Application Software for Packet Radio

**A *Packet Chess* program is used
to demonstrate the utility of
Application software for Packet Radio**

Robert Taylor, KA6NAN
Dewayne Hendricks, WA8DZP

Introduction

To date, there has been little use made in amateur packet radio of application software to provide a wide variety of services to the packet community. Most packet software delivers a limited range of capabilities such as file transfers, mail and database queries (White Pages). We feel that the use of higher levels of applications software is possible even given the constraints of today's packet speeds, that can offer a wider range of services. We use a program called *Packet Chess* to demonstrate how such services can be delivered. *Packet Chess* demonstrates how it is possible to have a real-time game operate over packet with a graphical user interface and at low data transfer speeds.

Problems with Packet Software

Current packet software is far too text oriented. There are too many commands to memorize and understand. For a new user, the command structure is not intuitive. The emphasis on text makes an 'on-the-air' session more work than fun. A different approach which would be more user friendly would be the use of graphic images. However graphical images are too difficult to utilize because a) current packet is slow, and graphical images would take too long to transmit., and b) there is a lack of standards, which could be used to exchange graphical information.

Some solutions using Application software for packet

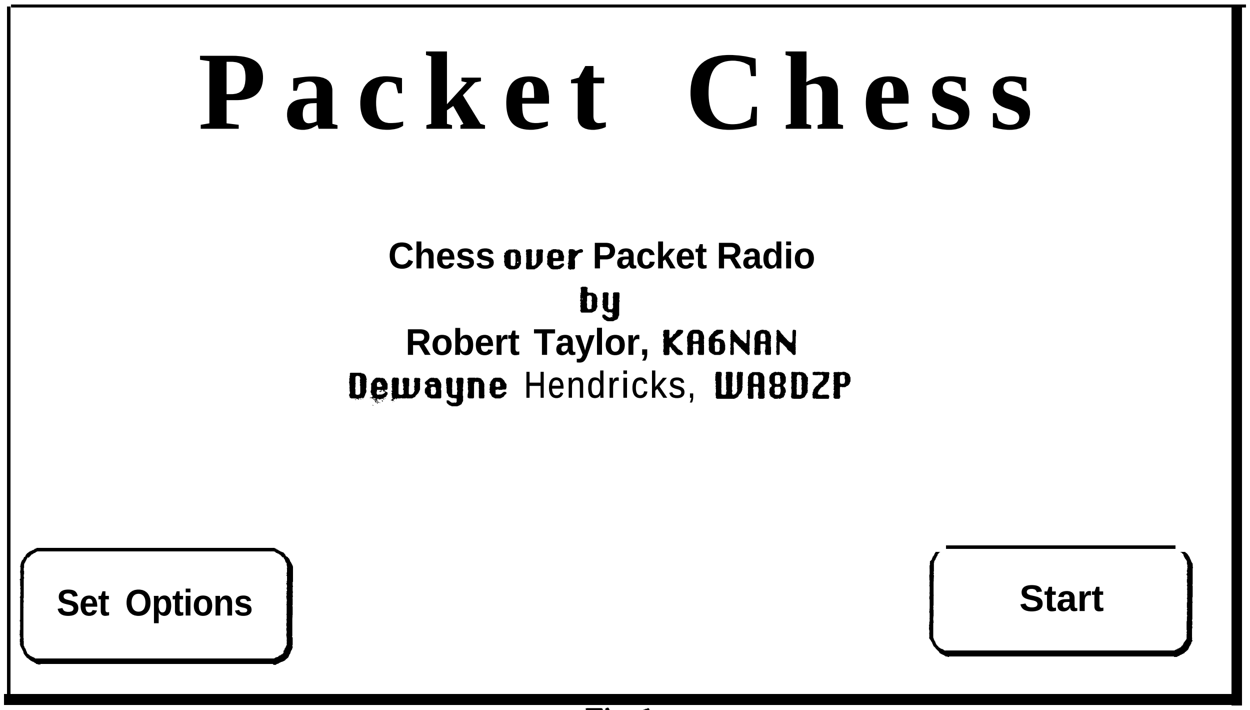
What do we mean by application software for packet? When connected stations utilize similar software, and this software is specifically written to provide a particular service, then we call these programs applications. The *Packet Chess* program described below is an example of this concept, since it is written specifically to facilitate the playing of chess. This applications program addresses the problems we cited above in a number of ways.

