This is the Robots help document.

First things first.   Though I *am* a NeXT Campus Consultant, NeXT was in no way, shape, or form responsible for this program.   I did it on my own, on my own time, on computers whose only relation to NeXT is that they built them.   Also, this program was written on my own, from scratch.   And similarities between this program and the old robots program are there because I wanted them to look alike, not because I used robot's author's code.   So, I, Scott Hess, retain the copyright to this program.   ie, you cannot make money off of this program, or in any way infringe upon users ability to get this program in its virgin state without stepping on my toes, which is illegal.   What this means is that you are not to mess with the executable in any way, shape, or form, or change any of the stuff I include, either.   This implies that people may run this program, and use this program as intended, free of charge, and that certainly is the implication that I want to get across.   You may also modify the .tiff files I enclose as demonstrations of character sets, though I would not recommend it.   After all, if you mess them up, they are gone!

That out of the way, down to business.

Why?

During finals this year, I was getting a bit frazzled.   I was also becoming concerned about the imminent rewrite of Stuart (my vt100 emulator on the NeXT) which was fast approaching.   So, to kill two birds with one stone, I started work on Robots.

Robots is a NextStep adaptation of my favorite unix game, robots.   In this game, robots chase after you, and you must run them into each other.   Your only defenses are a limited supply of antimatter, plus teleportation.   Other than that, you must rely on skill.

robots (the unix version) is surprisingly hard, especially when the number of robots on the screen becomes very large.   Since making the screen larger (ie, calling up a large terminal window) makes the game much easier, and also more intense, you find lots of people with large terminal windows playing robots.   This really drags a terminal emulator down, especially in the nether realms of 130x70 characters, and higher.   Terminal suffers under the load, Stuart nearly dies, and even xterm, which is the fastest emulator I have around slows down quite a bit. Besides, isn't it silly to run a terminal emulator to play a quasi-graphical game on a graphics workstation?

So, Robots was born.   Robots is a full graphical version of robots, with the ability to change the character sets, and have all sorts of fun with it.   It is fast, by virtue of the lesser amount of processing needed.   It is also a "real" NextStep application, in that it uses the defaults database, has an info panel, Preferences, etc.   The whole ball of wax, in other words.

How to use it.

What Robots doesn't have is a different user interface.   Or at least, not a substantially different interface for keyboard input.   You still run things through the keyboard (anyone want to send me a NeXT-friendly joystick? :-).   Which isn't bad, because then old users of robots will be right at home.   For the rest of you, Robots uses the vi keys, which are not really that obvious, but are really quite good once you get used to them.   Some of you may have run across them in other games, because they are quite popular (seeing as most unix users have used vi).   The movement keys are as follows:

```
yku
hwl
njm
```

The keys move your player one square in the given direction.   For example, y moves one square to the upper-left.   w "waits" - moves not at all.   These keys are really in a different order on the keyboard, but I trust you can find them.

For convenience of those who could care less what keys vi uses, I've also added a mapping so that the number pad keys also do stuff.   The mapping for the number pad, is:

```
789
456
123
```

Looks a lot simpler, right?   Easy to remember, right?   Well, I don't like them. But others might.   For really silly people, the arrow keys work as expected.   But there are problems, because there are only four arrows, and we really need nine.

So, now you're moving one space at a time with ease.   What next?   Wouldn't it be nice to be able to jump about a little faster?   Well, there is a way.   Using the ctrl key while pressing one of the direction keys causes that direction to continue until you are in danger of either getting killed, or running into something.   If all of the robots are dead, you stop also.   For the more daring, the shifted version causes the command to be repeated until either you die, the robots die, or you run into some immovable object, such as a wall.   Try shift-w sometime :-)   The shift and control keys also affect the numeric movement keys.   As a convenience to the numeric pad users, the command key is a synonym for the ctrl key, so that you can use either side of the keyboard.

So, what is all of this killing, anyway?   Well, you see, the robots come at you as fast as they can, by the most direct route.   These robots are not smart, as they even will run into each other in their enthusiasm to get to you.   This is what you want, though, because robots which run into each other turn into a scrap heap, and robots who run into a scrap heap are added to it.   All of the scrapping gives you points.   Which is good.

Some robots come at you a square at a time.   That is also how fast you move, so they aren't that bad.   Others come twice as fast, so watch out!   These are noticeable by the different shape or color.   They aren't that smart, either, as they sometimes run over their slower comrades and become a scrap heap.

Did I mention your offenses?   Of course, you've gotta have offenses, else you'd lose right away.   Your options are (given with the keypad equivalent in parens):

a (*) - antimatter, which kills all robots within one square of the player. Watch out for fast robots two blocks out, though!

t (+) - safe teleport. This teleports you to a position where you will not get chomped in the next round of robot movement.

r (-) - random teleport. This teleports you somewhere. You will not be teleported directly on top of a robot, but there is no guarantee that you won't end up with a robot adjacent to you!

