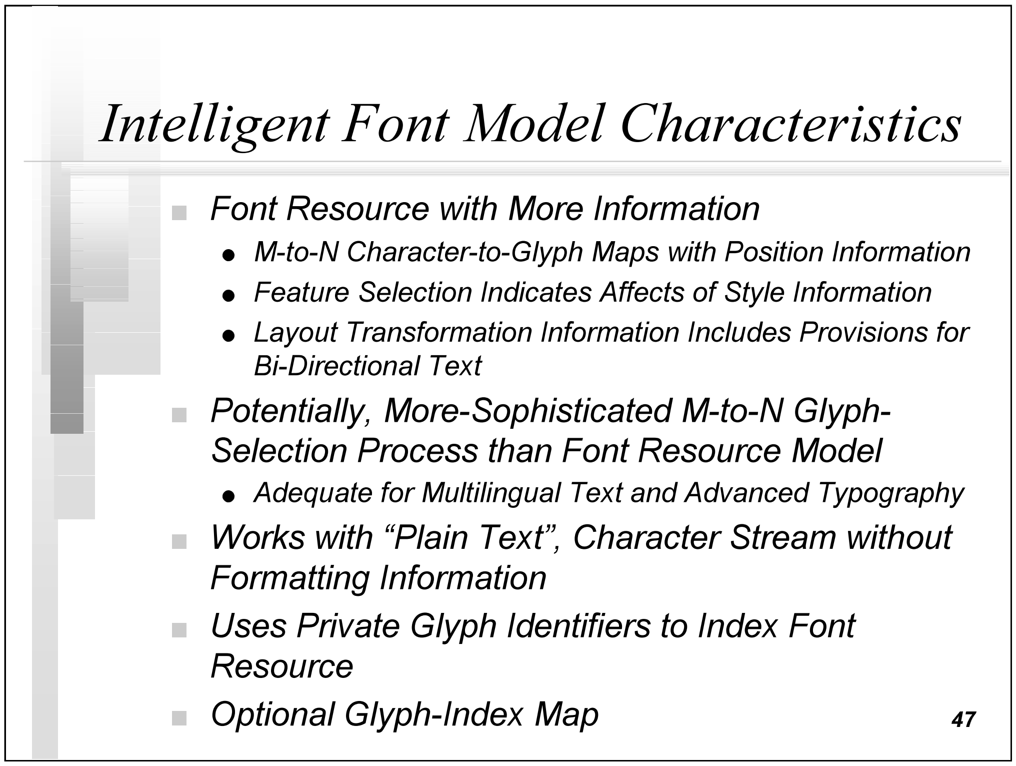
Within the layout and presentation process of the intelligent font model, the glyph selection process transforms coded characters to glyph identifiers. This process requires

- information about how the characters are coded
- the map from coded characters to glyph identifiers for the specified character coding

The process takes coded characters in memory or logical order and produces glyph identifiers in character or logical order. Logical order is the order in which a person would normally read the characters regardless of the normal direction of the characters. For Latin text included in the middle of Arabic text, the logical order would be the rightmost Arabic character to the end of the Arabic text, then the leftmost Latin character to the end of the Latin text, and then the rightmost Arabic character of the second group of Arabic text to the end of the Arabic text.

Next, the general layout process transforms the glyph identifiers in logical order into (possibly modified) glyph identifiers in display order. Display order is the order in which the characters are to appear on paper or on a screen.

