Contents

To get help on an item, click on the icon or the underlined text.

Rules of the Game and
Strategy Hints

Using a Mouse to
play Link Four

Using the Keyboard to
Play Link Four

Menu Commands and
Dialog Box Reference

Designing Pattern & Rule Files
for Rule-Based Play

Source Availability for
Borland Pascal Programmers

Release Notes and
Version History

Known Bugs & Problems

The **Search** button at the top of the Help Window can be used to search for topics.

Rules & Strategy Hints

LINK FOUR RULES

The game of Link Four is played by two players who take turns stacking marbles in one of seven possible stacks (red always plays first). Each stack may grow to a maximum height of seven marbles. The object of the game is to form a "link" of four marbles, a consecutive sequence of four adjacent marbles. A link may be formed horizontally, vertically, or diagonally.

Note that selecting a move involves only the selection of a column in which to stack a marble. The marble is always played immediately on top of the stack. That is, the marble is played on the lowest available row in the selected column.

STRATEGY HINTS

Near the beginning of the game, it is usually a good idea to play near the center of the board. Controlling certain positions in the center column can be a key to winning the game. Having two adjacent pieces of your color stacked in the center column can often be an extremely advantageous position as it can allow one to setup a win in two adjacent vertical cells, a situation which cannot be easily defended against.

Although the pieces near the center of the board are of the utmost importance, don't neglect the columns to the sides. In any given game, one or two of the cells in the outer columns are critical positions. It is important to learn to recognize these positions and use them to one's advantage (or at least attempt to prevent the opponent from doing so). Looking ahead several moves and carefully using the side positions can often be a winning strategy.

Playing Link Four with a Mouse

The best way to play Link Four is with a mouse (although it can be played with only the keyboard ... see Keyboard Use for details). Whenever it is a human player's turn to play, a "turn indicator" marble will be displayed. For the red player the turn indicator marble is initially displayed in the upper left of the window, and for the blue player it is initially displayed in the upper right. If the Suggest Move command has been used then the turn indicator marble will be positioned over one of the playable columns in the playing area.

To play a marble on a given stack, grab the marble and drag it onto the playing area and release it anywhere above the highest marble in the desired column. The playing grid can be used to help determine which column the marble will be dropped in when released.

While dragging the marble it is possible to release it outside of the playing area. Doing so will cause the turn indicator marble to be redisplayed in the position it was in before it was grabbed. This is sometimes convenient when you would like a little more time to consider a move and do not want to continue holding the mouse button down while doing so.

Grabbing an object in Windows involves positioning the mouse cursor over the object and holding down the left mouse button. Most objects are typically grabbed so they can be dragged to another location.

Dragging an object in Windows involves first grabbing the object and then moving the mouse in the application window. The object being dragged moves with the mouse and can be dropped in a new position.

Keyboard Use

Instead of using a mouse to drag the marbles to a column and drop them, the **Left**, **Right**, **Down**, and **Enter** keys can be used to manipulate the marbles.

The **Left** and **Right** arrow keys will move the marble across the top of the playing grid to allow a column to be selected.

The **Down** and **Enter** keys drop the marble into the column above which it is currently positioned.

Menu Commands

GAME MENU

- New
- Undo Move
- Suggest Move
- Exit

OPTIONS MENU

- Red Player
- Blue Player
- Background...
- Playing Grid Color
- Undo Enabled

Patterns & Rule-Based Play

OVERVIEW

Patterns & rule-based play allows the specification of a list of *patterns* which can occur on the playing board. To decide on a move, the computer searches the board for the patterns, and if a pattern is matched, one or more accompanying *rules* specify an *action* to be taken (such as making an immediate move or eliminating a column from consideration on the current move because it leads to an immediate loss situation).

RULE FILES

A rule file is simply a text file (default extension is .RUL) which contains sets of patterns and the rules associated with them. See the example rules or the complete sample ruleset for examples, or view one of the sample rulesets distributed with the Link Four program.

The overall structure of a rule file is that of a set of patterns followed by a set of rules, then another set of patterns followed by a set of rules, and so on. Sets of patterns are delimited with the keywords PATTERNS and ENDPATTERNS, which must appear on a line by themselves. Sets of rules are delimited by the keywords RULES end ENDRULES, which also must appear on a line by themselves. Following the end of all pattern/rule sets, the keyword ENDFILE must appear on a line by itself.

