Contents

1 Eden **2**
1.1 How to start Eden from the operating system 2
1.2 Starting a life from Prolog 3
1.3 Controlling Eden 3
1.4 Stepping through a life 4
1.5 Experimenting with the control commands 5
1.6 Two more commands — turning the view window on and off 5
1.7 A shorter way to start new lives 6
1.8 What are the objects? 6
1.9 Logging out and in again 6
1.10 The basic actions and objects 7
1.11 The manual bug 8

How to run Eden

Jocelyn Paine

February 15, 1996

1 Eden

This handout shows you how to run Eden. There is quite a lot of material here, and the best way to use it is to read it through once without trying out the commands, to get an idea of what I'll ask you to do. Then read through again, this time trying each command as I mention it.

1.1 How to start Eden from the operating system

Because CTC change their computer system from year to year, I can't include here instructions on how to log in. You should already know this from my demonstration, but if you don't, please ask. Note also that you need to log in in different ways on different machines. In particular, the method you use at CTC will not work in your college computer room, or in the Department. (This is unfortunate, and is the result of a long tradition of computer manufacturers refusing to standardise terminals and other equipment.) So if you want to work elsewhere than CTC, please ask me to show you how.

Let's assume then that you have already logged in and given your identification. You will then be talking to the VAX's *operating system*, the master control program which allocates the computer's "attention" between different programs and terminals. Each computer manufacturer has its own name for its operating system — the VAX one is called VMS.

VMS signals its presence by displaying a dollar *prompt* at the left hand end of the line. When you see this prompt, you can type a command, which you must terminate by hitting the RETURN key.

VMS accepts a variety of commands, many of which are to do with creating, copying, printing, and deleting files. If you have used another operating system, MS-DOS for instance, you'll be familiar with the type of thing they do. However, we shall not use these commands much. VMS also accepts commands which tell it to run programs, and we shall use one of these, the `eden` command. So when you see a `$` prompt, type

```
eden
```

and then hit the RETURN key. What this does is to tell VMS to start running Prolog, and to load into it the Eden microworld-simulator.

You should make sure that you are typing in lower case — that is, that you don't have the CAPS LOCK key on. Although VMS doesn't mind whether you use upper or lower case, Prolog does. Most letters in Prolog commands have to be lower-case, apart from the occasional use of upper-case for "variables".

1.2 Starting a life from Prolog

Once you have given VMS the `eden` command, there will be a short wait, and you will then see a Prolog start-up message followed by a new prompt — not a dollar, but the symbols `?-`. This tells you that you are now talking to Prolog. Actually, you are talking to a version of Prolog into which the code to run Eden has been automatically loaded. In the rest of these notes, I may use either the name Prolog or the name Eden, without being too careful about which.

The main thing you do, having started Eden, is to tell it to run a particular bug "brain" in a particular world. Let's demonstrate this with a simple production-system bug. Type the command

```
eps( psbug2, psworld2 ).
```

and go on to the next section. Make sure that you include all the punctuation, including the final dot. You must also hit RETURN afterwards. If you forget the dot or the RETURN, then the system will not regard your input as complete, and will just sit there waiting.

1.3 Controlling Eden

Once you have done this, the screen will start displaying one of the micro-worlds, and eventually settle down into something like figure 1, below:

```
---!   7!- (1 Continue? y/n/s(low) <cycles>/f(ast) <cycles>/l(isten)/p(op)/refre
 Action: none          Facing: east     Position: (3,3)
 Inventory: nothing    Here: nothing    Time: 0


 ***********
 *         *
 *  B   S  *
 *         *
 *+     1  *
 ***********
```

Figure 1 An Eden display.

This shows the situation at the start of the bug's life. The line at the top is a reminder of some of the commands you can give Eden, to control the way it steps through the life. Do not try any of these until I say so, in section 1.5.

Below it are two lines showing various bits of status information: the bug's previous action; the direction it is facing; its position; what it is holding; the object, if any, it is standing next to; and the "time". Below that is the micro-world itself, the bug being indicated by a `B`.

The reason for having the status information is that the display is, because of the terminals we are forced to use, very crude. The bug may be in the same square

as another object, but there is no way to display that in the diagram. Nor can we show what it is holding, or which way it is facing.

To appreciate how the control commands are used, you need to know how Eden runs a life. Essentially, Eden consists of two parts: a world-manager and a bug "brain". When you start a new life, these run in a series of cycles, each cycle being as follows:

1 The world-manager generates from the current world, an image on the bug's "retina". In some worlds, it may also generate various other perceptions, such as a "smell" indicating roughly where the food is.

2 It then activates the "brain". This "thinks" about the perceptions, and eventually comes back with a "motor command".

3 The world-manager works out how this action affects the bug and the rest of the world. It updates the world accordingly, displaying the changes on the screen.