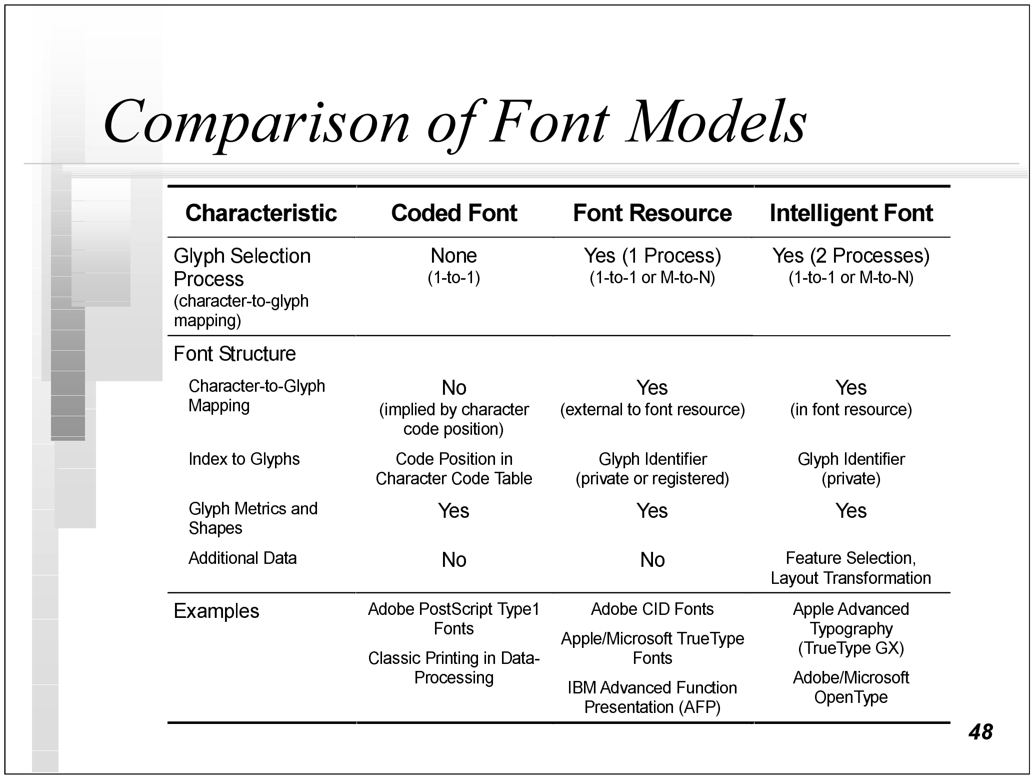
The presentation process is the final process. It takes the glyph identifiers in display order, the glyph positions, and the glyph shapes to produce the images on paper or a screen



Intelligent Font Model Characteristics

- *Font Resource with More Information*
  - *M-to-N Character-to-Glyph Maps with Position Information*
  - *Feature Selection Indicates Affects of Style Information*
  - *Layout Transformation Information Includes Provisions for Bi-Directional Text*
- *Potentially, More-Sophisticated M-to-N Glyph-Selection Process than Font Resource Model*
  - *Adequate for Multilingual Text and Advanced Typography*
- *Works with “Plain Text”, Character Stream without Formatting Information*
- *Uses Private Glyph Identifiers to Index Font Resource*
- *Optional Glyph-Index Map*

