Alan Creak
1 August 1996

HOW TO USE MAPWALKER : VERSION 1.

In this note, I present the first version of the documentation for Mapwalker. The software described is certainly less than complete, so comment, criticism, and correction of both software and document are welcome.

WHAT IS MAPWALKER ?

Mapwalker is a programme which is intended to simulate some aspects of a journey on foot through a city as it might be experienced by someone who is blind. It should be obvious that a perfect simulation is impossible, and it was never intended; instead, the programme monitors and records a walker's progress through a simulated city, as directed by a subject given no cues which a sighted person could obtain visually.

The map in the title defines the city. It can be composed to represent a real city, or − more practicably − part of a city, or it can be constructed as a very simple network to test hypotheses about the process of recall and use of route information without visual assistance, which was the original intention. The software is designed to help an experimenter evaluate the effectiveness of different teaching methods for remembering a set route to a given destination. For example, the experimenter might describe the route from start to finish once, and then leave the subject alone to traverse the route. Alternatively, the experimenter might ask the subject to repeat the route verbally to help in remembering it before attempting the journey. With this software it should be possible to carry out useful experiments despite the impracticality of testing the methods by physically walking through a real city, while preserving enough of the essential nature of the journey to give some insight into which teaching methods are more effective. In particular, it should be possible to conduct repeated experiments under controlled conditions, which is quite impossible in the uncontrolled environment of a real city.

THE HISTORY OF MAPWALKER.

The original suggestion came to me from Professor Steve LaGrow of Massey University in an electronic mail message of 15th July 1993. He wanted such a simulator as a tool with which to carry out experiments on the effectiveness of verbal preview on blind people's abilities to travel along specified routes in unfamiliar environments.

I thought it possible that students might be interested in addressing the problem as an assignment in my course at Auckland on Robotics and Real-time Control, and dressed it up (not very much; the original specification was very apppropriate for an assignment in real-time simulation) to suit the course. Two students[1] combined to tackle the assignment in 1993; they broke the ground, but didn't have time to get very far. A second pair, Chris Miller and Rem Kuipers, put in a lot of work in two assignments[2, 3] on the same topic in 1994, and constructed a working simulator. I gratefully acknowledge the work done by these students, much of which remains in the current version of Mapwalker, while some is embodied here and there in this documentation.

The simulator built by Miller and Kuipers conformed more or less to the specifications, but was written for MS-DOS (not necessarily a bad thing), presented a user interface which I found less than straightforward to use (a fairly bad thing), and was not very flexible in offering different combinations of parameters for the simulation (a thing which I wanted to change before exhibiting the programme to anyone else). I wanted to increase its flexibility so that it could simulate a range of conditions, including those originally specified by Steve LaGrow; now I've at last made a start on that, and also adapted the programme to run with Windows. The result shows evidence of a lot of adapting to Windows and rather little increase in flexibility, but it works after a fashion. The system structure is certainly better (though much compromise between MS-DOS and Windows remains, and should eventually be removed), and further adaptation should be comparatively straightforward.

THE PRINCIPLES.

Though Mapwalker can be used by itself (that's what I describe in the first part of these instructions), it is primarily intended as a system for experimenting on how well people can understand verbal instructions. Altogether, three people (not all real) are involved in a Mapwalker run :

- the experimenter, who designs, runs, and supervises the experiment, but who can be omitted if desired;
- the subject, the only really essential participant, who controls the progress of the walker; and
- the walker, who doesn't really exist.

The experimenter must provide a map, and can determine the parameters of the experiment. The city must be designed and constructed, and the nature of the subject's interaction with the model must be determined. To define the city, the experimenter must compose a map file, while the interaction details are determined by a definitions file which must also be set up according to requirements.

The subject controls the behaviour of the walker by entering instructions at the keyboard in response to reports of the current experiences of the walker issued by the simulator. Provided that a map file is available, a subject can use the programme; if no definitions file is specified, the programme will attempt to load a file of a standard name (**simcity.def**), while if that file is not available the programme will use a set of built-in parameter values which will usually work reasonably well.

The walker is a simulated being who travels around the simulated city on foot according to parameters determined by the experimenter, but following detailed instructions given by the subject.

INSTRUCTIONS FOR THE SUBJECT

HOW TO USE MAPWALKER.

