WHEAT Programs to Import and Export XY Chains

Gregory W Pouch

Abstract

The Windows-based Hydrogeologic Exploration and Appraisal Toolkit (WHEAT) is a set of user-friendly programs for the retrieval, analysis, manipulation and display of information on natural resources, especially geological resources. The principle goals in development of WHEAT have been: ease of use; hardware independence; applicability to resource management problems; end-user extendibility; and the ability to exchange information with other analysis packages. This document fulfills the goals of end-user extendibility and strengthens WHEAT's ability to exchange information by explaining how to use several WHEAT programs for importing, exporting and manipulating XY Chains of coordinates.

This document is intended for users who wish to construct a WHEAT database from available map coordinate information or wish to export data from a WHEAT database to some other format. Typical users would be geologists, surveyors, or foresters. It assumes the user is familiar with map coordinates, Microsoft Windows, WHEAT, and Microsoft Access. Although these programs can be used without MSAccess, the author strongly recommends using MSAccess for database management and design.

General Description of WHEAT Data Storage

WHEAT uses data stored in a Microsoft Access format relational database for both the attributes and coordinates of various geographic features. The Microsoft Access database format is also the native database format for Visual Basic 3.0 and 4.0. (A short description of relational databases is included later in this section.)

WHEAT uses a relational database to store information about geographic features, both descriptive (or tabular) attributes and locational attributes. The location(s) of features are described as X (east-west) and Y (north-south). WHEAT does not use any particular coordinate system, except that X and Y will be plotted as equal distances on the screen and on printing, and X increases to the right and Y increases to the top. WHEAT assumes coordinates are in a right-handed, isotropic coordinate system. There may be information about the database's coordinate system in the table ztblWHEAT_LOG, in the UNITS entry.

A relational database stores all information on a topic, such as water users or schools, in tables. Tables are organized sets of information containing records, one for each individual thing in the database. A record is composed of fields describing the thing. Thus, a table concerning schools might have records for each school in a county, and the records might include the name (as text), date founded (as date), operating budget (as a floating point number), date closed (as date), number of buses owned (as integer number), principal's name (as text), and number of pupils (as integer number). Notice that some of the fields listed, such as date closed and number of buses, will not occur for each individual school. In these cases, a relational database stores a null, indicating that no information is available.

While a relational database *stores* information in tables, it can *retrieve* information as virtual tables called queries. A query is the answer to a question asked about the datatable, such as: "show all schools with an operating budget above $500,000". A query can return data from several related tables by joining them. For example, suppose we had another table containing information about students. It might contain student's name, age, address, and name of school attended. We could then find out who a student's principal is by designing a query which relates the school name field in the student table to the name field in the school table. Queries are generally used for combining information from several tables, selecting subsets of records, or selecting subsets of fields. By allowing data to be stored in a single location (such as the principal's name in the school field) and related at a later time, relational databases can reduce storage requirements, and allow the updating of data to be considerably easier than if each record contained all data that might ever be needed.

FEATURE ATTRIBUTES

Attributes of geographic features that are not locational are simply stored in regular fields and are be accessed using standard Access/Visual Basic methods.

LOCATIONAL DATA

There are two general categories of features on a map: those defined by a single coordinate pair, such as the location of a house or a weather station, and those defined by a series of ordered coordinate pairs, such as a road or a lake. For retrieving single point locations, WHEAT uses two fields, one for X and one for Y. For retrieving multi-point features, WHEAT uses one field containing the locations in what is hereafter called XYChain format, which is simply an ordered array of (X, Y) locations stored in the database as a binary large object (or BLOB). (WHEAT sometimes uses Tiles, which are cardinally-oriented rectangles specified by two points in a total of four fields; for example, stored views in the zWHEATViews table.)

Internally, WHEAT stores point locations in a structure referred to as a RealPoint, which is simply X and Y as two four-byte floating point numbers (also known as singles, real*4s, and short floats). An XYChain is an array of RealPoints (or ordered set of point locations), and a RealRect is two RealPoints in the order upper left, lower-right. All geographic features in WHEAT are stored as some combination of RealPoint, RealRect, or XYChain.

Point Locations

For point locations, used for point symbols, text labels, and, as a pair, for tiles, WHEAT simply gets the values from two separate fields in the table. If a WHEAT program seems unable to find the field, make sure they start with X_ and Y_

XYChains

Unfortunately, MSAccess and other relational database systems usually do not provide an easy way to store arrays. WHEAT stores the array of RealPoints in a string which is stored in the database in a Binary Large Object (BLOB) field, also known as an OLE Object field or LongBinary field. This manual deals largely with a set of WHEAT programs used for getting coordinates into and out of WHEAT XY Chains.

(The reason that arrays are stored in a binary format rather than text is that loading them is several orders of magnitude faster than storing them as text and converting them every time the chain is loaded. Additionally, using a binary format prevents accidental modification. WHEAT does allow direct access to the coordinates, but uses special programs, described herein, to allow it.)