By making extensive use of a graphical user interface, it eliminates the need to memorize a large number of commands. You can present the player with the appropriate choices at the appropriate times, making program use much more intuitive. Further, the use of similar application programs reduces the need to send large amounts of data in order for the programs to interoperate. Since the programs start out with the same "shared data", a program can simply "refer" to data, rather than transmit it. For example, in the *Packet* Chess program, instead of sending the entire image of the whole chess board with each move, we can just send the move information itself, since the program at the other end already 'knows' what the board and pieces look like, and the location of all pieces. This means you can use graphical images, without the need to transfer large amounts of data over a relatively slow connection.

PACKET CHESS

What is *Packet Chess* ?

Packet Chess is a program written in 1988 by Dewayne Hendricks (WA8DZP), and Rob Taylor (KA6NAN), to demonstrate the utility of a different approach to application software on Packet Radio. The program was designed to operate on the Apple Macintosh personal computer and to take advantage of the unique user interface and capabilities available on that system. Figure 1 below shows the initial screen which the player sees once the program is started. At this point, a selection is made as to either start the game or to initialize the session parameters. The player proceeds with their choice by "clicking" on the appropriate button with the mouse.

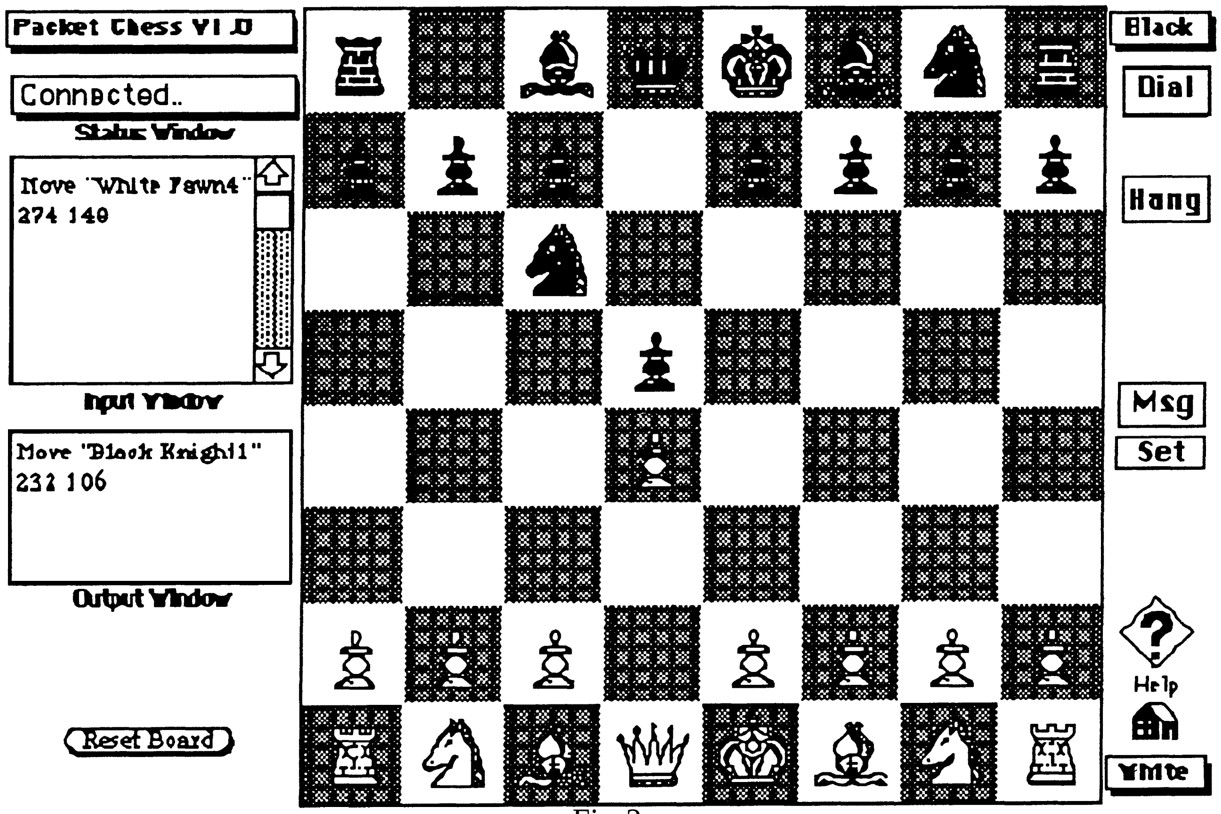

Packet Chess

Chess over Packet Radio
by
Robert Taylor, KA6NAN
Dewayne Hendricks, WA8DZP

Fig 1

What the program does.

Packet Chess permits two people to play chess over packet radio. Our design goal was to design the user interface so that the play of the game would be similar to what would occur if two players were actually facing each other over a chessboard in the same room. The program does NOT play chess, it simply allows two players to exchange moves easily. While playing, each player's screen looks like a chess board (Figure 2). To make a move, the player just "picks up a piece and moves it". This operation is performed by using the mouse to position the cursor over the piece and then clicking the mouse to select the piece and then using the mouse to move the piece to the desired location. You don't have to worry about commands to your computer, or your TNC, or even chess notation. Since all moves are displayed on both 'chess boards' at the same time, you simply play chess as normal!



