GUICS: A Generic User Interface for On-Farm Crop Simulations

Basil Acock, Yakov A. Pachepsky,* Eugene V. Mironenko, Frank D. Whisler, and Vangimalla R. Reddy

ABSTRACT

As knowledge of crop growth and development is quantified, it can be included in the computer code of crop simulators that mimic the essential features of plant–soil–atmosphere–management interactions. The next step necessary is to provide tools to simplify operation of the crop simulator by end users. Our objectives were to (i) develop a graphical user interface (GUI) specifically oriented to on-farm use and (ii) research the possibility of building a generic GUI that could be used with many crop simulators not necessarily having the same structure of input data. We call the interface GUICS (standing for Graphical User Interface for Crop Simulations). GUICS is built on the usability paradigm developed by software designers and has a user-centered design based on results of human–computer interaction studies. The usability of the interface is facilitated with special features that enhance the *directness*, *user-in-control*, *consistency*, *forgiveness*, *feedback*, and *simplicity* of the interface. Including a simulator in GUICS is a straightforward operation that does not require changes in the simulator code. The GUICS prototype was evaluated by its future users in interviews during which they had hands-on experience. Two years of on-farm use of GUICS has proved the usability of the interface.

KNOWLEDGE dissemination and delivery to the end users have always been important tasks in agronomic research. As knowledge of crop growth and development is quantified, it can be included in computer code of crop simulators that mimic plant–soil–atmosphere–management interactions. The relatively short 30-year history of computer crop simulation has seen the establishment of principles and techniques, as well as the development of several widely used crop simulators and crop simulator families, such as CERES, EPIC, CROPGRO, CROPSYST, GOSSYM, and GLYCIM.

Developing a simulator is the most important but only the first step in knowledge dissemination. The next step is to provide tools to simplify operation of the crop simulator by end users. The need for such tools stems from limits to the computer literacy of users, and from the lack of free time for learning and/or performing tedious procedures for entering data and displaying results. Without these tools, potential benefits from the use of the simulators may not be realized, and assembling the input data may be a formidable task. Developing a user interface that incorporates these tools has been recognized as an essential step toward increasing the utility of crop simulators and decision support systems (Landivar et al., 1989; Cox, 1996).

Command-line interfaces have been replaced during the last decade with graphical user interfaces (GUI) based on graphics (icons, pictures. and menus) instead of text. User interface design and implementation is a growing field in software engineering (Redmond-Pyle and Moore, 1995). The theoretical background of user interface development lies in human–computer interaction studies (Macaulay, 1995). The functioning of human cognition and memory has profound implications for interface development (Mandel, 1997). GUI development is centered on setting the usability requirements (Redmond-Pyle and Moore, 1995) that will simplify the performance of predefined tasks. The issues addressed are ease of learning, elimination of sources of errors, ergonomics, and prevention of frustration and negative feelings about the interface. The main issue, however, is understanding the user's needs.

Several graphical user interfaces to crop models are described in the literature (e.g., Hoogenboom et al., 1994; Van Evert and Campbell, 1994; Waldman and Rickman, 1996; Testezlaf et al., 1996). All of them have been built specifically for a single crop model or for a family of models. In spite of several attempts to standardize crop simulators, the numbers and contents of input and output files vary from one crop simulator to another. This does not mean, however, that each crop simulator or family (suite) of crop simulators must have its own GUI. Learning several different GUIs to perform similar tasks with different crop simulators is neither a necessary nor a desirable use of time for such busy people as farmers. Therefore, there is a need to develop a generic GUI that can be used with all, or at least most, crop simulators—a need that has already been discussed (e.g., Rewerts et al., 1989; Wu, 1992).

Crop simulators have various groups of users, including consultants, farmers, students, and researchers. The uses of crop simulators vary among the groups, and so do the usability requirements. Farmers constitute potentially the largest group of crop simulation users. On-farm crop simulations provide decision support information for farm operations. The on-farm simulations should provide (i) warnings of the need to perform certain management operations and (ii) a summary of the consequences of doing or not doing the operation.

B. Acock and V.R. Reddy, Remote Sensing and Modeling Lab., USDA-ARS, Beltsville, MD 20705; Ya. Pachepsky, Duke Univ. Phytotron and USDA-ARS Remote Sensing Modeling Lab., Bldg. 007, Room 008, BARC-West, Beltsville, MD 20705; E. Mironenko, Dep. of Horticulture, Univ. of Maryland at College Park, College Park, MD 20742; and F.D. Whisler, Dep. of Soil and Plant Sciences, Mississippi State Univ., Mississippi State, MS 39762. Received 23 Sept. 1998. *Corresponding author (ypachepsky@asrr.arsusda.gov).

Published in Agron. J. 91:657–665 (1999).

Abbreviations: DLL, dynamic link library; DSS, decision support system; GUI, graphical user interface.

Results of the simulations should be presented in terms that are most meaningful for farmers. Surveys of on-farm use of computerized decision support systems (DSSs), both simulation- and non-simulation-based, have shown that the complexity of DSS use is one of the most limiting factors (Greer et al., 1994). One reason for this complexity is the absence of experience and the lack of interest in developing user interfaces shown by developers of expert systems and models (Rewerts et al., 1989). A GUI needs to be developed specifically for on-farm use.

More attention must be paid to usability of crop simulators. User interfaces appear to be the key components. The software industry is currently migrating from GUI to object-oriented user interfaces (OOUI) that provide direct manipulation of objects instead of providing functions to perform (Mandel, 1997). At the same time, the research community is accumulating experience in coupling spatiotemporal data with models (Srinivasan and Engel, 1994; Christakos, 1998)—in particular, in site-specific agriculture. These new trends present important directions for the development of user interfaces for crop simulators in order to enhance their use. Precision farming is the emerging technology where these interfaces can find direct applications.

Our objectives were to (i) develop a GUI specifically oriented to on-farm use and (ii) research the possibility of building a generic GUI that can be used with many crop simulators not necessarily having the same structure of input data. We call the interface GUICS, which stands for Graphical User Interface for Crop Simulations.