47



Comparison of Font Models

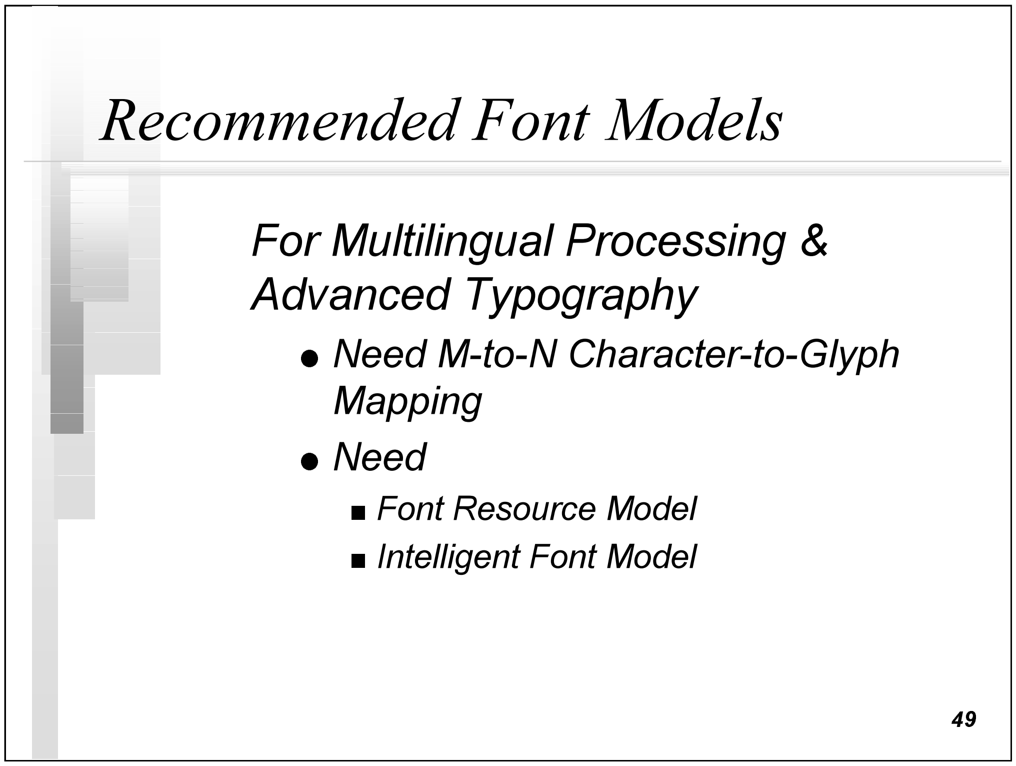
Characteristic	Coded Font	Font Resource	Intelligent Font
Glyph Selection Process (character-to-glyph mapping)	None (1-to-1)	Yes (1 Process) (1-to-1 or M-to-N)	Yes (2 Processes) (1-to-1 or M-to-N)
Font Structure			
Character-to-Glyph Mapping	No (implied by character code position)	Yes (external to font resource)	Yes (in font resource)
Index to Glyphs	Code Position in Character Code Table	Glyph Identifier (private or registered)	Glyph Identifier (private)
Glyph Metrics and Shapes	Yes	Yes	Yes
Additional Data	No	No	Feature Selection, Layout Transformation
Examples	Adobe PostScript Type1 Fonts Classic Printing in Data-Processing	Adobe CID Fonts Apple/Microsoft TrueType Fonts IBM Advanced Function Presentation (AFP)	Apple Advanced Typography (TrueType GX) Adobe/Microsoft OpenType

48

The principal differences between the font models are:

- the presence of a glyph selection process and the potential sophistication of that process
- how the glyphs are indexed in the font resource
- the presence of additional data in the font resource

Examples of the Coded Font are Adobe PostScript Type 1 fonts and classic printing in data processing. Examples of the Font Resource are Adobe CID fonts, Apple/Microsoft TrueType fonts, and IBM Advanced Function Presentation (APF). Examples of Intelligent fonts are Apple Advanced Typography (TrueType GX), and Adobe/Microsoft OpenType.