Connected..

```
Move "White Pawn4"
274 140
```

```
Move "Black Knight1"
232 106
```

Fig 2

How to use *Packet Chess*

First, a player enters the opponent's callsign and other parameters for the session, and then presses the 'connect' button. Once connected, you just play chess by moving the chess pieces! When a chess piece is moved (using a 'mouse'), a text string is generated (such as "Move white queen to x y"). This text string is then sent automatically to the other program, which then makes the corresponding move on that screen. The players never have to type in any 'command', they just play chess! The players can send messages to each other at any time. To do this they just enter there message in the "Output" window on the chessboard and select the "Msg" button. This capability simulates the normal dialog the players would have if they were sitting across from each other. There is an option to use a modem instead of packet to allow the program to be used over telephone lines. There is also a button that causes the entire board to be reset to the starting configuration. There is no attempt to validate the legality of any moves. Our goal was not to build a chess playing program, but simply allow two players to have a chess game as thev normally would.

How **was it written ?**

The *Packet Chess* program was written in HyperTalk™. This language is used by the HyperCard™ program on the Macintosh™ computer. HyperTalk was selected because the language can easily handle graphical objects and the other components of the Macintosh user interface. Since HyperTalk is a high level language, the programming was fairly simple. A set of subroutines were written in assembly language to send/receive information from the TNC and setup the necessary options on the Macintosh serial port.

On-Line Help

No matter how 'easy' you think your application is, it would probably benefit from some level of on-line help. On-line help is available in the Packet Chess program.. Figure 3 shows the screen which the player sees when he selects the help button on the chessboard screen. The player can use the mouse to select the topic of interest by selecting the appropriate index tab at the bottom of the screen. The graphical nature of HyperCard allows for an interesting way of presenting help information since images of information being discussed can be included in the help text.