As an example, part of I-70 passing through Lawrence, Kansas has coordinates passing through the following points:

```
(827081.6, 4323060)
(830628.8, 4323736)
(832790.3, 4324620)
(833990.9, 4325363)
(837328.4, 4328493)
(839079.6, 4329595)
(839777.3, 4330034)
(841534.6, 4331605)
(842781.1, 4332046)
(846055.9, 4332798)
```

A hex dump of this binary XYChain, used by WHEAT, would look like

```
99 EC 49 49 E7 ED 83 4A
4C CA 4A 49 30 F3 83 4A
65 51 4B 49 17 FA 83 4A
6E 9C 4B 49 E6 FF 83 4A
06 6D 4C 49 5A 18 84 4A
7A DA 4C 49 F6 20 84 4A
15 06 4D 49 64 24 84 4A
EA 73 4D 49 AA 30 84 4A
D1 C1 4D 49 1B 34 84 4A
7F 8E 4E 49 FB 39 84 4A
```

The first form is easy for humans to read and understand, but requires parsing a text string and translating the values into internal binary numbers to be used by WHEAT every time the XYChain is loaded, a slow and error-prone process. The second form is convenient for WHEAT but nearly useless for humans. This manual deals with programs that translate between human-readable form A and machine-readable form B.

Converter Programs

This section lists available programs to perform format conversion on WHEAT XYChain data. For all formats listed in this document, there are programs to both import and export. If you don't understand what a format should look like, try exporting to that format. It might help to find a small table containing relatively simple XYChains and perform a BLOB->Memo conversion in order to see the coordinates.

COORDINATES IN DATABASE

Coordinates in Text Strings Stored in the Same Table

One way to enter and store coordinates would be to have a memo (long text) field that contains the coordinates as plain text, with delimiters to separate X from Y and each pair of coordinates. For example, the following text locates that section of I-70:

```
827081.6, 4323060;  830628.8, 4323736;  832790.3, 4324620;  833990.9,
4325363;  837328.4, 4328493;  839079.6, 4329595;  839777.3, 4330034;
841534.6, 4331605;  842781.1, 4332046;  846055.9, 4332798;
```

The WHEAT program BLOBMEMO converts between coordinates stored in internal XY Chains and coordinates stored in as a single string of coordinate pairs. Note that each pair is terminated with a delimiter, including the last one.

To perform a conversion, start BLOBMEMO , open a database, and select the table of your choice. If you are converting from XYChains from BLOBs to Memos, select your BLOB field (it is probably already selected), select or name your Memo field (or use the default name), click the BLOB to Memo button, and wait while it processes. Going from Memo (text) to BLOB (binary) is just as easy, but remember that the delimiters must be present on all coordinate pairs. You can also choose delimiters in either case, or take the defaults.

Loose Points Stored in Another Table

One way to store chains would be store the locations in another table as (ChainSerialNumber, PointOrder, X, Y), like this

PointsOnContact

ChainID	Order	X	Y
54652	1	827081.6	4323060
54652	2	830628.8	4323736
54652	3	832790.3	4324620
54652	4	833990.9	4325363
54652	5	837328.4	4328493
54652	6	839079.6	4329595
54652	7	839777.3	4330034
54652	8	841534.6	4331605
54652	9	842781.1	4332046
54652	10	846055.9	4332798
54653	1	827197.6	4323915
54653	2	827081.6	4323060
54655	1	786898.6	4321270
54655	2	788179.2	4321343
54655	3	789158.2	4321969
54655	4	793464.5	4325875

This doubles the amount of storage space required (16 bytes for each location rather than 8 bytes), but is convenient sometimes.

Two programs convert these formats: BLOB2PTS and PTS2BLOB. With BLOB2PTS, you open a database, select a table, select a field containing the XYChains and a field containing the unique ID number, choose a name for the output table, click on the button labeled Convert XYChain to Points, and wait while it creates the new table containing the points as loose X, Y pairs.

PTS2BLOB converts a table containing loose X, Y pairs with chain IDs and order numbers into a table of chains. It has special requirements on the field names. The X field must contain the string X_, as in X_UTM14 or X_ or Coordinate_X_; the Y field must contain the string Y_. The point order must contain either the string SEQ (as in sequence) or ORD (as in order). The chain IDs must be in a field whose name contains SER (as in serial number) or ID (as in RoadID).

COORDINATES IN FILES

This section only deals with files that contain XYChains of points. If you have a file containing loose points, such as well locations, use the facilities provided by MSAccess or Excel.

Plain Text File

If you have a text file containing the XY locations for each chain, along with the serial number and the number of points, as shown below, you can use GENIMPEX (Generic Import-Export program) to perform the import. You can export to such a file using GENIMPEX.

To export a set of chains to a file, start GENIMPEX, click on the Export tab, select the database, table or query, and ID and XYChain fields, choose a file name, choose ascii or binary, and click on Export Now. To import, open the program, open the database, select the file to import, choose ascii or binary, choose a name for the table, and click on Import Now.