Within sets of patterns, the patterns themselves (which are rectangular arrays of pattern symbols ... see below for more detail) should be delimited by at least a single blank line. Within sets of rules, rules are placed on per line and require no other delimiters. Finally, any line in a rule file which begins with a semicolon ";" character is ignored, allowing comments to be placed in the file.

PATTERNS

A pattern is a rectangular grid (up to 7 by 7, the size of the playing board) of *pattern symbols*. Here is a list of the pattern symbols, along with their meanings, which are used to build patterns:

Symbol	Description
X	A marble of the current player's color
O	A marble of the opposing player
+	A marble of either color
-	An unoccupied position
*	A "playable" board position
?	Don't care (i.e. matches anything)
a..z	A "variable" to be used in a rule for this pattern

The **X** symbol is used in a pattern to specify a board position occupied by the current player (that is, the player that the computer is currently performing a pattern search on behalf of). The O symbol is used in a pattern to specify a board position occupied by the opposing player.

The **+** symbol is used in a pattern to specify an occupied position, regardless of which player's marble is occupying it. The **-** symbol is used in a pattern to specify a board position which is unoccupied. The ***** symbol is used in a pattern to specify a board position which is unoccupied but "playable". That is, the position is unoccupied, but either the position is on the bottom row or the position immediately below it is occupied. The **?** symbol is the "don't care" symbol. That is, when this symbol is used in a pattern, the symbol matches any position, regardless of whether it is empty or occupied by a marble of either color.

The **a..z** symbols, that is, the lower case letters "a" through "z" (case is significant) represent symbols whose values are variable, and will be specified in the rule statements which follow the current set of patterns. In any given pattern search using a particular rule, if a variable is not given a value, it defaults to the "?" symbol.

RULES

A particular set of rules always applies to the set of patterns which immediately precede it in the rule file. The first part of a rule is a list of variables and the pattern symbols to which they should be bound for the purpose of searching the playing board. If a position is found on the playing board in which all of the variables in a given rule are matched successfully in one of the patterns, the rule also specifies an action to be taken.

The syntax for specifying variables and the pattern symbols to which they should be bound is as follows (for those unfamiliar with common syntax notation, the square brackets indicate an optional part of the statement):

```
<variable>=<symbol> [,<variable>=<symbol> [,...]] :
```

Here are some examples:

`a=+,b=*:` legal
`a=?,b=X:` legal
`a=X b=O:` not legal - *commas* must be used as delimiters
`a=b,c=*:` not legal - "b" is not a valid pattern symbol
`A=*,B=?:` not legal - variables must be *lower case*
`a=X,b=O;` not legal - list must be terminated by a *colon*

ACTIONS

The second part of a rule is the action or actions to be taken if a match is found. The actions which can be specified by a rule are to (1) play in a specified column immediately, (2) remove one or more columns from further consideration during the current move, or (3) increase or decrease the priority of playing in one or more columns. Multiple actions can be specified for a single rule by separating them with spaces.

The syntax for specifying actions (which immediately follow the colon for a rule specification) is as follows:

```
<variable>[  <priority> | ![<priority>]  ] [...]
```

The actions to be taken for a rule are listed following the colon which terminates the variable bindings for a rule. An action is a variable name optionally followed by a number (from 1 to 3), an exclamation point "!", or both an exclamation point and a number. The variable name indicates the column that the action refers to. That is, the action being specified refers to the column in which the variable named in the action is located. This column is termed the *action column*.

An action which consists simply of a variable name, termed a "play" action, indicates that a marble should be played immediately in the action column. An action variable which is followed by a number from 1 to 3 indicates that the priority of the action column should be increased appropriately. See the discussion below on priorities for details about how priorities are used to decide on a move.

An action which consists of a variable name followed by an exclamation point, termed a "don't-play" action, indicates that the action column is to be removed from further consideration for the current move. This is most commonly done to eliminate a column which, by playing in it, would lead to an inevitable loss. An action variable which is followed by an exclamation point and then a number from 1 to 3 indicates that the priority of the action column should be decreased appropriately. The section below on priorities details how priorities are used in deciding on a move.