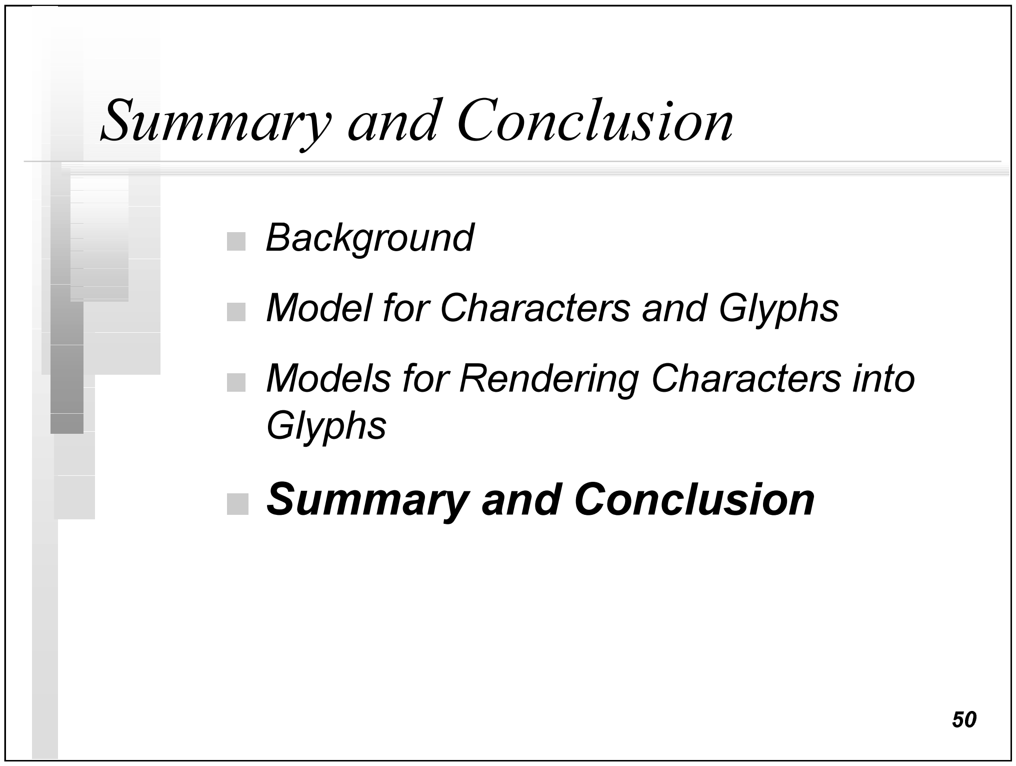
Recommended Font Models

For Multilingual Processing & Advanced Typography

- *Need M-to-N Character-to-Glyph Mapping*
- *Need*
  - *Font Resource Model*
  - *Intelligent Font Model*

49

For multilingual and advanced typography, the more sophisticated *M*-to-*N* glyph selection is required. Potentially, both the Font Resource Model and the Intelligent Font have this capability. However, frequently this potential is unrealized with the Font Resource Model because only a one-to-one glyph selection is implemented.