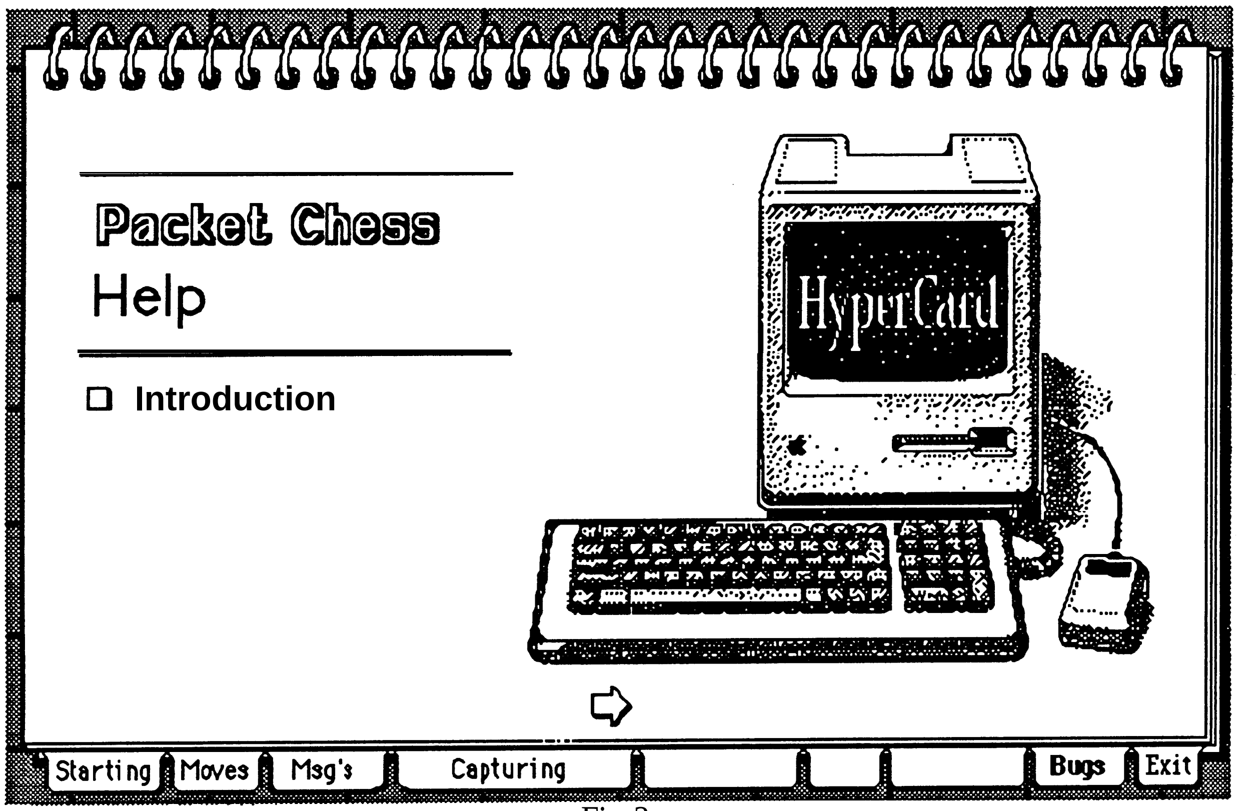
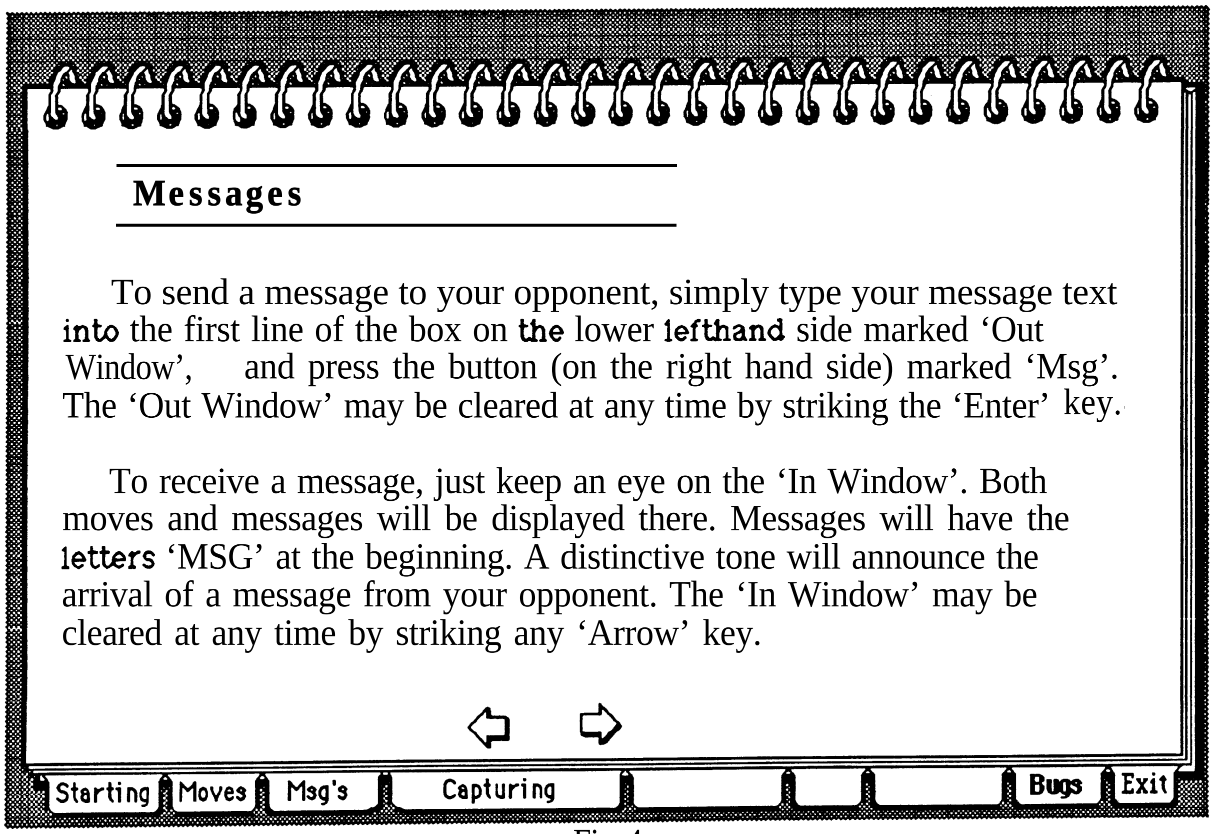