Following are some examples of complete rule and action specifications:

`a=+,b=*:b`	make immediately play in column b
`a=?,b=X:b`	legal but column b could be full - use `*` instead
`a=*,b=O:a!`	remove column a from further consideration this move
`a=X,b=*:b1`	increase the priority of playing in column b
`a=*,b=*:a!1`	decrease the priority of playing in column a

PRIORITIES

Actions are what define how the computer reacts when specific patterns and rules are located on the playing board. An action either specifies a priority or it does not. A "play" action which does not specify a priority causes the computer to react by making that play immediately. A "don't-play" action (i.e. an action column followed by an exclamation point) which does not specify a priority indicates that the column should be removed from further consideration during the present move. Actions with priorities are treated somewhat differently, however, as explained here.

Priorities are numbers from 1 to 3, with 1 being the highest priority and 3 being the lowest. A play action followed by a priority indicates that the priority of playing in this column should be increased appropriately. If the pattern search proceeds to completion and no "immediate" play action (i.e. there was no priority in the action) was encountered, then the column with the highest priority is selected for play. A don't-play action followed by a priority indicates that the priority of playing in the specified column should be reduced by the indicated amount.

The priorities of each column are tracked by keeping a count of how many 1's, 2's, and 3's are added to the column (or removed from the column, as the case may be). The column with the highest number of 1's is then selected for play. If two or more columns have an identical number of 1's then the 2's are checked, and finally the 3's if necessary. If two or more columns should happen to have identical sets of priorities, then the computer chooses one arbitrarily for play.

THE SEARCHING ALGORITHM

Patterns and rules are searched for in the order they are specified in the rule data file. This means that rules considered to be most important should be placed first in the data file. Starting with the first pattern in a set, the computer attempts to match the pattern (as well as its horizontal reflection) with the first rule in the associated rule set. The search then proceeds with the next pattern in the set and so on until every pattern has been tried with the first rule. The search proceeds with the next rule in this same manner until all patterns in the set have been checked against all rules.

The search then proceeds to the next set of patterns and rules and the process is repeated. The search proceeds in this manner until the end of the pattern and rule sets. (The search does terminate immediately, however, if a rule is matched which contains an "immediate" play action).

At the conclusion of the pattern searching process, the computer selects the move with the highest priority (if no "immediate" move was made) as described above under the section on priorities. Ties in priorities are broken arbitrarily. Also, if all possible moves were removed from consideration by actions taken during the search, a valid move will be selected arbitrarily by the computer.

Example Patterns & Rules

Following are a few example pattern and rule sets which illustrate their use. For a more complete example, see the sample ruleset. Following each of the rule and pattern sets is a paragraph which explains the example.

```
PATTERNS

a
X
X
X

ENDPATTERNS

RULES

a=* : a

ENDRULES
```

The above example illustrates a very simple single pattern and single rule which search for a vertical column of three of the current player's pieces with an open position just above them. Playing atop them would lead to an immediate win, so the action specifies an immediate play in column a. Note that since a similar pattern could be used to search for a horizontal win, an entirely new pattern set need not be developed, but the horizontal pattern need only be included in the pattern list as shown in the next example.

```
PATTERNS

a
X
X
X

aXXX

ENDPATTERNS

RULES

a=* : a

ENDRULES
```

This example is similar to the previous one, except now we have a rule which will select a win by extending any vertical *or* horizontal link of three pieces which can be extended to a fourth. (Note that this rule covers the situation `XXXa` as well, as all patterns are tried both as specified and horizontally reflected.) The next example shows a complete pattern and rule set which will find any winning move presently on the board

```
PATTERNS

aXXX
```

```
XaXX

a
X
X
X

a???
+X??
++X?
+++X

X???
+X??
++X?
+++a

X???
+a??
++X?
+++X

X???
+X??
++a?
+++X

ENDPATTERNS

RULES

a=* : a

ENDRULES
```

This example illustrates a set of patterns and a single rule designed to search the board for any winning play and take it immediately if found. (Note that this is not the only possible solution for doing this, but combining several patterns with a few rules keeps rule files simpler and shorter.) Note that because the computer also checks for horizontal reflections of each pattern but not vertical reflections, it was necessary to include additional patterns for the diagonal wins.