Other keys which are helpful:

? (=) - help. Gives the "safe" moves from the spot the player is in. Sometimes, this routine may miss moves where the player would actually be protected by scrap heaps, but you can't have everything.

d (.) - dots. Toggles through three states of dots which indicate where squares are. This can be helpful to see if that fast robot is going to chomp you in the next round.

What isn't implemented?

I didn't implement the ever-popular ability to type in a number to specify how far you should move. I don't use that option, because I get killed more often than not with it. Various other, unimportant keys are not there, either, but they were redundant anyhow. I don't have the ability to work with a high score file, yet, but I plan on adding that eventually. I would like to be able to work with the standard unix robots high score files.

Defaults.

There are some defaults which are user-settable in the Robots program. These are easiest to set via the Preferences panel, but for those of you who _must_ know:

CharacterSet - allows the player to use different characters for their game. See the documentation at the end of this document. This default defaults to **Faces-16x16**.

HighScores - nothing. This is meant for when I add the ability to save high scores. It defaults to **YES**, if that helps.

FieldHeight - the number of lines on the playing field. Default, **30**.

FieldWidth - the number of columns on the playing field. Default, **30**.

RobotsWindow[**X**,**Y**] - the position of the lower-left corner of the robots window as measured from the lower-left corner of the screen, given in pixels. Defaults to **225**, **225**, respectively.

Name - your name for the non-existent high score files. This defaults to your username.

MoveableHeaps - a boolean flag to indicate whether the player may move scrap heaps by running into them. If there is something blocking the motion, the scrap heap will not move. This defaults to **YES**.

The boolean defaults may be given in a number of ways, as taken from the following set: t, true, y, yes, any of the previous in any capitalization, and non-zero numbers are taken to be YES. n, no, f, false, any of the previous in any capitalization, and 0 are taken to mean NO.

If you ever get your defaults so screwed up that nothing works, try doing the command:

```
dread -o Robots | dremove
```

from the unix command line. This will remove all of the defaults from the database, allowing you to start over with a new set of defaults which are at least sane. Otherwise, you can selectively use dremove to remove defaults which you think are messing things up.

New character sets.

At this point there are three character sets included in Robots. One is the Faces set, which contains cutesy faces whose eyes follow the player. This set is large, and isn't that great to play, but is cute! The second set is the Bullseye set. This set is smaller than the Faces set, and thus is easier to play. It also gives better indications of where the player is after a teleport. It also has neat dots! The ASCII character set is included for those of you (us) who cannot give up your old habits, and must think you are playing on a terminal. This set is even smaller than the Bullseye set, and thus gives higher scores at times. Since it is non-square, though, it can be misleading.

Now that I've went over these sets, you want to know how to make your own. I know you do. It is really simple. All you need is a .tiff file of the following form (FR=Fast Robot, SR=Slow Robot, ME=player, Rb=any robot):

FRStill SRStill MEStill DeadRb Dot DeadMe1 DeadMe2 blank
Fast robots in various positions
Slow robots in various positions
The Player in various positions.

If you did not understand that, just go look at the included character set files. These are the same ones used in Robots. They are relatively obvious, except for possibly the ASCII-7x11 one. Due to the small size of that set, Icon kept wanting to enlarge my windows, so I padded out the file so that it's big enough that Icon like it. But the padding means nothing to Robots, as long as it runs along the right and bottom edges.

Each of the elements in the tiff file must be the same size. The size is indicated in the filename by the final -WxH.tiff, where W is the width of each cell, and H is the height. There should be 8 cells horizontally in the file, and 4 vertically. The order of the robots is easily seen in the Faces-16x16.tiff file. They move from facing left, and then in a clockwise rotation. You need not supply 9 different faces, though - in the ASCII file, for instance, all of the nine are the same, so when you play you see no difference. Print them out, if you need to.

For some reason or another, I wanted to have two dead player pieces. Unfortunately, I never used the first one! So, the DeadMe2 piece is the only piece that is used. To see which piece that is, look at Faces-16x16.tiff.

Another gotcha is that the background of the pieces must be light gray, so that it matches the background of the playing field. This isn't that bad of a deal, just remember to do it! Why is it so limited? Because I like light gray better than white, so I did it that way. Pffhht! In the future I may make the background color changeable, but not for now . . .

Getting source.

I mentioned in the beginning of this document that Stuart was somehow involved with this program. It turns out that I am going to be using many of the basic classes used in this program in my rewrite of Stuart. In fact, they basic classes are going to be so similar that I do not want to let them out right now. Its not that I don't trust you, but when I was writing Stuart, having these already for me would have cut original development time in half, and the speed would have doubled. Serious. So I'm not yet sure what to do. Sorry. Maybe I can eventually release at least those classes which I feel comfortable with, and that the general NeXT public will find uses for.

All correspondence, complaints, bug reports, monetary contributions (especially the last :-)**) to:**

scott hess
scott@gacvax1.bitnet

or:

scott hess
PO box 832
Gustavus Adolphus College
St. Peter, Mn 56082