The description which follows is composed of a straightforward set of instructions, in this typeface, *interleaved with comments and suggestion, in* this typeface. *Responses to the comments and suggestions (or, for that matter, to the straightforward instructions) are welcome.*

WHAT YOU ARE TRYING TO DO.

When you use Mapwalker, your aim is to find your way through a simulated city from a starting point to a destination, using only directions which you have been given at the start of the run and the very limited feedback from the city itself that you might get if you were blind and using a cane to sense your immediate surroundings. As you are not simulated, you can't walk through the city in person; instead, therefore, you do so by directing the movements of a simulated walker. The walker moves along the footpath one step at a time; he can tell when about to walk over a kerb or into a wall. You give instructions by pressing keys on the computer keyboard as instructed by menus or instructions which appear from time to time on the screen.

Your journey is timed, and, while speed is not regarded as of primary importance, you should try not to waste time. Two times are displayed on the screen throughout the run. They are the elapsed time, and the simulated time you have taken. The elapsed time runs at the same rate as the real clock, but can be suspended in some cases; the simulated time runs at least as fast as the real time, and sometimes faster, so it counts your thinking time honestly but can speed up the simulation for actions. Typically, for example, you can take two or three simulated steps every second, while each step might count for a more realistic second or more in simulated time.

You should also try to manage by yourself. It's possible to ask for hints (see below), but these correspond to entering into conversations with people in the street, so there is a significant simulated time penalty for each such request. Other penalties might be applied for other time-consuming activities.

THE SIMPLEST WALK.

There are several parameters which control the details of Mapwalker's behaviour, and which can be adjusted to give different sorts of behaviour, and therefore different sorts of walk. In this first description, all the frills are ignored; if you follow these instructions, you will be able to use Mapwalker at its simplest. Later, more details of the parameters and their use are given.

1 : Start the Mapwalker programme.

Find the file **mapwalk.exe**, and start it by any usual method. The main window appears, partially covered by a "dialogue box" entitled *Choose your style !*.

2 : Choose the stepwise definition style.

In the dialogue box, you are asked to make a choice between defining the session by following a *Step by step* guided path, or working out your own sequence of definitions. (You may also choose to *Cancel* the choice, but at present that does exactly the same as choosing *Do it yourself.*) All the choices offered in the stepwise sequence can also be found in menu items, but the step-by-step style leads you through a simple complete definition sequence, which I'll describe here. Except for the necessary choice of the map file, you can respond to each of the questions answered by pressing the *Enter* key, and a satisfactory session will be set up.

Are the parameters set up in this way satisfactory ? Should other choices be included in the stepwise routine ?

2 : Answer the question "Do you want a log window ?".

The log window displays the record kept by the programme as it runs. The record is always kept in a disc file called **logfile**, so that it can be analysed to determine what happened and to gather useful experimental data, but it is only displayed if you request it. The display begins at the time of the request, which can be made at any time from the *Set up* menu.

If you choose to display the log window, and you haven't changed the size of the main window, the main window is slightly reduced in size so that the top of the log window is visible behind it at the right side. The log window is entitled *Mapwalker log*. You can display it at any time by clicking on the window, and resize it in the usual way to suit your preferences. You must bring back the main window to the front (click again, if necessary) to carry on with the session.

3 : An information box introduces the choice of a map file.

When you click the OK button, you are presented with a standard file choice box. The choice is set up to show only files with the suffix **.map**; while this is the expected convention, you can use any map file name you want, and you can retype the assumed ***.map** with (perhaps) ***.*** if you wish. Choose the map file which you want.

> Perhaps this should be predefined – or at least predefinable – for an experiment. It could be included in the definitions file (see below). Perhaps there should even be a special experiment mode, in which everything is predefined.

If you find that the file you chose isn't the file you wanted, you can always choose another by selecting *Map File* from the *File* menu when the stepwise sequence is complete – or before, if you prefer, but that will interrupt the sequence.

4 : You are asked if you wish to choose a definitions file.

If you don't know what that means, you can safely answer *No*, whereupon Mapwalker will assume a set of safe definitions. The definitions determine various parameters of the simulation, such as the length of each simulated pace and the time taken for the pace.

If you do wish to choose a file, a sequence like that which you used to choose a map file begins. Follow it through in the same way.

