

OBJECT-ORIENTED CODE TO LOOKUP SOIL TEXTURE CLASSES FOR ANY SOIL CLASSIFICATION SCHEME

C.B.S. Teh^a and M.A. Rashid^b

^a Department of Land Management, Universiti Putra Malaysia, 43400 UPM Serdang, Selangor, Malaysia. Corresponding author (fax: +60-3-89434419; email: chris@agri.upm.edu.my)

^b Laboratory Unit of Multimedia Bioresource and Agroinformatics, Universiti Putra Malaysia, 43400 UPM Serdang, Selangor, Malaysia

ABSTRACT

TALCPP is an object-oriented C++ code to lookup soil texture classes based on any soil classification scheme. TALCPP describes each soil texture class in a given scheme as a polygon located in a biaxial (x, y) or (%sand, %clay) coordinate system. The principle to determine the texture class is to determine if, in the texture chart, the point of intersection between any two primary particles lies within a texture class polygon. The polygon inside test method is used to test if a point lies inside or outside a polygon. TALCPP also supports Range Lookup which is the lookup of the range of soil texture classes based on a given range of particle size distributions. The texture class is determined by testing if the source polygon enclosing the given range of particle size distributions could clip the target polygon of a particular texture class. A successful clipping between these two polygons indicates that the target texture class

belongs within the given range of particle size distributions. The polygon clipping method used is the algorithm adapted from Vatti's algorithm (1). Lastly, a graphical user interface program called TAL for Windows was also developed. This program uses TALCPP to determine soil texture classes, and is intended for everyone including non-programmers.

INTRODUCTION

A soil textural class is the relative proportion of the three primary soil particles: sand, silt and clay. A triaxial or biaxial texture chart is usually used to determine the texture class of a given soil. In reading the chart, the point of intersection of any two primary particles will locate the textural class. Though reading the texture chart is simple and fast, using computers to automate the task of looking up the soil texture classes is justified when one is handling a large number of soil samples, or for routine particle size analyses.

There is an increasing number of modern computer programs (2, 3, 4) to automate this lookup task since the introduction of a program called *Texture AutoLookup* (TAL) by Christopher and Mokhtaruddin (5). The use of these programs, however, are limited only to the USDA (United States Department of Agriculture) soil classification scheme, neglecting other important schemes such as the International scheme (used in Australia), the United Kingdom (UK) scheme (used in England and Wales), the FAO (Food and Agriculture Organization) scheme, and other schemes devised by countries such as Canada, New Zealand, Germany, France, India, and Belgium. Considering such a variety of soil classification schemes, a computer program is therefore needed that is generic enough to handle most, if not all, of these schemes.

Thus, this paper is to introduce an object-oriented C++ code called *TALCPP* that can lookup a soil texture class based on any soil classification scheme. And because this code is object-oriented, it is reusable and extendible, meaning that TALCPP can be used or be incorporated into another C++ program (such as a soil water model or soil survey program) seamlessly, without modifying the original code to achieve code compatibility. TALCPP is intended for programmers and modellers; however, a GUI (graphical user interface) program called *TAL for Windows* was also developed. This program uses TALCPP to lookup soil texture classes, and is intended for everyone including non-programmers and non-modellers.

TALCPP

Program Description

TALCPP is written in C++ language, and conforms completely to the C++ standards set by ANSI (American National Standard Institute) and ISO (International Standard Organization). Consequently, TALCPP is independent of hardware and operating systems, and can be compiled using any modern C++ compiler on any operating system.