INTERFACE USABILITY

Several paradigms have been developed recently to define more precisely what is called a user-friendly interface in popular literature (Redmond-Pyle and Moore, 1995). The usability paradigm (Mandel, 1997) includes (i) *effectiveness of task performance* or *user productivity*, (ii) *learnability*, including the primary learning time and relearning time in intermittent use, (iii) *flexibility* in adapting to changing tasks or a changing environment, and (iv) *attitude*, defined in terms of the users liking or disliking the interface. The usability of an interface is always defined in relation to specified users performing specified tasks. A task-oriented interface design (Macaulay, 1995) was adopted in this work.

A typical list of tasks performed by a user of a crop simulator is shown in Table 1. Broadly speaking, it includes data input, assembling data for a particular simulation, and analyzing the results. The list is general, and the specific content of any task depends on the user. For example, the analysis of crop simulation results may range from a sophisticated statistical and economic analysis (Thornton and Hoogenboom, 1994) to simply looking at the projected yield and the maturity date to determine profitability and the prospects of using the same machinery for several crops (Reddy et al., 1997). Another example is the comparison of simulated and measured data, which is common in research practice but is uncommon in the use of crop simulators on-farm.

Table 1. Basic tasks performed by a crop simulator user.

Category of tasks	Specific tasks
Input data in a text format	Select; View; Edit; Copy; Delete; Import; Export; Print
Input data from remote sources	$\left.\begin{array}{r}\text{Set}\\ \text{Edit}\\ \text{Delete}\end{array}\right\}$ the access address
	Actually get the data
Manage simulation scenario	Recognize/view data sets
	Assemble all data sets needed
	Run the simulation
	Copy a simulation scenario
	Delete a simulation scenario
	Vary data of some type to see the effect
Analyze results	$\left.\begin{array}{r}\text{View}\\ \text{Print}\\ \text{Compare}\end{array}\right\}$ in $\left\{\begin{array}{l}\text{graphic}\\ \text{tabular}\\ \text{text}\end{array}\right\}$ form

Our perception of users and user tasks is based on our experience of using the soybean crop simulator GLYCIM on farms in seven states of the southern USA in 1991 to 1997 (Reddy et al., 1997), and on a survey of on-farm computer use in the Great Plains (Ascough and Hoag, 1995). The soybean producers in the southern states started using the Windows interface WINGLY to work with GLYCIM in 1993. In the Great Plains, about 40% of agricultural producers use personal computers and are computer literate. The majority of them work in the Windows environment, and have Internet connections. The main reason for using computer models on farms is to increase profit, although learning more about how the crop responds to environmental factors and help in complying with governmental regulations are also important.

GUICS is a WIMP interface (Martin and Eastman, 1996); the acronym indicates that the interface includes windows, icons, menus, and pointers. It has a user-centered design (Microsoft Press, 1995), which presumes the usability enhancement by providing six interface features: directness, user-in-control, consistency, forgiveness, feedback, and simplicity. These features are discussed below.

Directness Features

Directness means taking into account the fact that humans are much better at recognition than at recollection (Redmond-Pyle and Moore, 1995). Previous developments of interfaces for crop models have shown a need to present and process information in a hierarchical form (Akins et al., 1993; Humphries and Long, 1995). The hierarchy of information units in GUICS is based on the fact that one run of any crop simulator makes predictions for a particular combination of weather, soil, crop cultivar, and farm operations. Data on weather, soil, and the like, are referred to as *datasets*, and datasets are organized by *data category* according to the type of data. A complete set of datasets for a crop simulator is referred to as a *scenario*. Several related scenarios may be combined into a group that is called a *project*. Usually a project encompasses several scenarios that describe

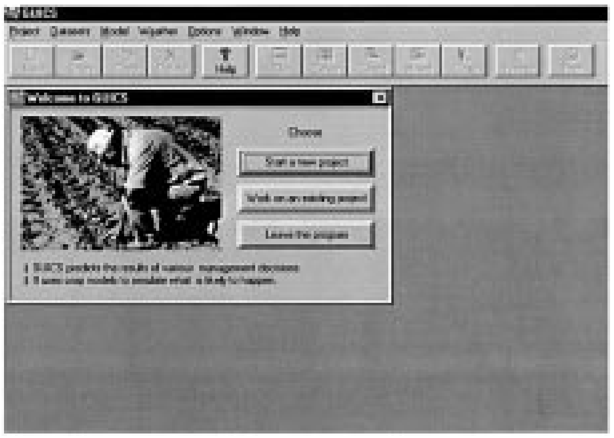

GUICS
Project Datasets Model Weather Options Window Help
Help
Welcome to GUICS
Choose
Start a new project
Work on an existing project
Leave the program

Fig. 1. Opening window of GUICS. The main window title bar and toolbar are present all the time. Toolbar buttons are enabled as needed; otherwise, they are grayed.

various fields, various management units within a field, various years, or various management practices. The projects are organized by users in a way that is convenient to them. In contrast, data categories are model-dependent and correspond to the various input files of the crop simulator. GUICS will accept any set of data categories, although the grouping of data by crop production factors (weather, soil, fertilizer, irrigation, variety, etc.) corresponds best to the user's mental model of crop growth as we encountered it.