In an actual rule file, a rule such as this would most likely be placed first in the file because pattern/rule sets are processed in the order they are encountered in the data file. Hence, the first thing we would like to do is check for and take a winning move. If no winning move is found however, we will most certainly need to search for moves which, if not taken, leave the opponent open for a win on the next move. The pattern/rule combination to accomplish this, however, will appear nearly identical to the one in this example, with only the X's changed to O's. As shown in the next example, however, we can actually expand the pattern/rule combination of the present example to handle checking for both win and loss situations!

```
PATTERNS

abcd

a
b
```

```
c
d

a???
+b??
++c?
+++d

d???
+c??
++b?
+++a

ENDPATTERNS

RULES

a=*, b=X, c=X, d=X : a
a=X, b=*, c=X, d=X : b
a=*, b=O, c=O, d=O : a
a=O, b=*, c=O, d=O : b

ENDRULES
```

This example illustrates a complete pattern/rule combination which first checks for any possible winning move and takes it. If no winning move is found then a search is made for any move which, if not taken, leaves the opponent open for a win on the next move. Again if such a move is found it is taken immediately. It is important to note that this works because a search is performed using each pattern with the first rule, then each pattern with the second rule, and so on. Therefore the rules which search for a winning move are placed first, and the rules which attempt to block an opponent win are placed last.
This example is worth studying and understanding as it illustrates how to effectively combine patterns and rules to perform very powerful searches. These same actions could have been accomplished using several different patterns and rules for each case, but the method shown here is much more succinct and effective. A pattern/rule combination similar to (or identical to) this example should appear at the beginning of any rule data file.

```
PATTERNS

-OOO
a???

O-OO
?a??

-???
aO??
??O?
???O

O???
?-??
?aO?
???O

O???
```

```
?O??
??-?
??aO

O???
?O??
??O?
???-
???a

ENDPATTERNS

RULES

a=* : a!

ENDRULES
```

This example illustrates the use of a "don't-play" action. In any of the listed patterns, a play in column "a" would lead to an immediate win by the opponent on the next move. Therefore, we need to ensure that any such position is not considered for play during the current move. The patterns do most of the work here, and the single rule says that if position "a" is playable, then remove the column containing "a" from consideration for the present move. For a more complete example of using the don't-play action to eliminate columns which could lead to an immediate loss, see the sample ruleset.

Sample Rule-Based Patterns & Rules Set

This is a complete, though relatively simple, ruleset which would probably play moderately well against a novice player (i.e. it might win a game or two, but not for long as the player gains experience). It does serve as a good starting point for developing rulesets which play very well. A bit of creativity and some experimentation can yield a ruleset which is very difficult to beat. See the example rulesets distributed with Link Four for ideas.

For a set of various examples with explanations regarding their use, see the example patterns & rules.

```
;-------------------------------------------------------------------------
-
; Patterns and rules to check for immediate win or prevent losing on next turn
;-------------------------------------------------------------------------
-

PATTERNS

abcd

a
b
c
d

a???
+b??
++c?
+++d

d???
+c??
++b?
+++a

ENDPATTERNS

RULES

a=*, b=X, c=X, d=X : a
a=X, b=*, c=X, d=X : b
a=*, b=O, c=O, d=O : a
a=O, b=*, c=O, d=O : b

ENDRULES

;-------------------------------------------------------------------------
-
; Eliminate columns which lead to an immediate loss
;-------------------------------------------------------------------------
-

PATTERNS

abOO
```

```
cd++

a???
cb??
?dO?
???O

ENDPATTERNS

RULES

a = O, d = * : d!
b = O, c = * : c!

ENDRULES

PATTERNS

O?-?
+O-?
++-?
++aO

ENDPATTERNS

RULES

a = * : a!

ENDRULES

PATTERNS

O??-
+O?-
++O-
+++-
+++a

ENDPATTERNS

RULES

a = * : a!

ENDRULES

;-------------------------------------------------------------------------
-
; Try to setup or prevent obvious two-way win situations
;-------------------------------------------------------------------------
-

PATTERNS

*abc*
```

```
*????
+a???
++b??
+++c?
++++*

*????
+c???
++b??
+++a?
++++*

ENDPATTERNS

RULES

a=*, b=X, c=X : a
a=X, b=*, c=X : b
a=*, b=O, c=O : a
a=O, b=*, c=O : b

ENDRULES

;-------------------------------------------------------------------------
-
; Last chance rule to prefer various columns
;-------------------------------------------------------------------------
-

PATTERNS

?bca???

ENDPATTERNS

RULES

a = * : a1
b = * : b2
c = * : c3

ENDRULES

ENDFILE
```

Source Code Availability

LINK FOUR PROGRAM DESIGN & PURPOSE

Link Four was programmed using Borland Pascal version 7.0. Link Four was initially designed as an instructional program for a beginning Windows programming class using the Borland Pascal language. As an instructional program, the code has been carefully organized, cleanly written, and contains extensive internal documentation to aid in the Windows programming learning process. It is intended that the source code will ultimately be published in an introductory text on Windows programming using the Borland Pascal language.