TALCPP describes each soil texture class in a given scheme as a two-dimensional polygon. In Figure 1, for example, a hypothetical soil classification scheme consists of five texture classes, and each of these texture classes is described as a polygon located in a (x, y) or $(\%sand, \%clay)$ coordinate system (Table 1). The principle to lookup a soil texture class is to determine if a given particle size distribution lies within the boundaries of a texture class. In other words, it is to determine if a given point lies inside a polygon. For this task, the *inside test* method (6) is

used, whereby a line segment is constructed between the point in question and a point outside the polygon (Figure 2a). A point outside the polygon is simply any point with an x -coordinate smaller than the smallest x -coordinate of the polygon's vertices (corners). Once the line segment is constructed, TALCPP counts the number of intersections of the line segment with the polygon boundaries. Odd number of intersections means the point in question is inside the polygon; otherwise, an even number of intersections indicate a point outside the polygon. The exception is when the intersection happens at a polygon vertex. For this situation, TALCPP looks at the other endpoints of the two segments which meet at this vertex (Figure 2b). If these points lie on the same side of the constructed line, then the point in question counts as an even number of intersections. But if they lie on the opposite sides of the constructed line, then the point is counted as a single intersection. Note that if a point lies exactly on a polygon boundary, TALCPP considers this point as being inside the polygon.

For each texture class of a given soil scheme, the (x, y) coordinates of its polygon vertices (*e.g.*, Table 1) are stored in a pre-defined format in a text file called the "scheme definition file". Currently, thirteen scheme definition files have been prepared. These files are for the soil schemes: 1) USDA (7), 2) UK (8), 3) Canada (9), 4) International (10, 11), 5) India (12), 6) International Society of Soil Science (13), 7) Switzerland (14), 8) Belgium (15), 9) New Zealand (16), 10) FAO (13), 11) AISNE, France (17), 12) Germany (18), and 13) the Shepard's proposed scheme for sedimentologists (19).

TALCPP also supports user-defined classification schemes. This means users can prepare their own scheme definition file for a scheme they regularly use, or for a totally new scheme. And most importantly, the inside test method to lookup soil texture classes, as described previously, is still applicable and unaffected by the soil classification scheme type.

TALCPP has two C++ classes: `polygon` and `soiltex`. The `polygon` class stores the (x, y) coordinates of the vertices of a single polygon, and handles polygon-related calculations such as the inside test method, to determine the orientation, centre, and area of the polygon. The `soiltex` class represents the soil classification scheme and holds all texture class definitions in that scheme. To implement this design, the `soiltex` class has a container for polygon objects where each polygon object represents the definition of one texture class. This class loads the texture class definitions from the scheme definition file, and coordinates the inside test method. This test is performed iteratively on each stored polygon object. Note that the inside test method continues up to the last stored polygon object even if the point is determined to be inside a previous polygon texture class. This is because a point may lie on the boundaries of two or more polygons.

The lookup of texture classes from a given point in the texture chart is known as *Point Lookup*. TALCPP also supports *Range Lookup* which is the lookup of texture classes from a given range of particle size distributions. Range Lookup is particularly useful when only rough estimates are available on a soil's particle size distribution; for example, to determine the USDA texture classes of a soil having between 30 to 50% clay and between 20 to 40% sand. As stated earlier, the inside test method is used for Point Lookup, but for Range Lookup the polygon clipping method adapted from Vatti's algorithm (1) called *Generic Polygon Clipper* (GPC) is used. The GPC algorithm was developed by Alan Murta (unpublished work; further information from <http://www.cs.man.ac.uk/aig/staff/alan/software//gpc.html>), and GPC can handle clippings of concave and convex polygons. GPC is written in C language, and it can be downloaded from <http://www.cs.man.ac.uk/aig/staff/alan/software/>.

In the Range Lookup, the range of texture classes is determined by testing if the source polygon enclosing the given range of particle size distributions can clip (intersect) the target polygon of a particular texture class. A successful clipping between these two polygons indicates that the target texture class belongs within the given range of particle size distributions. The polygon clipping method is implemented in the GPC library, and in turn the entire GPC library is encapsulated or wrapped by the polygon class. This is to shield users from the implementation details of GPC, and to achieve object-orientation. Finally, the `soiltex` class coordinates the polygon clipping method so that the success or failure of clipping the source polygon with a target polygon is checked iteratively for every texture class in the soil scheme.