Fig 3



Messages

To send a message to your opponent, simply type your message text into the first line of the box on the lower lefthand side marked 'Out Window', and press the button (on the right hand side) marked 'Msg'. The 'Out Window' may be cleared at any time by striking the 'Enter' key.

To receive a message, just keep an eye on the 'In Window'. Both moves and messages will be displayed there. Messages will have the letters 'MSG' at the beginning. A distinctive tone will announce the arrival of a message from your opponent. The 'In Window' may be cleared at any time by striking any 'Arrow' key.

Fig 4

Figure 4 shows in detail one of the help screens. All help information is available at any time during a game.

Some ideas for other 'Packet Programs"

Many other forms of packet could benefit from our new approach to application software. For example: emergency communications between hospitals could be implemented as a "form" that gets filled out on the computer screen. The user would simply 'check-off' which hospitals to send the message to. On receipt at the other locations, the message would display/print out in the same format in which it was entered. Another example would be emergency co-ordination centers could utilize a large collection of maps. The locations of accidents and/or people and vehicles could be plainly indicated. Maps could be made to zoom in and out. This could be done verv efficiently because each site would have digitized forms of the maps stored locally. All that would have to be transmitted to the remote site would be a simple command like "display map 87E"! All the user would see at the remote site is a map of the accident area, and people/vehicles in their reported locations, along with any text messages.

Future Directions

Although the packet chess program was designed for 'regular' AX.25 packet, our current interests are in developing TCP/IP related applications. We want to generalize the ability of other applications to interoperate via TCP/IP. There are plans to incorporate Inter-Process Communication (IPC) ability into the Macintosh version of the KA9Q Internet Protocol Package. The would permit a single, stable version of the Macintosh version of the KA9Q package to work with a constantly growing set of applications.

Summary

Many packet services would benefit from application packet software. This is an opportunity for hams to truly "advance the state of the art", especially in the area of emergency communications. While much important work is being done in the area of network software, we cannot overlook the need for application level solutions. Improvements in the 'ease of use' and functionality are necessary and useful step in the evolution of packet radio.