The interface start-up window shown in Fig. 1 takes the user to the job immediately, as is recommended for application-oriented interfaces (Wood, 1998). The menu bar shows the limited number of mental objects ('Project', 'Datasets', 'Model', 'Weather') that the user needs to operate the crop simulations. The 'Project directory' window opens next. It contains a list of projects that have already been created, as well as the buttons 'New', Open', 'Copy', 'Delete', and 'Quit'. If no projects exist, all buttons except 'New' will be disabled and

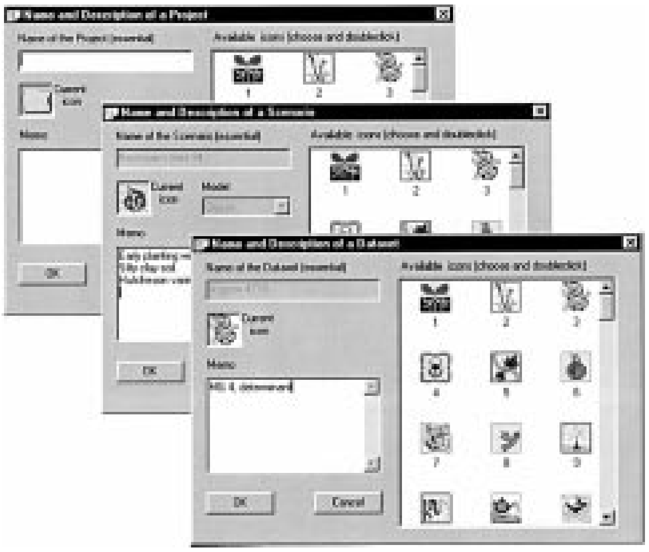
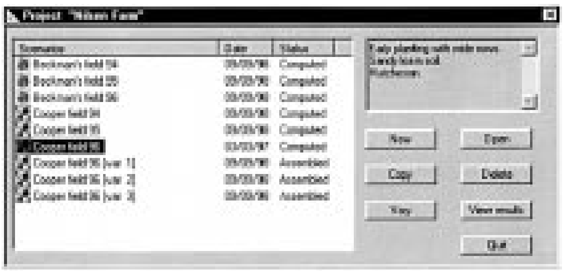

Name and Description of a Project
Name of the Project (essential)
Available icons (choose and doubleclick)
Current icon
Memo
OK
Name and Description of a Scenario
Name of the Scenario (essential)
Model
Name and Description of a Dataset
Name of the Dataset (essential)
Cancel

Fig. 2. Similarity in tools for future recognition of items: A name, a memo, and an icon are all possible attributes of a project, a scenario, or a dataset.



Scenarios	Date	Status
Beckman's field 94	[illegible]	Computed
Beckman's field 95	[illegible]	Computed
Beckman's field 96	[illegible]	Computed
Cooper field 94	[illegible]	Computed
Cooper field 95	[illegible]	Computed
Cooper field 96	[illegible]	Computed
Cooper field 96 [var 1]	[illegible]	Assembled
Cooper field 96 [var 2]	[illegible]	Assembled
Cooper field 96 [var 3]	[illegible]	Assembled

Fig. 3. A scenario dialog box showing, names, memos, scenario status, and creation data along with available actions represented by buttons. The recently assembled variants of the 'Cooper field 96' scenario are shown.

grayed, to point the user to the action required. When a new project, a new simulation scenario, or a new dataset is created, the next window presents three tools for its future recognition: the name, the memo, and the icon (Fig. 2). Giving the entity a name helps the user to recognize the item. The memo also facilitates recognition, because it appears any time an item is highlighted by clicking in a list of available projects, scenarios, or datasets (see the example shown in Fig. 3).

The usefulness of icons for recognition is user-specific, especially when the same icons are used for several similar items (Jones, 1992). The purpose of an icon is to act as a landmark for experienced users (Martin and Eastman, 1996). GUICS provides a collection of picturesque icons that a user may apply; otherwise, default icons are assigned to the items. Icons in the weather data acquisition module are an exception, as their color shows how recently the weather data have been updated.

Another way to help recollection is to use *wizards* that automate tasks through a dialog with the user (Microsoft Press, 1995). In GUICS, wizards are implemented to point out data categories that need to be reviewed when a new scenario is created. The user is guided though the process of picking datasets, as shown in Fig. 4, and one data category is shown at a time. To help users recognize whether changes have been made to a particular scenario, each scenario has a status attribute, which is set to 'Assembled' or 'Computed', depending on whether the scenario is new or changed, or has been run (Fig. 3). Most buttons are provided with balloon *tooltips* explaining their meaning. Wizards are also im-

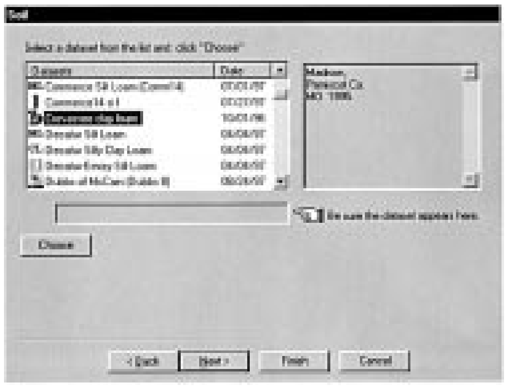

Soil
Select a dataset from the list and click "Choose"
Datasets
Date
Be sure the dataset appears here.
Choose
< Back
Next >
Finish
Cancel

Fig. 4. A dialog box from the wizard used to create a new scenario.

plemented for filling the input data files. Data tables to be filled are shown to the user one at a time. The buttons 'Next' and 'Back' provide for navigation within the data input and scenario creation processes, and they are coordinated with the wizard.

User-in-Control Features

In the User-in-Control interface, all actions should be initiated by the user and not by the software, the user should be able to personalize the software, and any interaction should be possible at any time (Microsoft Press, 1995). As the skills and preferences of users vary widely, GUICS allows them to perform the most common functions (creating a new item, opening an existing item, copying, and deleting) in any of three ways: with a pull-down menu, with buttons on the toolbar, or with buttons on the screen specific to the particular item. People are always being interrupted, and so the interface should allow users the option of changing focus (Mandel, 1997). The 'Cancel' button is active all the time the user is working with projects, scenarios, or datasets. When a simulation is running, the toolbar button 'Close' allows the user to interrupt the simulation. GUICS can also use any simulator-specific graphics (see Plugging a Crop Simulator into GUICS, below). The toolbar 'Freeze' button allows a simulation to be paused at any time, to view graphical details.

Simulation results can be presented in several forms: graphics, tables, or text. Some interfaces show some simulation results immediately after the simulation run ends (e.g., Humphries and Long, 1995), but then the user needs to recall how to get to other forms of output. GUICS presents a choice of the type of output immediately after the simulation run (Fig. 5). The user has a choice of how graphs are arranged on the screen, using the toolbar buttons 'Tile', 'Cover', 'Cascade', and 'Mixed'. Various types of printing are available using the Windows common printer dialog.