Sample Use of TALCPP

The code excerpt in Figure 3 shows one way to determine the texture class for a given particle size distribution (Point Lookup). Note that all TALCPP classes are defined in the namespace `tal`. The soil object of type `soiltex` is created first, then initialized with the UK soil classification scheme by calling the method `LoadScheme` to load the texture class definitions from the scheme definition file named “uk.dat”. The validity of the scheme is checked by calling the method `IsSchemeValid`. A scheme is considered valid if all its texture class polygons tile each other perfectly (no gaps and overlaps), and the total area of all polygons is 5000%² (note: the total area of a texture chart is 0.5 x 100% x 100%). Once the scheme passes the validity check, the texture class of a soil with 33.5% sand and 23.5% clay is determined by calling the method `PointLookup`. The lookup result is stored in a container of `TEXTURE` objects, which stores the names and integer codes of texture classes.

Another example is shown in Figure 4, which is the code excerpt to determine the range of USDA texture classes for a given range of particle size distributions (Range Lookup). The Range Lookup of a soil having between 10 to 50% sand and between 30 to 50% clay is determined by using the soiltex method RangeLookup.

In the given two examples, the result container object returned after a Point or Range Lookup can be indexed and displayed to the screen. An example of how to do this is shown in Figure 5. Note that the TEXTURE object attributes mCode and mName are the integer code and name of the texture class, respectively.

TAL FOR WINDOWS

TAL for Windows is a GUI program that runs only in Microsoft Windows 95 and onwards (Figures 6 and 7). This program uses TALCPP to determine the soil texture classes based on any soil classification scheme. Users will only need to load the appropriate scheme definition file for a particular soil classification scheme. Data entry in TAL for Windows is intuitive because the particle size distribution is entered in a similar way to spreadsheet programs (Figure 6). The soil texture classes and soil data can be printed or saved to a comma-delimited text file (CSV format). Hence, these text files can be imported from or exported to a text editor or spreadsheet. TAL for Windows also has charting capabilities (Figures 8 and 9), where the charts can be saved as a picture file, copied to the clipboard, or be printed.

TAL for Windows was developed using MFC (Microsoft Foundation Classes) and compiled with Microsoft Visual C++ version 5.0 (patched with Service Pack 3). TAL for Windows occupies approximately 2.5 Mb of disk space.

Both TALCPP and TAL for Windows (including their documentation and scheme definition files) are available upon request from the corresponding author, or can be downloaded for free from <http://www.agri.upm.edu.my/~chris/tal>.

ACKNOWLEDGMENT

Dr. Detlef Deumlich (Centre for Agricultural Landscape and Land Use Research, Germany) and Mr. Simon Six (Catholic University of Leuven, Belgium) have contributed some scheme definition files.

REFERENCES

1. Vatti, B.R. A Generic Solution to Polygon Clipping. *Commun. ACM* **1992**, 35: 56-63.
2. Budihal, S.L. A FORTRAN Program to Classify the Textural Class of a Soil According to the USDA Triangular Textural Diagram. *J. Ind. Soc. Soil Sci.* **1997**, 45: 382-383.
3. Gerakis, A.; Baer, B. A Computer Program for Soil Textural Classification. *Soil Sci. Soc. Am. J.* **1999**, 63: 807-808.
4. Liebiens, J. Spreadsheet Macro to Determine USDA Soil Textural Subclasses. *Commun. Soil Sci. Plant Anal.* **2001**, 32: 255-265.
5. Christopher, T.B.S.; Mokhtaruddin, A.M. A Computer Program to Determine Soil Textural Class in 1-2-3 for Windows and Excel. *Commun. Soil Sci. Plant Anal.* **1996**, 27: 2315-2319.
6. Harrington, S. *Computer Graphics: A Programming Approach*; McGraw-Hill: New York, 1983.