Following the initial design and implementation of the Link Four program, a module was added to assist in teaching the artificial intelligence concepts of pattern matching and heuristic search to gifted junior high and high school level students. The resulting Rule-Based Play feature has proven highly successful, not to mention being a great deal of fun.

IMPORTANT PROGRAMMING CONCEPTS ILLUSTRATED

The intent was to design an instructive program which was an entire, moderately sized application, to bring together the fundamentals of Windows programming concepts. In addition to basic program structure, the program illustrates several important and useful Windows programming concepts and techniques, including cooperative multitasking, handling 256 color bitmaps, smooth dragging of bitmap objects, design and integration of Windows help files, and more.

Also, not limited to the realm of only Windows programming, the Link Four program illustrates fundamental techniques of game-playing programs. The computer-based strategy used by the program plays very well, though it is actually quite simple. The strategy of the game was not a primary design issue, but programmers new to game trees and heuristic searching can learn a great deal from the simple but effective algorithms used by the Link Four program.

OBTAINING THE SOURCE CODE

The author of the Link Four program is Ron Pacheco. He holds an M.S. in Computer Science from the University of Missouri-Rolla, and a B.S. in Computer Science from Harding University. Ron presently teaches mathematics and computer science at Harding University, and operates a private consulting business in the central Arkansas area. Ron has been programming under the Windows operating system for over six years, and has recently begun teaching Windows programming to others. The Link Four project is a part of these efforts.

For those interested in using the code as a learning tool, experimenting with the code, or contributing to the Link Four project, the complete Borland Pascal 7.0 source code and all associated files, including all resources and help project files, can be obtained from the author for $15 U.S. For this fee the author grants the right to use any or all of the code in any commercial or non-commercial programming project, so long as modified versions of the Link Four program itself nor any of the source code is distributed without the express written permission of the author.

For those interested, a portion of the fee will be used for postage, packaging, and media, and the remainder will be used by the author in his efforts toward developing an introductory Windows programming text using the Borland Pascal language.

To obtain the latest version of all source files on a high-density, 3.5 inch floppy disk, send a $15 U.S. check or money order, payable to Ronald T. Pacheco, to:

R&R Engineering
Link Four Project

4 Marshall Drive
Searcy, AR 72143-5011

For those with Internet access who are interested in contacting the author, he can be reached via e-mail at pacheco@acs.harding.edu. The author welcomes questions, comments, and suggestions regarding the Link Four project or ideas for developing an introductory Windows programming text based on the Borland Pascal programming language.

Game | New

This option ends the game in progress (if any) and begins a new game using the currently selected player options.

If a new game is started with the undo feature disabled (see Options | Undo), undo will not be available during play. If disabled, the undo feature cannot be enabled during play without ending the current game.

Game | Undo Move

If the undo feature is enabled (see Options | Undo) then the Undo option of the Game menu will back the game up from its current state to the point just before the last play made by a human player. The undo feature is not available if the computer is playing itself.

If two humans are playing then the undo feature (if enabled) will be available during either player's turn and will remove marbles played on the board one at a time in reverse of the order in which they were played.

If a human player is playing against a computer opponent then the undo feature (if enabled) will be available only during the the human player's turn and will remove the last two marbles played (the most recent marble placed by the computer and the most recent marble placed by the human player).

Game | Suggest Move

This command informs the computer that it should suggest a move for a human player (this option is not available when the computer is playing against itself).

If a human is playing against the computer, then the computer suggests a move using the same level of play that the computer is currently using. For example, if the computer is currently playing at the "Expert" level, then it suggests a move by analyzing a move for the human player at this same level.

If both players are human then then the computer always suggests a move using the "Good" level of play.

Game | Exit

This command ends any game in progress and terminates the application.

Options | Red/Blue Player

Each player can be designated as a human player, can be set to one of the four available computer players, or can be designated as a "rule-based" player.