Consistency Features

Consistency allows users to transfer existing knowledge to new tasks and to learn more quickly. It makes the interface more intuitive, predictable and comfortable (Microsoft Press, 1995). Using industry standards facilitates consistency (Macaulay, 1995). The screens, menus, dialog boxes, and buttons in GUICS are like those in other Windows applications. When GUICS was beta-tested by more than 60 researchers who downloaded it from the Internet in 1996, one of their suggestions was to use the World Wide Web style to make it more attractive. We decided, however, to make GUICS similar in appearance to other GUIs in the Windows environment. The usability requirements of WWW interfaces are quite different from those for a GUI (Mandel, 1997).

Pressing buttons with the same name results in the same action anywhere in GUICS. For example, the 'Copy' button always presents a box asking for a new name, and the 'Delete' button always presents a box asking for confirmation. The toolbar buttons 'New', 'Open', 'Copy', and 'Delete' work for projects, scenarios, and datasets in the same way. Windows keyboard shortcuts work in all situations in the same way as in other Windows applications: doubleclick will open an item, and selection of several items from a list is always done by holding down the 'Shift' or 'Ctrl' key while clicking. When multiple variables are available for selection to be displayed in tabular or graphical form, small boxes to checkmark are shown beside the variable names (Fig. 6). As a user works with various models in GUICS, the appearance of the choice lists is always the same as in the examples in Fig. 6—although the content of the list is model-specific.

Forgiveness Features

There are at least three reasons to reinforce interfaces with forgiveness features. Humans are curious and they like to explore interfaces by trial and error; some even intend to break the system. User errors are unavoidable, and hitting the wrong key or selecting the wrong item is a common part of working with interfaces. Also, in the middle of an action, a user may just forget how to continue (Macaulay, 1995; Microsoft Press, 1995).

To facilitate forgiveness, GUICS does not allow any incomplete item to exist. No further progress will occur if the name of a project, scenario, or dataset is not entered. The same will happen if some data category is omitted during the assembly of a scenario. If a simula-

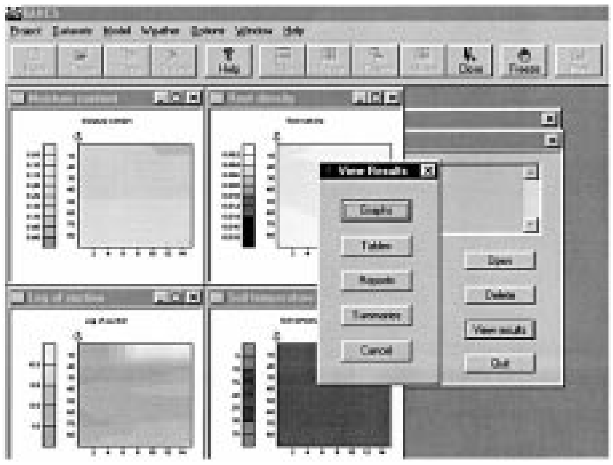


Fig. 5. The 'View results' dialog box appearing after the simulation run or runs have ended. The GLYCIM model-specific two-dimensional graphics show intermediate soil simulation results.

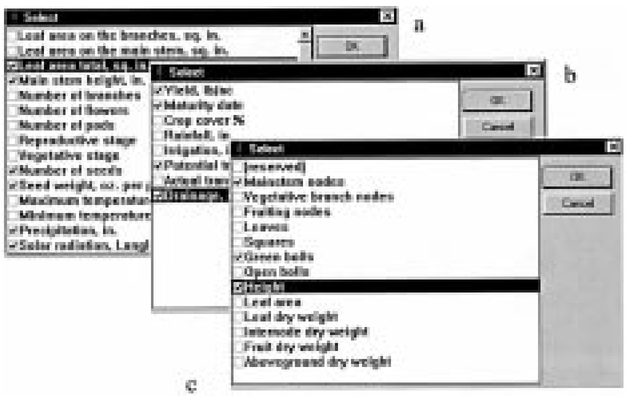


Fig. 6. Similarity in dialog boxes to select variables for graphic and tabular presentation: (a) graphics in GLYCIM, (b) tables in GLYCIM, and (c) graphics in the CPM cotton crop model.

tion run is interrupted, the scenario will have the status 'Assembled' and no results will be available for viewing. All datasets have default contents to prevent the creation of bad data files. Deleting a dataset results in the deletion of all scenarios that included this dataset.

Deleting files in error is the most undesirable of all actions. The 'Delete' key of the keyboard is not used in GUICS except in text boxes, where it can be used to correct typing errors during input. Warnings are issued at any attempt to delete a project, scenario, dataset, or weather station. If a group of items is to be deleted, warnings are issued separately for each item. When a dataset is changed, a dialogue box appears that provides information about the consequences of the choice (Martin and Eastman, 1996).

To help a user remember how to continue in the middle of an action, a reference help is included in GUICS. This is a task-oriented help, which explains all the actions possible in GUICS. It does not duplicate the GUICS User Manual (Acock et al., 1998), which is written to answer "How do I?" questions and shows all 55 windows and dialog boxes that can be encountered in GUICS.

Preventing faulty data from reaching the simulation is a critical part of the forgiveness of interfaces built for agricultural models (Akins et al., 1993; Cox, 1996). Since GUICS is a generic interface, we assume that checking whether the numbers in a data file are reasonable is a function of the model code. GUICS does, however, perform two types of checks that are not model-specific. First, when a dataset is imported, the data structure is checked for consistency with the input data script for this dataset as used in the simulator under consideration (see Plugging a Crop Simulator into GUICS, below). Second, weather datasets need special attention, because the quality of the weather data downloaded from farm weather stations via modems is not always satisfactory.