7. Soil Survey Division Staff. *Soil survey manual*; USDA Handbook 18; U.S. Government Print Office: Washington, 1993.
8. Avery, B.W. *Soil Classification For England And Wales [Higher Categories]*; Soil Survey Technical Monograph 14; Rothamsted Experimental Station: Harpenden, UK, 1980.
9. Canada Department of Agriculture. *A System of Soil Classification for Canada*, Revised Edition; Publication 1455. Information Canada: Ottawa, 1974.
10. Marshall, T.J. *Mechanical Composition of Soil in Relation to Field Descriptions of Texture*; Bulletin 224; CSIR: Melbourne, Australia, 1947
11. Leeper, G.W.; Uren, N.C. *Soil Science: An Introduction*, 5th Edition; Melbourne University Press: Carlton, Australia, 1993.
12. All India Soil and Land Use Survey Organization. *Soil Survey Manual*, Revised Edition; New Dehli, 1971.
13. Verheye, W.; Ameryckx, J. Mineral Fractions and Classification of Soil Texture. *Pedologie* **1984**, 2: 215-225.

14. Jäggi, F.; Frei, E. Vorschlag eines neuen Körnungsdiagrammes. Bulletin Bodenkundliche Gesellschaft der Schweiz **1977**, 1: 42-48.

15. Sys, C. *La Cartographie des Sols au Congo ses Principes et ses Methodes*; Serie Technique No. 66; Publications de l'Institut National pour l'Etude Agronomique du Congo: Congo, 1961.

16. Gibbs, H.S. *New Zealand Soils. An Introduction*; Oxford University Press: Wellington, 1980.

17. Baize, D. *Soil Science Analyses: A Guide to Current Use*; Wiley: Chichester, UK, 1993.

18. Boden, A.G. *Bodenkundliche Kartieranleitung*, 4th Edition; E. Schweizerbart: Stuttgart, 1994.

19. Shepard, F.P. Nomenclature Based on Sand-silt-clay Ratios. *J. Sedimentary Petrology* **1954**, 24: 151-158.

Table 1. Texture class polygons for the hypothetical soil scheme

Texture class	Polygon vertices (x, y) or (%sand, %clay)
silt	(30, 0), (0, 0), (0, 30)
silt loam	(40, 0), (30, 0), (0, 30), (0, 50), (40, 50)
clay	(50, 50), (0, 50), (0, 100)
sandy loam	(70, 0), (40, 0), (40, 50), (50, 50), (70, 30)
sand	(100, 0), (70, 0), (70, 30)

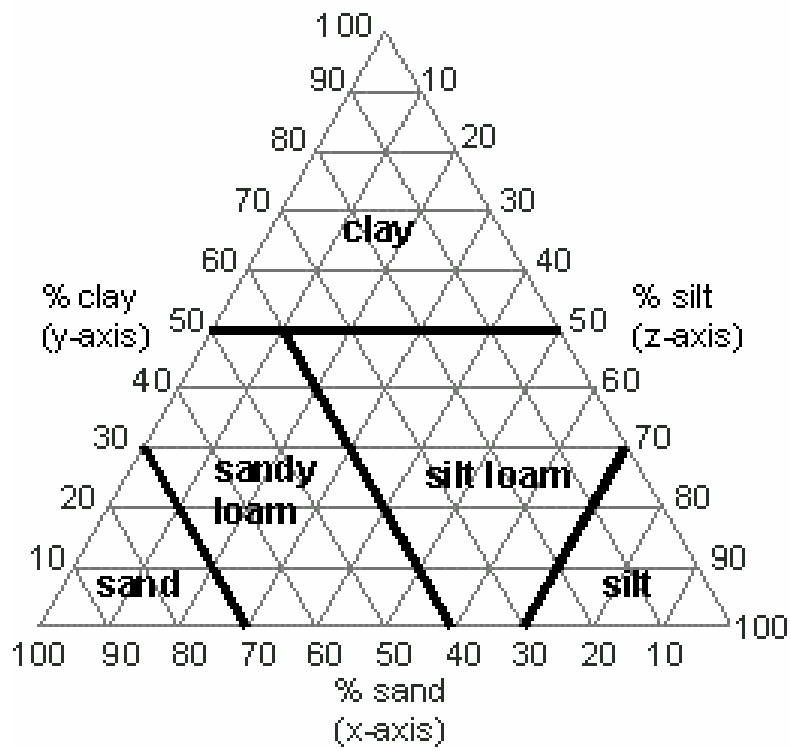
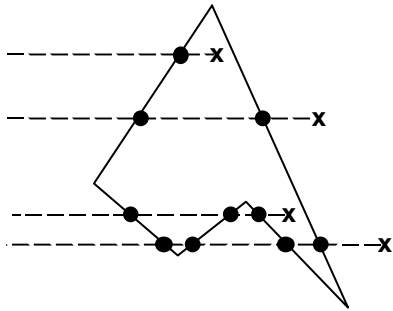


