In 1993, the University of Chicago library and the ARTFL project began a multi-year collaboration which resulted in the formation of something that came to be called electronic text services, which has two components. The first, electronic full-text services, is based on technology which allows us to assimilate texts marked up in sgml and index them in what is now called PhiloLogic, the new ARTFL search engine. The second component, electronic open stacks, or EOS, is based on technology which allows us to create digital books consisting of scanned page images. It is this second component which this brief presentation will describe.

First, let me show you an example of what such a digital book looks like, to provide some context for a description of their page-turning mechanism.

e.uchicago.edu/h/eos
browse the sanctuary of artemis orthia at sparta
apollo tetraxeir
299
second line of inscription

The question is, how does this work? When you select a call number for an EOS book from the browse list, a cgi-bin script dynamically queries a relational database to assemble a digital table of contents. The relational database we use is called MYSQL. it is free, commercial, fast and robust. It is easy to install and maintain, and easy to use. ARTFL is making plans to use it for the Dublin Core metadata it is creating for the texts it's indexing under PhiloLogic.

This database implementation consists of two tables. One is called pages; the other bibliography. Bibliography is actually something of a misnomer, because though the bibliography table might originally have been conceived of as such functionally, by the person who originally designed and programmed this system, actually it contains both rudimentary administrative and descriptive metadata. There is room for improvement in this table structure, needless to say. Doing this is on our project list. One of the entries in this table is linked to another entry in the pages table. The pages table contains an entry for every page in every eos book. Again, there is room for improvement here, because right now the table contains almost 10,000 rows, representing 37 books. We need to break this up. It was done this way for the convenience of building the system initially.

There is an entry in the pages table for each page of each digital book. some of these are designated as milestones [=feature codes]. I did not invent this terminology, but I keep using it because it's simple to say. It is the milestones which you see displayed when you click on a call number for a digital book. It is the rows designated as such for this book which the cgi-bin script pulls from the database.

What links the two tables is the call number for each book. The call number is also what physically on disk identifies each component of each digital object. That is, each page of each book consists of a filename which begins with the call number of the book we digitized. The reasoning behind this system

is that the filename itself is the crudest form of structural metadata for the digital object as a whole. Even if we lost the eos database for whatever reason, and it is the page table in it which in fact contains the structural metadata, we could still use the call number associated with the digital object's filename to look up the paper copy in the online catalog and figure out what it was. Again, I realize this is crude, but so was the first wheel in all likelihood. In its defense, I will say that cataloging practice at our library, and I don't think we're unique in this, is to mark the verso of the t.p. with a book's call number, and this practice is about akin to that, except we do it for each page.

We create page identifiers for each page at the point of scan. These consist of its call number and unique page object number. Spaces are represented by underscores. So an entry could look like this:

PJ1553.A1_1908_cop3_82.JPG

All the images go into a directory named after the call number. Then a little perl script reads the contents of the CALL_NUMBER directory and generates a [CALL_NUMBER.idx_]template file for editing. This template is then edited to include milestones and page identifiers--these are the actual page numbers from the printed text, which allowed me to navigate the scanned images in the demo as if I were using a printed book. This template file is tab-delimited. It's on our project list to change this procedure so that someone who's never heard of a tab-delimited file but who is otherwise competent to scan a book, can do so and still create these metadata. When this editing has been completed, another perl scripts creates a file ready to be uploaded into the page table of the EOS database. An interactive perl script is then run to create the information to be added to the bibliography database. The scanning tech we've employed so far then does the last two steps: uploading each file into the corresponding table of the EOS database. This person clearly has update permissions in the db. A more ramped-up version of this approach will limit the damage that could be caused by an inexperienced operator. Still the advantage of the system so far is that, once it has been designed and installed, it can be used routinely by the department doing the scanning, in our case, the preservation department.