To automate access to weather data as far as possible, GUICS allows users to download weather from several weather stations and has two versions of the weather data for each station. One version is the raw binary data downloaded from the stations, and the other is the ascii file used as input by the crop simulators. The format for both files is specified to GUICS in a script file (explained in more detail below, in Documenting Input and Output Datasets). All weather stations accessed by GUICS to run a particular model must have the same format. The days for which weather data are available and the gaps in the sequence are listed each time the weather dataset is highlighted (Fig. 7). To extend the ascii file, the user has to refresh it from one of the binary files in his or her possession. To fill a gap, the user has the option of patching the ascii file from a binary file downloaded from another weather station (perhaps that of a friendly neighbor) programmed in the same way as the station used before. The 'Refresh' and 'Patch' buttons are present in the weather data dialog box (Fig. 7) but are absent in the dialog boxes for other data.

Feedback Features

The absence of feedback from software is disconcerting. Users need to know that the software is responding to their actions, and the feedback needs to be as close to the user's action as possible (Bolte et al., 1991). No user action in GUICS is left without feedback given by a visible change in a list of items, by a change in the appearance of the screen, or by a message. Humans prefer nonanthropomorphic, nonthreatening, affirmative, and nonpatronizing messages from computers (Macaulay, 1995). We have tried to select this type of wording in GUICS messaging, and the content of the messages was a separate issue in the design evaluation (see below). The two longest processes that GUICS manages are the crop simulation itself and the downloading of weather data. Progress is shown to the user in each case, respectively, as the date of the day being simulated, or the number of data blocks downloaded.

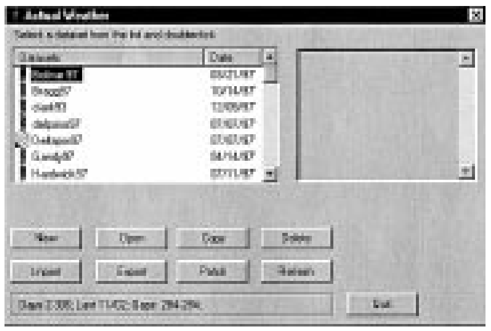
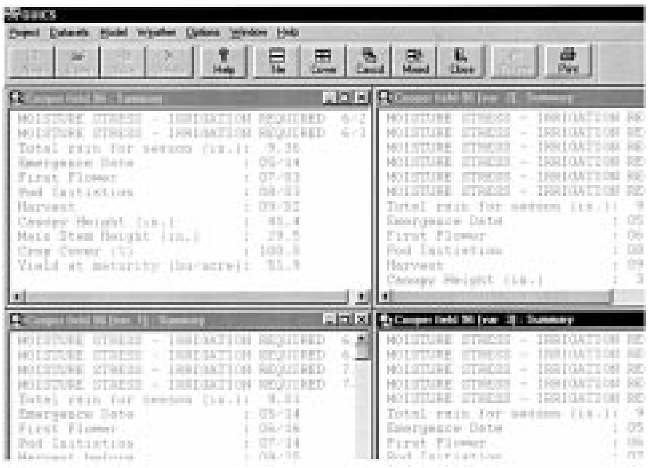


Fig. 7. A dialog box to manipulate weather datasets.

Simplicity Features

The *easy to learn and easy to use* requirement presents the challenge of balancing maximum functionality and maximum simplicity while reducing the presentation of information to the minimum required to communicate adequately (Microsoft Press, 1995). This requirement has a lot to do with the functioning of human short-term memory, which typically can cope with no more than seven or eight options (Mandel, 1997). The number of active buttons or available menu options in GUICS never exceeds this number. The GUICS screens contain the minimum number of words to read (e.g., Fig. 3). Another simplification technique implemented in GUICS is clustering or progressive disclosure of information: large amounts of information are broken into smaller clusters that are easier to remember (Martin and Eastman, 1996). The two-tier organization of the information in GUICS ('project scenario', 'data category dataset') discloses information progressively, reducing the amount available for a user to work with at any one time. Wizards for assembling new scenarios and for importing new datasets serve the same purpose.

Varying scenarios is another example of progressive disclosure in GUICS. Often users want to vary one aspect of a scenario, that is, to create one or more new scenarios which differ from the original in only a single category of data. An example is creating several scenarios that differ from the original one in weather data, while the datasets in all other categories (soil, cultivar, etc.) are kept the same in all variants. The button 'Vary' shown in Fig. 3 leads first to a list of data categories and then, after the category is chosen, a list of datasets is presented for selection. After the selection is made,



```
MOISTURE STRESS - IRRIGATION REQUIRED 6/2
MOISTURE STRESS - IRRIGATION REQUIRED 6/1
Total rain for season (in.):  9.36
Emergence Date              : 05/14
First Flower                : 07/03
Pod Initiation              : 08/03
Harvest                     : 09/30
Canopy Height (in.)         :  41.4
Main Stem Height (in.)      :  29.5
Crop Cover (%)              : 100.0
Yield at maturity (bu/acre):  51.9
```

```
MOISTURE STRESS - IRRIGATION RE
MOISTURE STRESS - IRRIGATION RE
MOISTURE STRESS - IRRIGATION RE
MOISTURE STRESS - IRRIGATION RE
MOISTURE STRESS - IRRIGATION RE
MOISTURE STRESS - IRRIGATION RE
Total rain for season (in.): 9
Emergence Date             : 05
First Flower               : 06
Pod Initiation             : 08
Harvest                    : 09
Canopy Height (in.)        : 3
```

```
MOISTURE STRESS - IRRIGATION REQUIRED 6
MOISTURE STRESS - IRRIGATION REQUIRED 6
MOISTURE STRESS - IRRIGATION REQUIRED 7
MOISTURE STRESS - IRRIGATION REQUIRED 7
Total rain for season (in.):  9.81
Emergence Date              : 05/14
First Flower                : 06/16
Pod Initiation              : 07/14
```

```
MOISTURE STRESS - IRRIGATION RE
MOISTURE STRESS - IRRIGATION RE
MOISTURE STRESS - IRRIGATION RE
MOISTURE STRESS - IRRIGATION RE
MOISTURE STRESS - IRRIGATION RE
Total rain for season (in.): 9
Emergence Date             : 05
First Flower               : 06
```

Fig. 8. Display of summaries for several scenarios.

the list of scenarios has added to it a scenario with the same name as the original plus the number of the variant in brackets (Fig. 3). The memo of the dataset is appended to the memo of the original scenario, to form memos of the variants and to facilitate their recognition. The results of several scenarios can be viewed simultaneously in the same table, in graphs with different colors, or in summary or report windows shown simultaneously (Fig. 8).