Figure 1. Texture chart for a hypothetical soil classification scheme

(a)



(b)

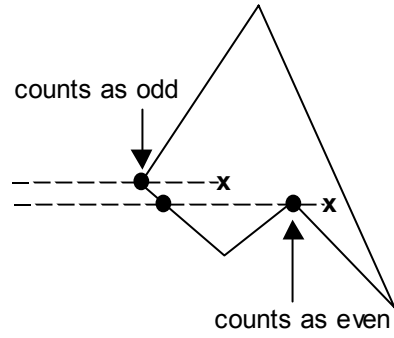


Figure 2. (a) The inside test method, and (b) the intersection with a polygon vertex

```
std::fstream fin("schemes/uk.dat"); // file with the UK scheme
tal::soiltex soil;                  // soiltex object
soil.LoadScheme(fin);               // load the scheme from file

if (soil.IsSchemeValid())           // ensure scheme is valid first
{
    double sand = 33.5;              // 33.5% sand
    double clay = 23.5;             // 23.5% clay
    std::vector<tal::TEXTURE> result = soil.PointLookup(sand, clay);
}
```

Figure 3. C++ code excerpt for Point Lookup

```
std::fstream fin("schemes/usda.dat"); // file with the USDA scheme
tal::soiltex soil;                    // soiltex object
soil.LoadScheme(fin);                 // load the scheme from file

if (soil.IsSchemeValid())             // ensure scheme is valid first
{
    tal::RANGE sand(10, 50);          // 10 to 50% sand
    tal::RANGE clay(30, 50);         // 30 to 50% clay
    std::vector<tal::TEXTURE> result = soil.RangeLookup(sand, clay);
}
```

Figure 4. C++ code excerpt for Range Lookup

```
// display every texture class code and name
for (int i=0; i<result.size(); ++i)
    std::cout << result[i].mCode << ", "
               << result[i].mName << std::endl;
```

Figure 5. C++ code excerpt to index and display the results after a Point or Range Lookup