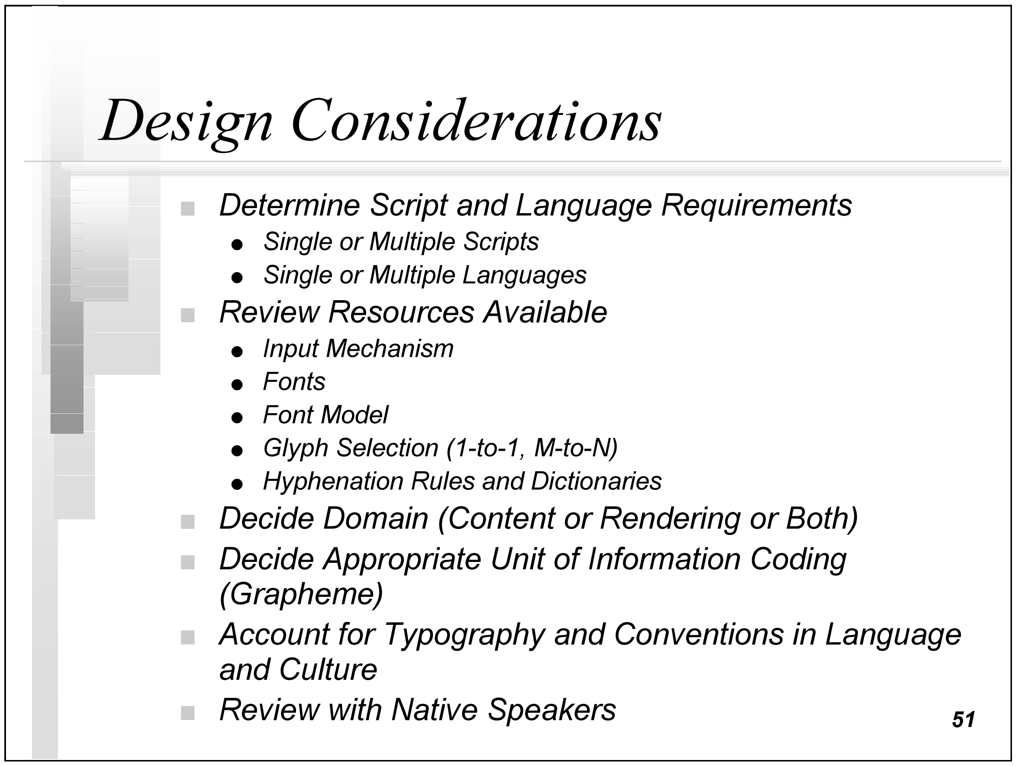


Summary and Conclusion

- *Background*
- *Model for Characters and Glyphs*
- *Models for Rendering Characters into Glyphs*
- ***Summary and Conclusion***

50

At this point, let me try to draw some conclusions and summarize the presentation.



Design Considerations

- *Determine Script and Language Requirements*
  - *Single or Multiple Scripts*
  - *Single or Multiple Languages*
- *Review Resources Available*
  - *Input Mechanism*
  - *Fonts*
  - *Font Model*
  - *Glyph Selection (1-to-1, M-to-N)*
  - *Hyphenation Rules and Dictionaries*
- *Decide Domain (Content or Rendering or Both)*
- *Decide Appropriate Unit of Information Coding (Grapheme)*
- *Account for Typography and Conventions in Language and Culture*
- *Review with Native Speakers*

51

Let me first attempt to summarize how a developer might use the information presented by listing some design considerations.

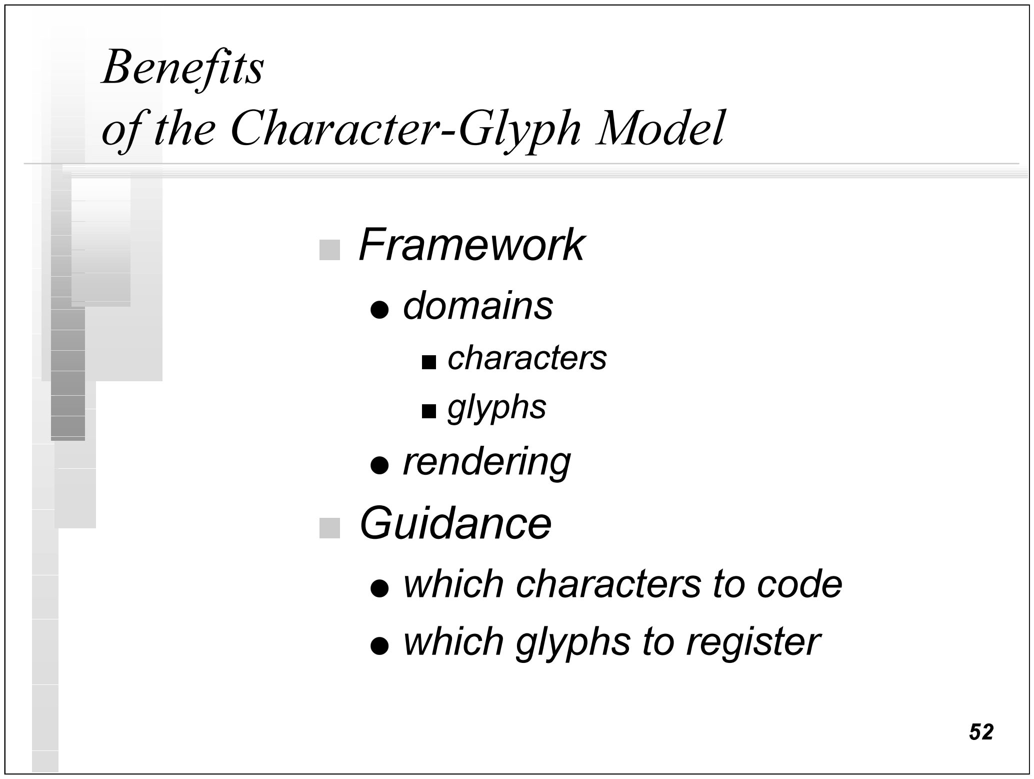
First, determine your script and language requirements. Are you working with one or multiple scripts, or one or multiple languages in your product?