Ease of comprehending output has been cited as a precondition to the use of decision support tools in agriculture (Akins et al., 1993; Cox, 1996). We adopted the progressive disclosure strategy here, arranging the output in several levels of complexity (tables, summaries, graphs, and full reports) so that users can go as deep as they want in their analysis of results.

PLUGGING A CROP SIMULATOR INTO GUICS

Adding a crop simulator to GUICS is not an operation an end user will perform often. The main challenge here is to minimize the required changes in the code of the simulator. This section describes the solution that we have developed. Our principal assumption is that the simulator reads input files and produces output files. We do not in this version of GUICS accommodate simulators working with data management software.

To make an existing crop simulator capable of working with GUICS, three operations are required. First, the input files and the simulator output may need to be reorganized. Second, the structure of the input and output datasets must be documented. Third, some standard code must be added to the simulator to notify GUICS with an appropriate Windows message when the simulation ends. The GUICS User Manual (Acock et al., 1998) contains details of these operations, and only a brief description is given here.

Modifying and Documenting Input and Output

The objectives of these operations are (i) to arrange for the input and output file names to be read from a single file RUN.DAT and (ii) to produce a PROFILE file that will be used in GUICS to name and display the datasets. The sequence of operations is straightforward:

1. Numbering the input data files.
2. Assuming that each file represents a data category, and giving a descriptive name to each data category.
3. Assigning to the input files of each category a single matching extension; e.g., in these examples soil data files are all given the extension .SOI.
4. Making the code read the input file names as character strings from the RUN.DAT file in the same sequence as the input files were numbered.
5. Making the code read the input data in free format.
6. Modifying the output of the simulator to produce one or more of the following four files: (i) a Graphics file that contains columns of daily calculated variables showing crop or soil status; (ii) a Summary file, which is a relatively small text file containing condensed information about the simulation results; (iii) a Report file, which is a text file containing as much detailed output as needed for analysis of a particular simulation; and (iv) a Table file, which contains the most important numerical results of the simulation, such as yield, total irrigation, total rainfall,
7. Creating a file named PROFILE that describes the input datasets and the presence or absence of each of the four output files.

Table 2. An example of the PROFILE file.

Entry	Meaning
Profile for the model of soybean crop	File content description
GLYCIM	Model's short name
6	Number of data categories
Weather	Data category short name
Actual Weather	Data category descriptive name
.WEA	File name extension
Weather	Data category short name
Forecast Weather	Data category descriptive name
.WEA	File name extension
Soil	Data category short name
Soil	Data category descriptive name
.SOI	File name extension
Initials	Data category short name
Initialization	Data category descriptive name
.INT	File name extension
Variety	Data category short name
Variety	Data category descriptive name
.VAR	File name extension
Irrigate	Data category short name
Irrigation	Data category descriptive name
.DAT	File name extension
1	Graphics file present
1	Table file present
1	Summary file present
1	Report file present
1	Model specific graphics present

An example of the PROFILE file for the soybean crop simulator GLYCIM is shown in Table 2. Names given to files should be short, because they are used in GUICS in window titles and in lists.

Documenting Input and Output Datasets

GUICS script files are used to describe each item in each of the model's input and output files. Separate a

```
'BOSKET4.HYD ' 'Bosket sandy loam,DBES, F9, Stoneville,MS 1989   '
 3
.6492e-04 .132e+00 .4574e+02 .459e+00 .381e+00 .380e-01 .1299e+01 3.57e+00 3.476e+00
.6492e-04 .132e+00 .4574e+02 .459e+00 .381e+00 .380e-01 .1299e+01 3.57e+00 3.476e+00
.46000e+02 .505e+00 .000e+00 .10000e+01 -.10000e+00  -0.01e+0 999    2130509298
26 .6492e-4 .132e+0 .4574e+2 .459e+0 .381e+0 .918e-1 .1299e+1 3.57e+0 3.476e+0 -.532e-1
  53   10    -2130509254
71 .1231e-2 .165e+0 .3868e+2 .495e+0 .297e+0 .137e-8 .1191e+1 3.12e+0 5.293e+0 -.433e-3
  56   10    -2130509210
201 .2834e-3 .143e+0 .4184e+2 .505e+0 .362e+0 .105e+0 .1162e+1 3.12e+0 5.293e+0 -.30e-1
  75   5        0
```

b

```
Script for Soil Data file for Glycim
@0
Database name
char(%s/)
@1
Soil name and location
char(%s/)
@2
Number of soil horizons
int(%3d/)
@3
Unused
dummy()
@4
Unused
dummy()
@5
Unused
dummy()
@6
Table of soil properties
table(14,&2)
@7
Maximum depth of soil horizon, cm
float(%4.0f)
@8
Hydraulic diffusivity at -15 bars, cm^2/day
float(%11.4E)
@9
Volum. water content at -15 bars
float(%7.4f)
@10
.......
```

Fig. 9. The script file for GLYCIM soil datasets: (a) a soil dataset and (b) script for such files.

script files are written for each input data category, and also for Graphics and Table files. To develop a grammar for scripts that would accommodate the usual structure of simulator input files, we reviewed more than 50 crop simulators.[1] We found that a very simple grammar served to fully describe the data types, positions, and names to be used in the dataset editor. An example of applying this grammar to GLYCIM soil files is described below.