1.4 Stepping through a life

The line at the top of the screen is a reminder of the commands you can give to control how Eden steps through a bug's life. You have several possible replies:

1. Typing `y` will run the life for one cycle. You will then be invited to type another control command.

2. Typing `n` stops the life and returns to Prolog. You will then get another `?-` prompt.

3. Typing `s` followed by a space and a number runs the life for that number of cycles before asking for another control command. We call this "slow" mode, because the screen is updated on every cycle.

4. Typing `f` followed by a space and a number runs the world in fast mode for that number of cycles. With fast mode enabled, Eden does not update the screen between cycles. This allows you to run a large number of cycles much faster than in slow mode. However, it also means that you won't see what the bug is doing. This option is mainly useful when testing machine-learning programs, and you won't need it much in this course.

5. Typing `l` enables you to pass a sentence for Bug to "listen" to. You probably won't use this much either.

6. Typing `r` refreshes the screen. This is useful if, say, the operators have just broadcast a general message, and you want to remove it.

7. Typing `e` allows you to "edit" the world as the bug runs. This is useful for experimenting with "opportunism" — testing how well your bug can adapt to sudden changes. To do this, you need to know how to use the editor, and I'll go into this in a separate handout.

8. Typing `v` allows you to edit the view window. Again, you need to know the editor first. Editing the view window is useful if there's a lot of output, and you want to go back and look at it. It's also useful if you want to print the window.

In each case, you must hit RETURN after replying. Note that your keystrokes are not displayed as you respond to this question. One thing that catches many people out is if you accidentally type some extra characters before answering the question and hitting RETURN, Eden will not understand what you meant and will repeat the question. In such a case just be patient and retype.

1.5 Experimenting with the control commands

Let's now run through these. I assume that you haven't done anything since starting the life in section 1.2, so you should still have the initial display of figure 1 shown on your screen. The simplest thing you can do is to stop the life and return to Prolog. To do this, type `n` and then hit RETURN. **It is important that here, and other places where RETURN is needed, you do type it. If not, the system will just sit there waiting, and you will never get a reply.**

The display should then clear. Once again, you will see Prolog's `?-` prompt.

You can start another life by typing

```
eps( psbug2, psworld2 ).
```

again. Remember to type the dot and the RETURN. For practice, do that now, and then exit with `n` again.

You are now back in Prolog again. Start a third life as above. This time, hit `r` and then RETURN. `r` stands for "refresh" or "redraw". The screen will flicker briefly as it is redrawn. I receive quite a lot of Email, and when it arrives, VMS announces its arrival by displaying a message. This may well happen when you are running Eden: if so, use `r` to remove the message.

Having typed `r`, you are still at the start of a life, and Eden is waiting for another control command. This time, type `s`, then hit a space, then type `30`, and then hit RETURN. You will see the bug moving around in the upper part of the screen. At the same time, the lower part will display what its "brain" is doing, in the form of production-system rules. After about 20 moves, no more rules will apply, and the life will finish, returning to Prolog.

If you want, you can try that again. It is always possible to run a bug over and over again.

You have now tried the control commands `n`, `r` and `s`. The final one to try is `y`, which runs a bug in single-step mode. This is where Eden pauses after each cycle and asks you whether you want to continue. Start a new life, and then type `y` and RETURN. The bug will go through one cycle, and then pause. Type `y` and RETURN again. Keep doing this until the life exits.

1.6 Two more commands — turning the view window on and off

There are two more control commands which are not shown on the status line. These are + and −, which turn the view window on and off respectively.

When you start a life, the view window is turned on, and will display all the production system's internal workings. Though useful, this does slow down the display, so it's useful to know how to turn it off.

You should be back in Prolog, with the `?-` prompt. Start a new life by typing

```
eps( psbug2, psworld2 ).
```

Don't forget the dot and RETURN!

Then type − and RETURN. Eden will then invite you to type another control command. Type `s`, a space, `30`, and RETURN. This time, the bug will run through

a life as above, but without the view output. It will eventually exit to Prolog, as before.

You can turn the view window back on with `+`, as I now show. Start a new life. Type – and hit RETURN. Then type `s`, a space, `10`, and RETURN. This will run the bug through 10 cycles. Next, type + and RETURN. Then type `s`, a space, `20`, and RETURN. This will run the bug through the rest of its life, this time with the view window on.

Of course, you can also single-step, using `y`, with view either on or off.

1.7 A shorter way to start new lives

It's a bit tedious to have to type

```
eps( psbug2, psworld2 ).
```

every time you want to reincarnate the bug. Eden actually remembers which brain and world were last used; if you want to use the same ones again, you don't need to explicitly mention them. So if you are in Prolog, and you want to re-run a bug you've already run, you can use a short form of the command:

```
eden.
```

Try this now. You must still have a dot and a RETURN to end the command.

Note: do not confuse this command with the `eden` command that you typed to VMS. Although they have the same name, these two commands do two quite different things, and you are talking to two different programs when you give them. This is explained further in my handout on *Who am I talking to? — operating systems and the computer*. Note also that you cannot give the `eden.` command when you start the first life from Prolog, because Eden won't know which brain and world to use.