Second, review the resources available on the platforms on which you are building your products. Does it have an input mechanism for the script/languages of your product? Are fonts available? Which font model is used for rendering? Does it provide one-to-one or *M*-to-*N* glyph selection? Are dictionaries with hyphenation rules available?

Third, decide the domain or domains required for your product. Is it in the content domain, or the presentation domain, or both? Based on your answer, consider if you should recode the information to optimize it to your processing.

Fourth, consider the typography and conventions used in the language and culture. This will have sometime subtle but import differences between cultures.

Finally, have native speakers review your product before you ship it. They will find problems that the "experts" miss and which may turn your product into a failure.



Benefits of the Character-Glyph Model

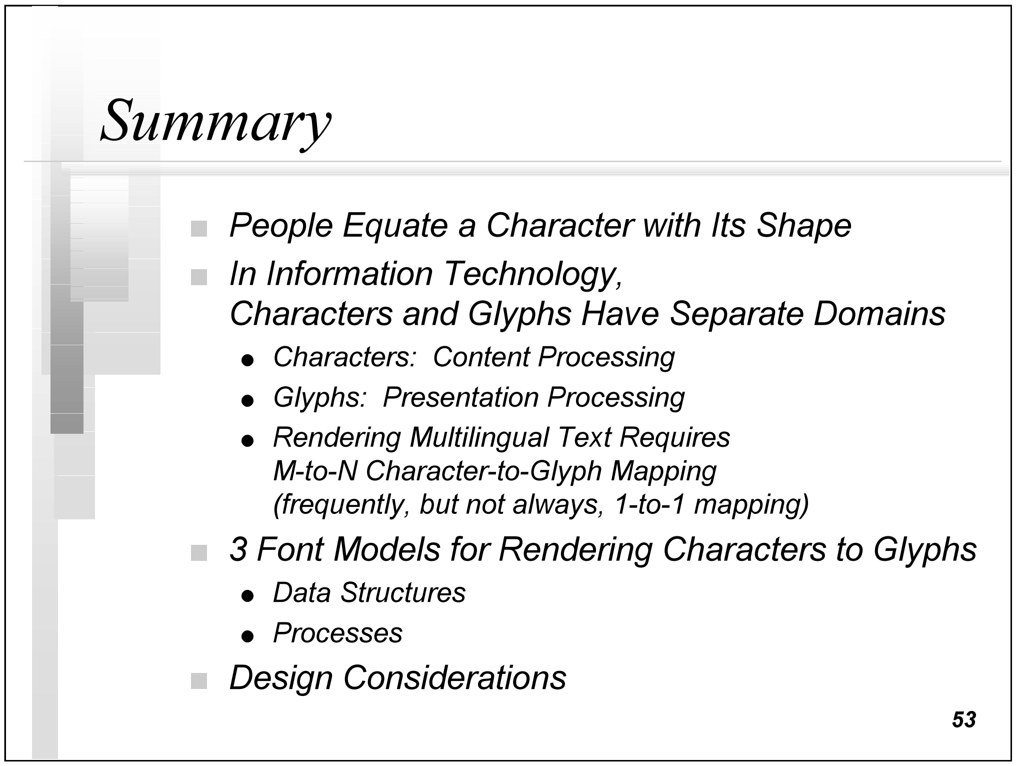
- *Framework*
  - *domains*
    - *characters*
    - *glyphs*
  - *rendering*
- *Guidance*
  - *which characters to code*
  - *which glyphs to register*

52

What are the benefits of the Character-Glyph model that was just described?

First, it provides a framework to developers and the standards committees to characterize the character domain and the glyph domain and to describe the glyph-selection process and models for rendering characters into glyphs for presentation.