A typical GLYCIM soil hydrology file is shown in Fig. 9a. It begins with two text lines. This is followed by the number of horizons (three, in this case). The next three lines are inherited from previous versions and are kept to allow us to use the datasets with old versions of the model. A table containing 14 columns and three rows follows. The number of the rows is the same as the number of soil horizons. The variables in the columns are depth to base of soil horizon (cm), hydraulic diffusivity at -15 bar (-1.5 MPa) (m^2 d^{-1}), volumetric water content at -15 bar, slope of log (hydraulic diffusivity) vs. volumetric water content, saturated water content, volumetric water content at field capacity, volumetric water content of air-dry soil, bulk density (g cm^{-3}), the exponent in the water retention curve equation, saturated hydraulic conductivity (cm d^{-1}), water potential of air entry (bar), sand content (%), clay content (%), and an integer pointer (not used). The beginning of the script file for this soil hydrology dataset is shown in Fig. 9b. The description of each entry in the dataset occupies three lines: line 1, '@' followed by the entry number; line 2, the name of the entry; line 3, the datum type followed by datum format in parentheses. Note that the formats in the scripts are used to display data in the dataset spreadsheet editor and not to read data from the original data files. Soil files of other models have other parameters, and the script file will be different. As the script is produced, GUICS will handle those different files.

We found that no more than two hours total were needed to write and test scripts for any of the crop simulators we have reviewed.

Bracketing a Crop Simulator

We have worked with simulators coded in C++ and Fortran dialects, although any other language can be used if the compiler produces a Windows application (i.e., a program that uses Windows resources via Windows API). We have used Microsoft C/C++ and Microsoft Fortran 90. Programs written in dialects of Fortran 77 were recompiled with Fortran 90. In general, it is not necessary to use Microsoft compilers to produce Windows applications; for example, GUICS can also run a simulator compiled using a Borland compiler.

A Fortran program must be transformed to become a subroutine without parameters. We supply a standard WinMain program that (i) provides an interface with Windows, (ii) starts the aforementioned subroutine, and (iii) sends a final message to GUICS and returns.

A program written in C to run under Windows needs only to be updated to include the final message. A pro-

[1] The list of simulators is available from the corresponding author upon request.

gram written in C to run under DOS needs changes analogous to those for a Fortran program. We do not supply a WinMain program for DOS-C simulators.

We also recommend that the simulator be modified to include sending the current date being simulated to GUICS for display.

Using Simulator-Specific Graphics

Graphics specific to a crop simulator can be incorporated in GUICS by transforming the simulator and the graphics into dynamic link libraries. This is reflected in the PROFILE file of the simulator, and by including the corresponding item in the 'Options' drop-down list on the menu bar. Since graphics may not be needed for every run, the options include turning the graphics DLL on and off.

DESIGN EVALUATION AND DISCUSSION

Prototyping graphical user interfaces (GUIs) and evaluating the prototypes is the prevailing strategy in GUI development (Martin and Eastman, 1996; Redmond-Pyle and Moore, 1995; Mandel, 1997). We evaluated the GUICS prototype by giving hands-on experience to a group of end users. These users were seven farmers from five southern states who already had experience with WINGLY, a GLYCIM-based decision support tool for soybean growers. WINGLY had been developed to warn soybean farmers when water stress was affecting the crop and to estimate the effects of various scenarios on soybean yield. Over a two-year (1991–1993) trial period, WINGLY users reported up to 400% increase in irrigation water use efficiency and up to 40% increase in yield (Manning, 1996).

To evaluate GUICS, we used the general guidelines of usability testing (Rubin, 1994) and conducted observational interviews (Martin and Eastman, 1996). Acceptance was of concern, because users often prefer a familiar interface to the one that is supposedly improved (Rudisill et al., 1996). In the interview, we (i) outlined the new features of the interface, (ii) demonstrated how to get warnings of stress effects on yields (as shown in Fig. 8), (iii) asked the user to go through the whole process of crop simulation by himself, and recorded all difficulties that he experienced, (iv) asked users to point out any inconveniences, and discussed with them the usability of GUICS, (v) asked whether the users would prefer to keep WINGLY or to get GUICS. We also asked about the need for mapping tools for input and output, and about the need for resources to be accessed through Internet.

Two of the growers were able to go through the process of crop simulation immediately after our demonstration. Wizards appeared to be a big help. The main difficulties in operating the GUICS prototype arose because we had not been consistent in implementing Windows shortcut conventions and in including error messages in the wizards. Two users found the icons confusing. Guidelines on naming datasets and scenarios and on writing memos were requested. None of the users saw an advantage in combining various scenarios into projects for on-farm use. One of the users indicated that the accumulation and collection of garbage data might become an issue as data manipulation becomes easier. An automated update of ascii weather files was requested. Tools to generate several predicted weather files were desired. The ability to have several crop models running under the same interface was welcomed. None of the users objected to replacing WINGLY by GUICS, provided they were given a converter to transform WINGLY data files to GUICS data files.

Discussion of the need for mapping tools revealed a variety of interests, mostly related to the familiarity of the users with precision farming technology. All users agreed that it would be convenient to use a mapping unit as the kernel of a project relating soil, weather, management, and cultivar data. Two users thought that the soil mapping units could be kernel units, whereas one user thought that a field might be the more appropriate unit. Three users had yield monitors and thought that crop simulation should be related to yield map analysis, and that the appropriate tools should be integrated into GUICS. One user pointed out that the accumulation of data eventually might make desirable a database that would support complex queries. None of the users was aware of Internet resources that could help them to use GUICS as a decision support tool, although all them expressed interest in information on such resources.

GUICS was amended as a result of these interviews, and the amended version was used on 15 farms in 1997 and 1998. The only problems that we encountered were errors in downloads of weather data leading to faulty weather files being used in simulations.

The GUICS version 1.7 and the GUICS User Manual (Acock et al., 1998) are available at the anonymous FTP site asrr.arsusda.gov from the directory /pub/guics1_7/. The contact person is the corresponding author of this paper. The code is written in C++ using Microsoft Foundation Classes. GUICS runs under Windows 95, Windows 98, and Windows NT 4.0. GUICS with the GLYCIM example requires about 4.5 Mb of hard drive space.