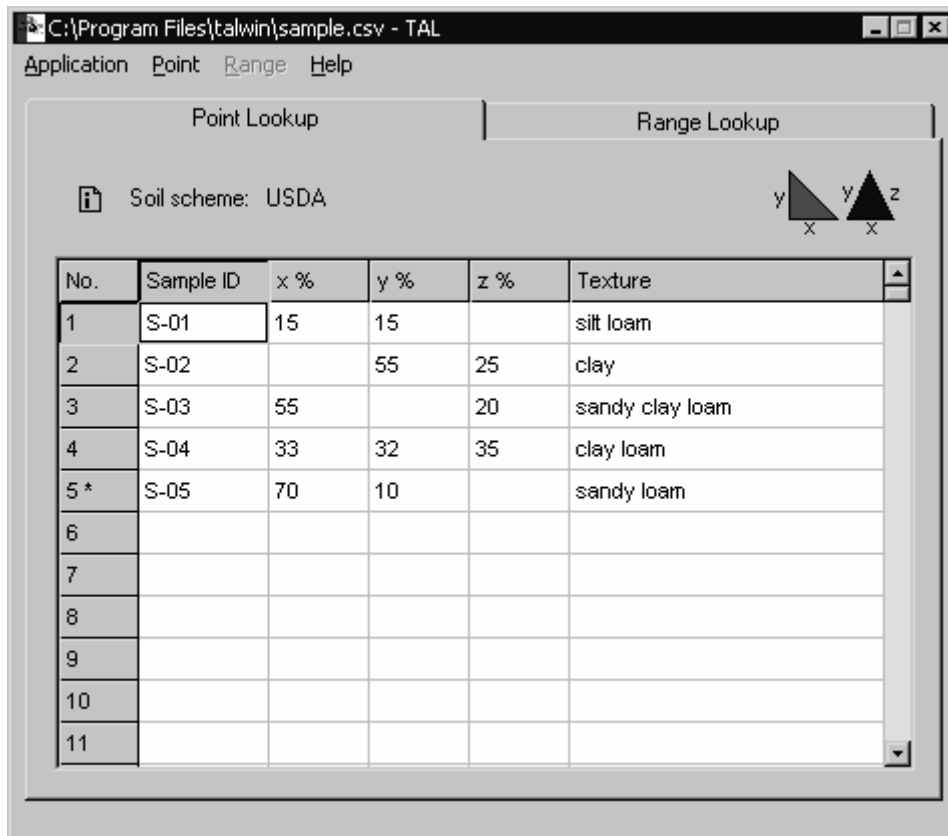


Figure 6. Point Lookup in TAL for Windows

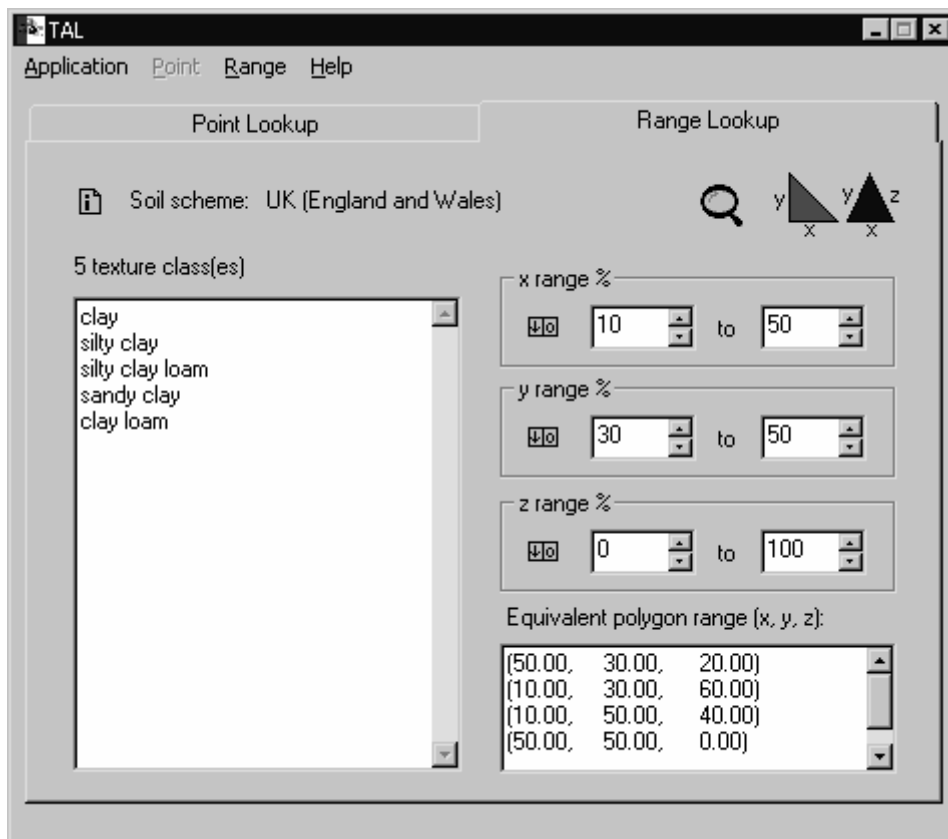


Figure 7. Range Lookup in TAL for Windows