Second, it provides guidance to the standards committees about whether to code an entity as a character or to register it as a glyph. Note that I said "guidance" because frequently other considerations besides the idealized principles presented here become important in the decision process.



Summary

- *People Equate a Character with Its Shape*
- *In Information Technology,*
  Characters and Glyphs Have Separate Domains
  - *Characters: Content Processing*
  - *Glyphs: Presentation Processing*
  - *Rendering Multilingual Text Requires*
    M-to-N Character-to-Glyph Mapping
    (frequently, but not always, 1-to-1 mapping)
- *3 Font Models for Rendering Characters to Glyphs*
  - *Data Structures*
  - *Processes*
- *Design Considerations*

53

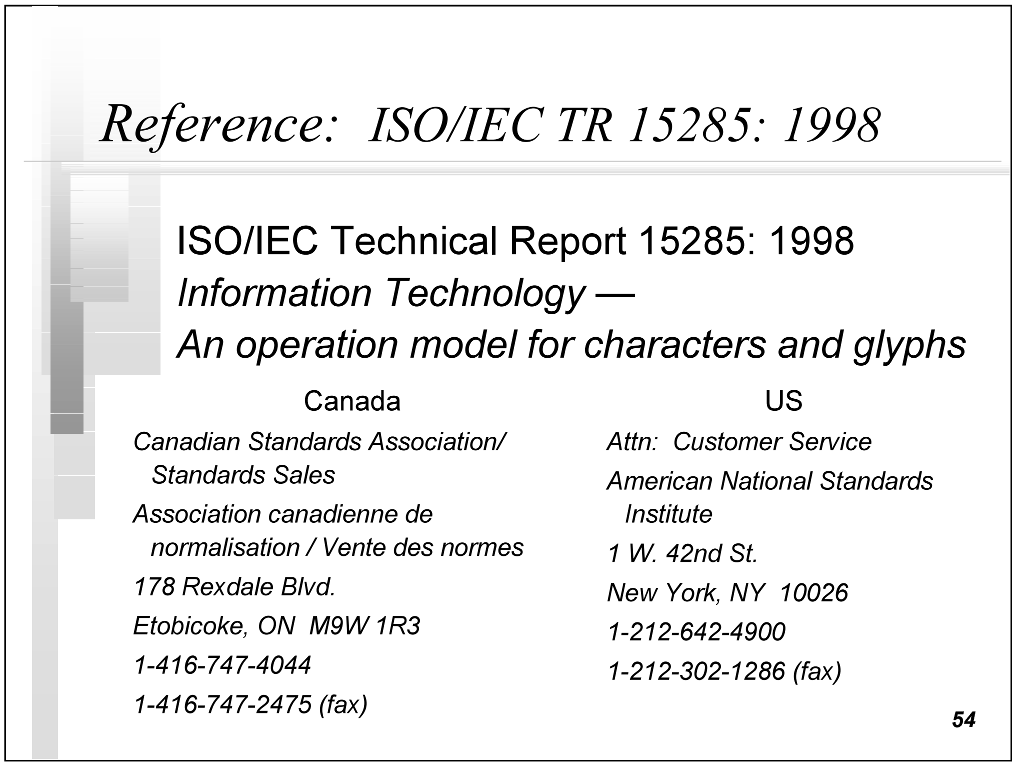
Let me conclude the presentation with a summary.

People equate a character with its shape. To a person, they are inseparable.

However, in the world of Information Technology, we have decided to distinguish between the character attribute of its information content, which we code as characters, and the attribute of its shape, which we call a glyph. We divide these into a character domain and a glyph domain. Converting from characters in the character domain to glyphs in the glyph domain requires a glyph selection process. The glyph selection process is frequently a one-to-one mapping but sometimes requires a more complex *M*-to-*N* mapping. In general, rendering multilingual text requires an *M*-to-*N* mapping.

Three technologies, or three font models, are available for rendering characters into glyphs. Each uses similar data structures and processes to render text. The presentation compared the font models and recommended two to use for multilingual rendering.

