NAPGEN - AUTOMATIC NAPLPS PAGE GENERATOR

Michel Plante
Research & Development Group, Industrial Technical Services
Montréal, Québec H5B 1B3

ABSTRACT

The creation of videotex pages from already encoded information (generally in the form of ASCII or EBCDIC codes) involves most of the time a large amount of work from the people doing it and usually they are bound to simply retype the whole information in the new videotex page using some page creation equipment (hardware and/or software). What is needed then is a simple tool that would automate the process of inserting this information (from an ASCII database) into the final videotex page without having to redo all the encoding by hand.

The following paper deals with the issue and explains NAPGEN (NAPLPS Page GENerator), the application program that was written specifically to perform the conversion function from ASCII to NAPLPS with the addition of graphics made possible by the NAPLPS videotex standard itself.

The paper starts by explaining exactly the purpose of this tool and our goals in making it, and we then follow with a description of the project itself, NAPGEN.

RESUME

La création de pages vidéotex à partir d'information déjà encodée (sous format ASCII ou EBCDIC) requiert la plupart du temps une assez grande quantité de travail des gens effectuant la conversion et presque tout le temps ils en sont contraint a simplement retaper toute cette information dans la nouvelle page vidéotex en utilisant un équipement de création de pages vidéotex. Le besoin serait donc d'essayer d'automatiser le procédé d'insertion de ces informations (provenant de banques de données ASCII) dans les pages finales vidéotex sans avoir à les retaper manuellement.

L'article qui suit adresse ce problème et explique le fonctionnement de NAPGEN (NAPLPS Page GENerator), le programme d'application qui a été conçu spécifiquement pour la conversion d'informations ASCII en pages NAPLPS avec l'ajout, naturellement, d'objets graphiques rendus disponibles par le standard vidéotex NAPLPS lui-même.

L'article débute avec une explication du pourquoi de cet outil et de nos buts en le construisant et se poursuit ensuite avec une description du projet lui-même, NAPGEN.

INTRODUCTION

People in the computer industry already have an incredible amount of information stored on magnetic tape or on disc or on whatever media you can think of. The information is there, stored somewhere and awaits to be examined. But each information has been stored in some particular way and to retrieve it requires some skill. And often enough the presentation of the data on the screen is terse and can be difficult to interpret. From this we realized that it would be nice to be able to take parts of these existing databases, any database, and to combine them with some graphics elements into one entity, a NAPLPS videotex page.

The NAPLPS standard (North American Presentation Level Protocol Syntax) is powerful in that it fully specifies how to encode graphics and text strings for an eventual exchange of this information in between computers.

But the NAPLPS standard is only a tool. To be able to communicate videotex information we suppose that there must be at least three communicating partners: the database manager, the one who owns the database and maintains it, the information provider, who puts some of his information in the database in the form of NAPLPS pages and finally the information receiver, the end-user who examines the final pages of information.

For those information providers that do not require that their information be updated too often (one shot advertising for instance), the cost of creating a few NAPLPS pages of information is very reasonable. But most of them will in fact require that their information be updated on a monthly, weekly or even daily basis. But as we know the generation of any kind of coded information , (NAPLPS pages or any encoded data , ASCII, EBCDIC ...etc) i.e. "putting it in the computer" is very time consuming. And the time it requires to maintain even the smallest amount of pages up to date, becomes rapidly too prohibitive for any information provider and therefore too costly.

Ideally, the database manager would like to offer to its information providers a very time and cost effective way to maintain all those pages of information alive and well without having to invest in any kind of page creation equipment.

OUR GOALS

NAPGEN was developed to provide the database manager with a tool to automatically produce NAPLPS pages and our basic goals for this project was twofold :

To be able to convert the already stored data from plain ASCII code to NAPLPS code.

Once these new NAPLPS pages have been created we also need a tool that will assist us in maintaining the data presented in them up to date.

NAPGEN

What we have developed is a way to interface every possible database to our application program, regardless of the way it was stored. To do it NAPGEN needs basically four different information that will accordingly be stored in four separate files as follows:

The interface file (IF), contains the data itself, in the form of ASCII text, to be copied into the videotex page.

The Field Description File (FDF), contains the information on how the data was stored (the format) in the IF.

The Data Description File (DCF), contains the information on how the data will be presented in the final videotex page. It is itself a NAPLPS page.

And finally, the Graphics Element File (GEF), contains different graphics elements (objects) stored in the form of NAPLPS codes.

The interface file (IF) will make the connection between our application program and the database in question. The IF will be provided by the information provider and involves very little effort to create. The IF is a simple sequential, variable length record file that contains a header record and the data itself that the information provider wants to insert in his NAPLPS pages. The header record contains among other things the names of the DCF, FDF and GEF files.

The work involved then for the information provider is the creation of the IF from his existing ASCII database. It is simply a matter of reformatting the stored information in his database (unknown format to NAPGEN) to a known format in the IF that NAPGEN will understand.

The known format of the IF is entered in a second file, the FDF, the Field Description File. The FDF will contain, among other things, the exact starting position of each field in an IF record, its length and its type. At this point NAPGEN knows exactly where the data is located within the IF but it still does not know where to place it in the final NAPLPS page.

We need a third file, called DCF, the Data Characteristic File. This file is a NAPLPS created on a page-creation equipment that contains descriptors in the form of text strings at the exact position in the page where the information provider will want his data to be located. Not only do the descriptors define the place where the data is going to be placed, but they also define the attributes the data will have.

Let´s say the information provider wants such and such data to appear on the screen as a double size, blue and blinking character he may do so by simply creating his descriptor as being a double sized, blue and blinking character on the page creation equipment.

The DCF is really a template that will be used by NAPGEN to create the final NAPLPS page. This template gives the user a very efficient way to easily change the attributes of his data by just changing the attribute of the descriptor of the desired data.

Since NAPLPS provides very powerful graphics capabilities it is obvious that we would like to add a few graphics elements to the final picture. This is also easily done by specifying the names of graphics elements that the user wants to see included in his page. The names of these graphics elements are really the names of the GEF.

Graphics elements are also created on a page-creation package. They are by themselves NAPLPS pages with one or more graphics object stored in each one of them. Then, when the information provider accumulates a sufficient number of these graphics elements, he can easily combine them as he wishes to create more or less complex images, just by specifying to NAPGEN the filenames of the graphic elements he wants to see in his final NAPLPS page.

It is allowed however to use only one or even no graphics element at all in the final videotex page. For instance the user might be interested in simply generating a page of text with a dark background. Then the output of any text editor or word processor (without control characters) could be fed directly into NAPGEN to turn this purely ASCII text into a NAPLPS page, readable by any NAPLPS decoder.

Once NAPGEN is properly set up, benchmarks here at I.S.T have shown it possible to create up to one hundred NAPLPS pages, with graphics elements and data, in less than 2.5 minutes on a DEC VAX 11/750 running VMS.

It always involves some effort at first from the part of the information provider who wants to make use of NAPGEN. The first few template pages needed probably takes longer to create with NAPGEN than with the usual page-creation package. But once these "templates" or "models" have been created then the product becomes very attractive in that it allows the information provider to rapidly create new images or to update existing ones.

REFERENCE

Videotex/Teletext Presentation Level Protocol Syntax (North American PLPS).
ANSI Standard X3.110-1983/CSA Standard T500-1983; December 1983.