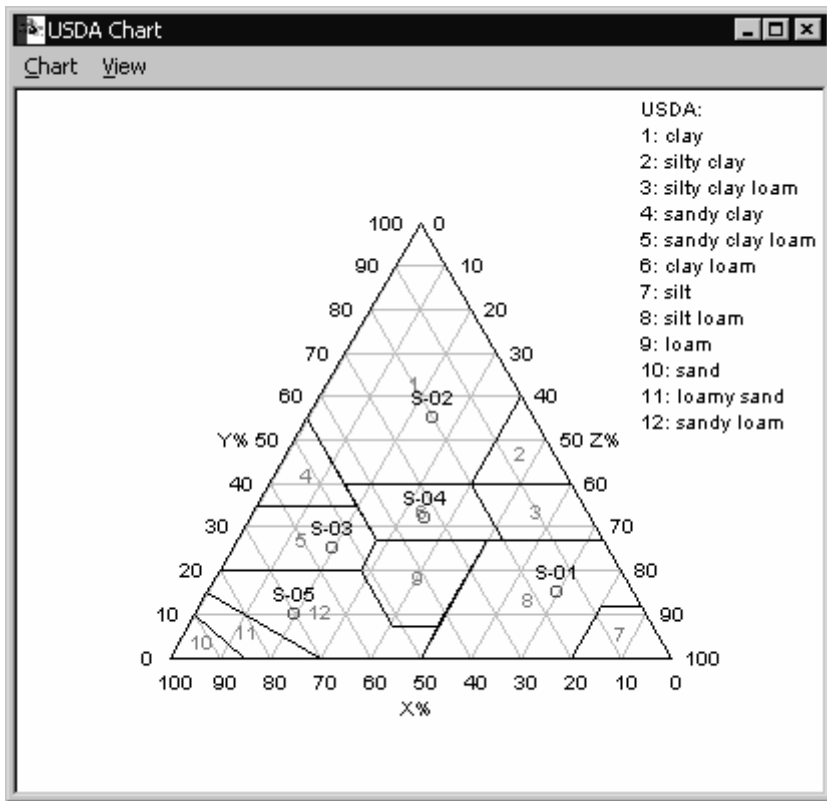


Figure 8. Point Lookup chart in TAL for Windows

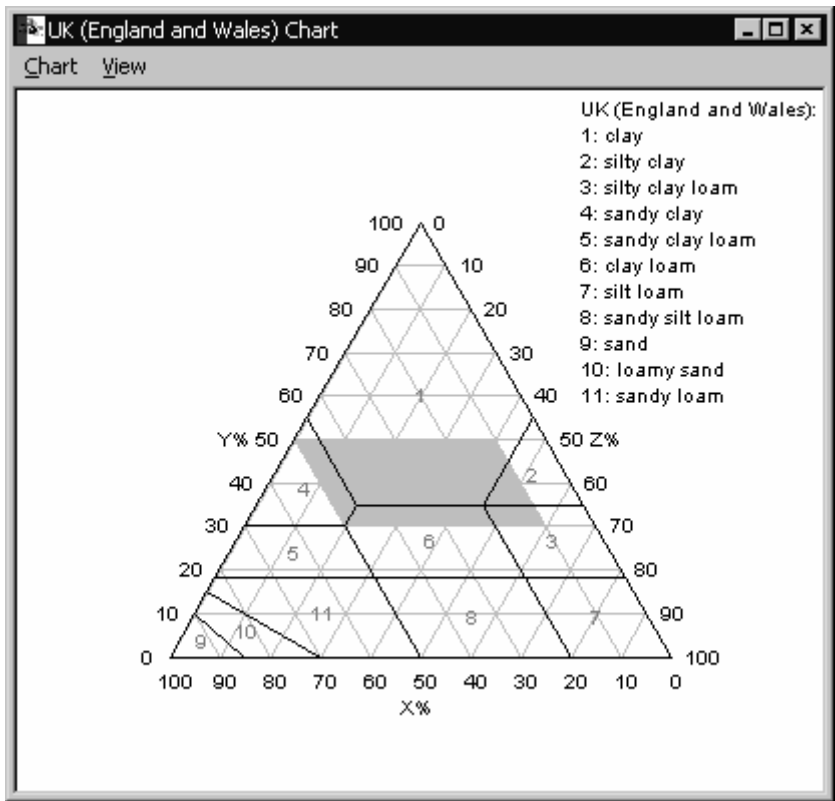


Figure 9. Range Lookup chart in TAL for Windows