Finally, we listed several considerations for the developer of multilingual products.



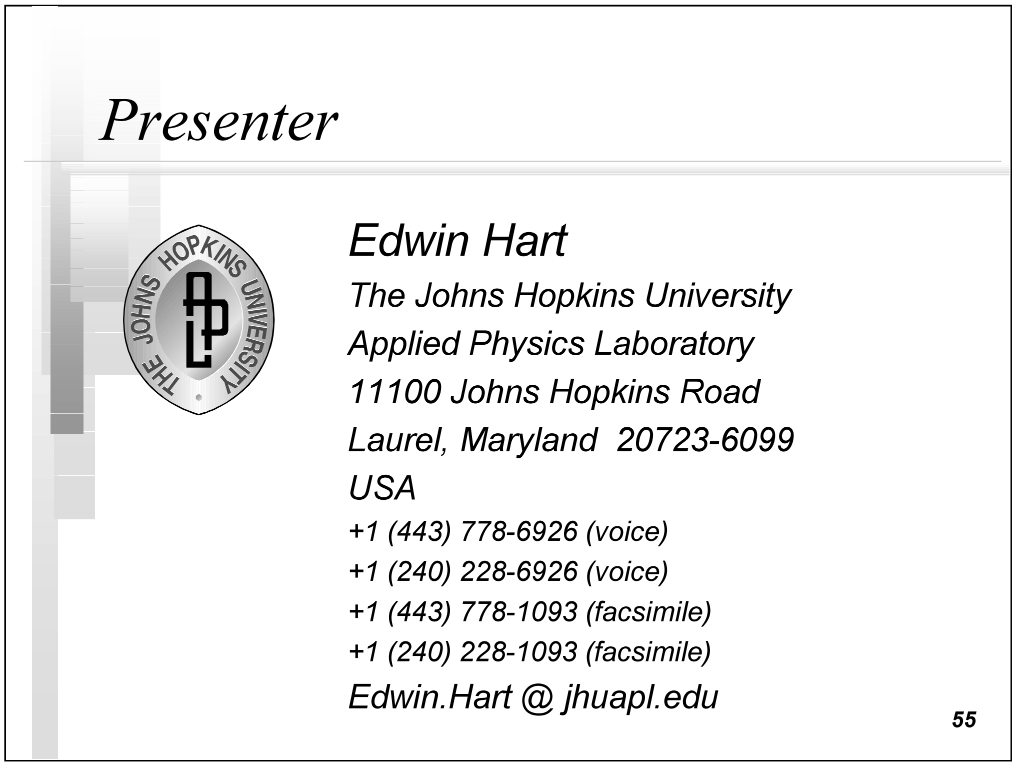
Reference: ISO/IEC TR 15285: 1998

ISO/IEC Technical Report 15285: 1998
Information Technology —
An operation model for characters and glyphs

Canada	US
Canadian Standards Association/ Standards Sales	*Attn: Customer Service*
Association canadienne de normalisation / Vente des normes	*American National Standards Institute*
178 Rexdale Blvd.	*1 W. 42nd St.*
Etobicoke, ON M9W 1R3	*New York, NY 10026*
1-416-747-4044	*1-212-642-4900*
1-416-747-2475 (fax)	*1-212-302-1286 (fax)*

54

You can obtain a copy of the ISO Technical Report on which this presentation is based from your national standards organization. In Canada, it is the Canadian Standards Association (CSA), or in French, Association canadienne de normalisation; in the US, it is the American National Standards Institute (ANSI). The figure shows the addresses and phone numbers for each organization.



Presenter

Edwin Hart

The Johns Hopkins University

Applied Physics Laboratory

11100 Johns Hopkins Road

Laurel, Maryland 20723-6099

USA

+1 (443) 778-6926 (voice)

+1 (240) 228-6926 (voice)

+1 (443) 778-1093 (facsimile)

+1 (240) 228-1093 (facsimile)

Edwin.Hart @ jhuapl.edu

55