1.8 What are the objects?

One can create many different kinds of object in Eden, and you need not be limited to those in `psworld2` or in the *AI and PopBeast* example. But so that you know what is going on in `psworld2`, here's a table of the objects it contains:

`*` Boulder. Can't be picked up or moved through. I usually use these to delimit the world.

`+` Food. We use the symbol + because it adds something to your energy! Can be picked up by doing a `grab`. Once picked up, it can be eaten by doing a `use`. You can move over it.

`1` Coin (one-p piece). Can be picked up and moved over. Once picked up, if you are facing a slot machine, `use`ing the coin will cause it to disappear, and a piece of food to fall from the machine.

`S` Slot machine. Can't be picked up or moved through.

1.9 Logging out and in again

After that interlude, back to computer commands. The next thing to do is to try putting it all together, making sure that you can log in and start Eden even if I'm not around.

You should still be talking to Prolog, as shown by the `?-` prompt. First, exit from Prolog by hitting the CONTROL and Y keys at the same time. This will get you back

to VMS, which will display a `$` prompt. Incidentally, hitting CONTROL and Y will get you out of almost any program.

Next, type `logout` and RETURN. This tells VMS that you want to finish the session. You will see a long display of filenames, and will then be disconnected from the VAX, back talking to the CTC microcomputer network.

Now, try logging in again. Once logged in, type your identification. Then start up Eden. Then start a life and run through it. Then exit from Prolog and log out.

Continue trying this until you can do it without any help.

1.10 The basic actions and objects

This section is mainly for reference. It summarises the eight basic bug actions, and the properties of the objects used in the *AI and PopBeast* and *Production systems* handouts. You will need it before you try out the manual bug described in the following section, and before you start writing your own brains. Eden is not, by the way, limited to the objects listed below, and it is possible to create new types.

There are eight actions:

```
forward     back
right       left
grab        drop
use         wait
```

Forward and back move one square forward or back, unless the square being moved into is blocked, in which case nothing happens.

Right and left turn 90 degrees in the appropriate direction, but don't move forward or back.

Drop and grab pick up or put down an object. If the bug tries to grab an object but the object is "too heavy", nothing will happen; also nothing will happen if it is already carrying something. Dropping an object leaves it in the current square; if this is already occupied by another object (of whatever type) nothing happens, and the bug remains holding the object.

Waiting does nothing for a turn.

The effect of using an object depends on its type, so this is the next thing to look at, in the table below.

The table also indicates, in the Share column, what happens on moving into the same square as an object. A y in the Share column means that the bug can enter the same square as the object, n means it can't.

The Grab column indicates the effect of grabbing an object. y means that it can be picked up; n, that it can't.

The numbers refer to notes below.

Symbol	Name	Grab	Share	Use
`B`	Bug	-	-	-
`+`	food	y	y	(3)
`*`	boulder	n	n	n
`@`	rock	n	n	n
`#`	door	n	n	n
`Q`	quicksand	n	(1)	n
`T`	hammer	y	y	(4)
`k`	key	y	y	(5)
`:`	narrows	n	(2)	n
`1`	coin	y	y	(6)
`S`	slot machine	n	n	n

1 Moving forward or back off the quicksand is OK; any other action will drown you.

2 You can only pass into a narrows if you are not holding anything.

3 Using food eats it.

4 Using a hammer only does something if you are facing a rock. The rock disappears, the hammer remains in your inventory.

5 Using a key only does something if you are facing a door. The door and key both disappear.

6 Using a coin only does something if you are facing a slot machine. The coin disappears, and a piece of food is dropped into a randomly chosen empty square nearby.

In this table, the "narrows" corresponds to a narrow gap, that you can only move through if not carrying anything.

Note that an unsuccessful action does not crash the system, it merely has no effect.

1.11 The manual bug

It's sometimes useful to be able to try out actions manually before building them into your bug brain. You can do this with the "manual bug". This has an "empty" brain which just reads commands from the keyboard and carries out the appropriate action.

To start it up, get into Prolog, and type

```
manual( psworld2 ).
```

Note that you use the command `manual`, not `eps` or `eden`, and that you only give it the name of a world.

When you get the world display, give the control command `s`, space, and `200`, followed as usual by RETURN. You will then get a display inviting you to type one of the letters `f`, `b`, `l`, `r`, `g`, `d`, `w`, `u` or `q`. The first eight are the initial letters of the eight basic bug actions. The final one, `q`, causes the bug to quit the life, returning to Prolog.

Now you know what the actions do, you can try these out. For example, try picking up the coin and using it in the machine.

You might also like to try the puzzle set by the `defaultworld`. This is the world with which I introduced Eden in *AI and PopBeast*. Try

```
manual( defaultworld ).
```

and see whether you can get the bug to the food.

If you want to run the manual bug in the same world more than once, you need not type the full command again. As in section 1.7, you can just type

```
eden.
```

followed of course by RETURN.