Example GenImpEx ascii file.

```
WHEATCHAIN ASCII
 54414          7
 779151.3       4415224
 779796.9       4412822
 780222.3       4410278
 780569.8       4404342
 780664.9       4401382
 781015.8       4390462
 781235.1       4390277
 54416          2
 798216.8       4394561
 798396.8       4387524
 54417          8
 781235.1       4390277
 781934.3       4389684
 782944.9       4388299
 783723.6       4387714
 790804.9       4388187
 792039         4387276
 793788.8       4387345
 798396.8       4387524
```

Simple Binary File

The binary equivalent of the above simple ascii file can be imported or exported using GenImpEx as well. The file must be purely binary data (no record lengths or end of record marks) and the first 8 bytes must be WHEATCHN (case sensitive). The procedure is the same as for simple ascii text files.

You might have a binary file if you have some way of generating coordinates and a program that converts to this format. For example, you could have GPS that produces coordinate streams in some other format and have written a program to read that format and produce this one. You would use binary rather than ascii because it is much faster. If you can't get the binary files to work, use ascii instead.

Arc/Info Generate Format File

For point data coming from ARC/INFO, just ADDXY to the PAT, use infodBase to export it, and import the dBase file using Access; you will probably want to convert a lot of the field types to long integers instead of the doubles that dBase uses.

For line and polygon data coming from ARC/INFO, make sure the COVERAGENAME-ID is unique, UNGENERATE the coverage using the line or poly option. Transfer the data to the PC and import it using ARCS_IN4. (Open a database, open a generate file, and let it process.) Export and import the AAT and/or PAT using InfoDBase as above. After modifying the attributes table, create a query joining the two, then modify that to a Make query to create the new table. If you have many generate files in the same directory, you can import them all at once using File->Import_All_Files_In_Directory.., then choosing a wildcard string and the directory to search.

To export data from WHEAT to Arc/Info Generate format, use WHT2ARC Choose a database, a table or query, the type of coverage to produce, the fields that contain the coordinate data, additional fields to write to a dBase file, and click on Export. When finished, you will get an UNGENERATE file and a dBase file to import into Arc/Info.

COORDINATES IN EXCEL SPREADSHEET

If you can get the coordinate data into an Excel Spreadsheet, or generate it there from field notes, you can use the WHEAT_XL add-in to translate data between Microsoft Excel to and WHEAT.

The illustration below shows how the data must be arranged for export. The field names must be in the top row. The second row must contain field types (there is a list below the illustration). Actual data is in the third row and down to the first row containing a blank in the starting column (after the end of a chain). For XYChains, the coordinate data for a feature lies in the rows below the regular data for the feature.

	A	B	C	D	E	F
1	SerialNumber	Tier	Range	Section	XYBLOB_UTM14	
2	Long	Text	Text	Text	Binary Large Object	
3	224415392	29S	40W	1		
4					265688.2	4160382
5					267296.3	4160334
6					267252.6	4158727
7					265641.5	4158777
8					265688.2	4160382
9	224415360	29S	40W	2		
10					264072	4160432
11					265688.2	4160382
12					265641.5	4158777
13					264025.2	4158830
14					264072	4160432
15	224415328	29S	40W	3		
16					262457.3	4160484
17					264072	4160432
18					264025.2	4158830
19					262419.4	4158879
20					262457.3	4160484
21						
22	This is past the end of the data.					

To export data from Excel to WHEAT, select menu-item File->WHEAT->Export Data... Choose the database to export data into. A dialog will prompt you to select the range containing the data (point at the upper-left corner of the data), the output table name (under your control, except that you can't use a previously-used name) and the database (which you already chose, but you can over-ride it here). Click on OK and the data will be exported to the database.

Importing data is much easier. Simply select menu-item File->WHEAT->Import Data... , choose the area to export data into (only the upper-left corner is used by the program), choose the table or query containing the data, and click on

OK. (Once again, you can change databases here as well as initially.) The data will then be written to the spreadsheet.

You can create a new database, or add the WHEAT special tables to an old database, using the menu-item File->WHEAT->New Database...

Field types should be one of the following types. Normally, you should use long integers (INTEGER*4), singles (REAL*4), text (CHARACTER*256), and LongBinary (BLOB data).

Allowable TypeNames	Data Type
Boolean	Boolean
Byte	Byte
Integer	Integer
Long	Long
Single	Single
Double	Double
Currency	Currency
Date/Time Date Time	Date
Text String	Text
Binary Large Object (BLOB) Binary Large Object BLOB LongBinary OLE Object XYChain OLE	LongBinary
Memo LongText	Memo
All others	Text

Kansas Geological Survey
Open-file Report

Disclaimer

The Kansas Geological Survey does not guarantee this document to be free from errors or inaccuracies and disclaims any responsibility or liability for interpretations based on data used in the production of this document or decisions based thereon. This report is intended to make results of research available at the earliest possible date, but is not intended to constitute final or formal publication.