The main advantage of GUICS is that a farmer can use the same interface to run different crop simulators corresponding to the various crops in the farm operation. The farmer can access whatever level of detail is needed. GUICS proved to be easy to learn and easy to use.

ACKNOWLEDGMENTS

Part of the work was funded by the United Soybean Promotion Board through cooperative agreements between the USDA-ARS, the University of Maryland, and the University of Florida. The code of GUICS was written by E.V. Mironenko. The icons were designed by L. Pachepsky. The online help was developed by Jackson Fisher. The help of Tom Hodges is acknowledged. The advice of M.C. Acock, D. Bragg, W.C. McCain, J.E. Hester, K.B. Hood, D.P. Madison, Jr., J. Mullens, W.G. Ragsdale, D.J. Timlin, and H.T. Wilder, Jr., is greatly appreciated. We extend our thanks to the 62 researchers who participated in the beta-testing of GUICS.

REFERENCES

Acock, B., E.V. Mironenko, Ya.A. Pachepsky, and V.R. Reddy. 1998. GUICS: Graphical user interface for crop simulations with Windows 95. Version 1.7. User's manual. USDA-ARS, Remote Sensing and Modeling Lab., Beltsville, MD.

Ascough, J.C., II, and D.L. Hoag. 1995. Farm computer use in the Great Plains. USDA-ARS-NPA, Great Plains Systems Res. Unit, Fort Collins, CO.

Akins, D.C., T.L. Wagner, J.T. Willers, R.L. Olson, and M.R. Williams. 1993. New graphical user interface for the rbWHIMS insect management expert system. p. 1041–1043. *In* Proc. Beltwide Cotton Conf., 1993. Am. Cotton Council, Memphis, TN.

Bolte, J., J. Fisher, and D. Ernst. 1991. Coupling a graphical user interface with an object-oriented simulator for salmon hatcheries. Winter Pap. 917513. ASAE, St. Joseph, MI.

Cox, P.G. 1996. Some issues in the design of agricultural decision support systems. Agric. Syst. 51:1–27.

Christakos, G. 1998. Spatiotemporal information systems in soil and environmental sciences. Geoderma 85:141–180.

Greer, J.E., S. Falk, K.J. Greer, and M.J. Bentman. 1994. Explaining and justifying recommendations in an agriculture decision support system. Comput. Electron. Agric. 11:195–214.

Hoogenboom, G., J.W. White, J.J. Jones, and K.J. Boote. 1994. BEANGRO: A process-oriented dry bean model with a versatile user interface. Agron. J. 86:182–190.

Humphries, S.W., and S.P. Long. 1995. WIMOVAC: A software package for modelling the dynamics of plant leaf and canopy photosynthesis. Comput. Appl. Biosci. CABIOS 11:361–371.

Jones, L.R. 1992. The current state of human–computer interface technologies for use in dairy herd management. J. Dairy Sci. 75:3246–3256.

Landivar, J.A., G.W. Wall, J.H. Siefker, D.N. Baker, F.D. Whisler, and J.M. McKinion. 1989. Farm application of the model-based-reasoning system GOSSYM/COMAX. p. 688–694. *In* J.K. Clema (ed.) Proc. 1989 Summer Computer Simulation Conf., Austin, TX. 24–27 July 1989. Soc. for Computer Simulation, San Diego, CA.

Macaulay, L. 1995. Human–computer interactions for software design. Int. Thomson Computer Press, London.

Mandel, T. 1997. The elements of user interface design. John Wiley & Sons, New York.

Manning, E. 1996. GLYCIM: Crop model for soybeans. Progressive Farmer 11:33.

Martin, A., and D. Eastman. 1996. The user interface design book for the applications programmer. John Wiley & Sons, New York.

Reddy, V.R., Ya.A. Pachepsky, F.D. Whisler, and B. Acock. 1997. A survey to develop a decision-support system for soybean crop management. p. 11–18. *In* Proc. ACSM/ASPRS Annu. Conv. 1997. Vol. 4. ACSM (Am. Congr. on Surveying and Mapping) and ASPRS (Am. Soc. for Photogrammetry and Remote Sensing), Bethesda, MD.

Redmond-Pyle, D., and A. Moore. 1995. Graphical user interface design and evaluation (GUIDE). Prentice Hall, Englewood Cliffs, NJ.

Rewerts, C.C., B.A. Engel, J.B. Rogers, and D.D. Jones. 1989. An end user interface for agricultural software. Winter Pap. 89-7607. ASAE, St. Joseph, MI.

Rubin, J. 1994. Handbook on usability testing: How to plan, design and conduct effective tests. John Wiley & Sons, New York.

Rudisill, M., C. Lewis, P.B. Polson, and T.D. McKay (ed.) 1996. Human–computer interface design: Success stories, emerging methods, and real-world context. Morgan Kaufmann, San Francisco, CA.

Srinivasan, R., and B.A. Engel. 1994. A spatial decision support system for assessing agricultural nonpoint source pollution. Water Resour. Bull. 30:441–452.

Testezlaf, R., B.M. Jacobson, F.S. Zazueta, and T.H. Yeager. 1996. A graphical user interface for real time irrigation control in greenhouses. Proc. Soil Crop Sci. Soc. Fla. 55:59–62.

Microsoft Press. 1995. The Windows interface: Guidelines for software design. Microsoft Press, Redmond, WA.

Thornton, P.K., and G. Hoogenboom. 1994. A computer program to analyze single-season crop model outputs. Agron. J. 86:860–868.

Wood, L. (ed.). 1998. User interface design: Bridging the gap from user requirements to design. CRC Press, Boca Raton, FL.

Van Evert, F.K., and G.S. Campbell. 1994. CropSyst: A collection of object-oriented simulation models of agricultural systems. Agron. J. 86:325–331.

Waldman, S.E., and R.W. Rickman. 1996. MODCROP: A crop simulation framework. Agron. J. 88:170–175.

Wu, X. 1992. Two strategic planning models with graphical user interface for forest and wildlife habitat management and environmental assessment. p. 555–556. *In* Proc. Soc. Am. Foresters Natl. Conv. 1992.