The "Novice Computer" level is intended primarily for those who are new to the game to become familiar with some basic strategy. This level is very fast, but even beginning players will not find it challenging for more than a few games.

The "Good Computer" level plays fairly well and is still quite fast. This level is good for a quick game now and then, but poses no real challenge for a skilled player.

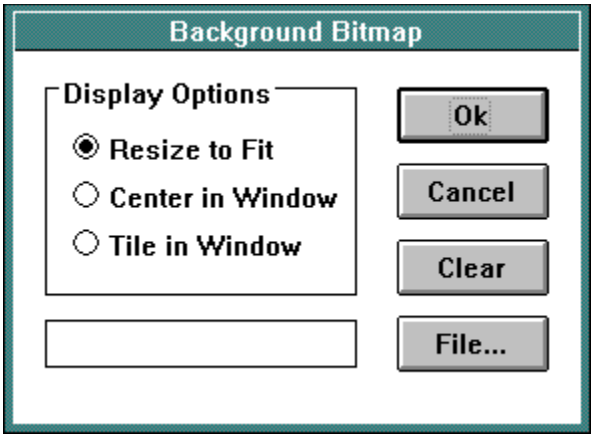
The "Expert Computer" level plays very well while keeping the playing time to a reasonable minimum. The skilled player should find this level quite challenging.

The "Master Computer" level is the best computer opponent and is quite difficult to beat even by the best of players, but the added skill comes at the cost of playing time. On a 33MHz 386 playing time averages about two minutes per computer move.

The "Rule Based" option allows users to design heuristics files consisting of Patterns & Rules which will be used by the computer in determining its plays.

Options | Background...

This command displays the following dialog box used to change the background of the Link Four window.



Click on dialog box items for more information.

Options | Playing Grid Color

The playing grid displayed in the window can be shown in either black (the default) or white. The color selected usually depends on any Background Bitmap being displayed. For certain backgrounds a black grid will show up better, while for others a white grid will show up better.

Options | Undo Enabled

This command toggles the availability of the Undo Move feature while playing a game. This is included for those players who want to *know* that they won for themselves.

If undo is enabled at the start of a new game, it can be disabled at any point during the game. Once disabled, however, the undo feature cannot be reenabled without ending the current game (this keeps us honest).

The display options control how the background bitmap is displayed in the window. The **resize** option causes the bitmap to be scaled to fit inside the window. The **center** option causes the center of the bitmap to be centered in the window (no resizing takes place and the bitmap is trimmed if necessary). The **tile** option causes the bitmap to be replicated so that the window is completely "tiled" with the image.

This box displays the filename of the bitmap currently being used for the background. The value in this box cannot be modified directly, but is updated automatically whenever a new file is selected using the File... dialog.

Applies the current background bitmap file and display options and closes the dialog box.

Cancels any changes made since the dialog box was opened and closes the dialog.

Clears any background bitmap currently selected and returns the background to the current system window color. This button closes the dialog box.

Activates a common file dialog box to allow the selection of a device-independent bitmap file (.BMP) to use as the background.

Release Notes & Version History

Version 1.30

Version 1.30 is the first public release of the Link Four program. Prior to this release, a few copies were distributed with the understanding that no further distribution should take place. The following is a list of changes made in version 1.30:

1. The online help file has been completed.
2. Cooperative multitasking has been added to the rule-based thinking mode.
3. A "thinking" marble is now displayed during moves governed by rule-based searches.
4. A minor bug which caused the window background not be redrawn properly when the window was resized has been corrected.

Version 1.20

The following is a list of changes made in version 1.20:

1. Added a rule-based play system which allows the move search algorithm to be "programmed" by the user. The user specifies a set of board patterns and actions to be taken when those patterns are located. Rule-based play was initially implemented for use in an AI course for gifted and talented junior high school students, where it worked exceptionally well.

Version 1.11

The following is a list of changes made in version 1.11:

1. A minor bug related to screen repainting during a computer move search has been corrected.

Version 1.10

The following is a list of changes made in version 1.10:

1. Cooperative multitasking has been added to allow other processes to run while the computer is "thinking" about its next move.

Version 1.00

Version 1.00 is the first beta test version. Limited distribution only for testing purposes.

Known Bugs & Problems

There are currently no known bugs in the Link Four software. Bug reports should be sent via Internet e-mail to pacheco@harding.edu.