> I've left this here because it seems likely that a definitions file will be common in experiments, but it seems to be intrusive, because it isn't soething one might ordinarily expect a subject to know about. There is a way out; if you don't choose a file, Mapwalker will look for one called **simcity.def** anyway, and accept those parameters if there is such a file, so an experimenter could set up the **simcity.def** file with parameters as required. Perhaps this step should be omitted.

5 : The walker is ready to start.

A final information panel announces that the setting-up sequence is complete, and a headline appears in the main window. When you click the *OK* button on this panel, the experimental run begins. Don't click until you're quite ready to start the clock !

6 : The journey begins.

The journey displays appear in the main window. The first display shows your initial position, and your destination. You proceed by pressing ordinary keyboard keys, as directed by instructions which appear from time to time. A message on the screen will tell you of any exceptional condition, such as the walker's arrival at a kerb or at an obstacle. You can also select menu items if you wish, but these are regarded as extraordinary and might incur time penalties.

> Should you be able to read a description of the map first ? That would require a description in the map file, but there's no special difficulty about that except that the description has to be

composed by someone. (Automatically composed descriptions might be possible, but seem not to be too easy to construct.)

The initial and final positions are chosen at random by Mapwalker. Clearly, for a controlled experiment something more systematic is desirable. What should it be ?

The common instructions are of two sorts, roughly corresponding to behaviour at a junction and behaviour on a simple road. At a junction, you can commonly choose one of several possibilities by pressing a numeric key, while on a road segment you control the walker's motion by pressing one of the arrow keys.

Is this too elaborate ? A more realistic simulation might simply give a choice of the four arrow keys at any time, unless some exceptional operation (such as a turn through an awkward angle) was possible. It's as it is because that's how the students did it, and it would take a little work to change, but I suspect that the change would be worthwhile in the interests of verisimilitude.

Crossing the road is important, and sometimes allowed, though not well implemented, in the current simulator. (Historical reasons again.)

A clock showing two times (the simulated time and the elapsed time) is also displayed.

Should the clock be displayed always ? – or be chosen each time ? – or never displayed in the simplest procedure ? A virtue of displaying the clock is that it's clear that something's still happening, but it isn't really necessary for the simulation.

The clock at the moment begins as soon as the previous OK button is pressed. Should it be deferred until the subject has had time to read and absorb the details of the journey ?

Follow instructions. Once you start moving the walker, continue to control the movement with the arrow keys until a further message appears. Notice that each arrow key operation corresponds to one step or turn; you have to keep on giving instructions if you want the walker to keep moving. (Just how many steps you require can be adjusted by changing the simulated step length, but that's outside the scope of the simple simulation.) There's a built-in step delay, so you can't travel very quickly just by holding down the arrow key – indeed, it's likely to be quicker to get used to pressing the button at just the right time for every step.

What sort of variety should be available for the step mechanism ? At present, the step size is set so that each road segement is many steps (around 100 for the simple maps) in length. This gives the subject plenty of time to forget the directions while making the steps one by one, and concentrating on not walking into the road when a junction appears. It is equally easy to set the step length large, so that it exceeds the road length; the walker then leaps long streets at a single bound, moving directly from junction to junction. The step time delay is also variable, and can be fiddled to suit requirements.

If not all is well – if you get lost or tired or fed up – there are menu items which might be helpful. You can choose *Hint* from the *Run* menu to give a (not very precise, but better than nothing) description of your current position, or *Give up* from the *Run* menu to end the journey. If really desperate, choose *Exit* from the *File* menu, which stops the whole run. There's more information about these choices in the ELABORATIONS section below.

There's a time penalty for a hint. The time itself isn't significant for the real experiment, but the request for a hint is noted in the log. Should penalties be counted separately from the simulated time ?

You should try to behave as though you're walking along a street – so don't just keep on pressing the keys without looking to see what's happening. You can easily keep on stepping when you come to a kerb, which is the equivalent of walking out into the stream of traffic.

Should that be penalised too ? At present, there's just a warning message, even though walking into the road is much worse than pausing to ask for help. It seems fair enough that a subject should be expected to pay attention to feedback from his "cane".

7 : The journey ends.

Eventually, all being well, you will reach the destination, when a congratulatory message is displayed on the screen. Your times are also displayed.

You are now invited to proceed with another journey, either with the same or a different map, or you can choose to stop the session.

Or an automatic experiment supervisor could set up the next experiment. Now that the basic stuff is working (after a fashion), that should be fairly straightforward.

ELABORATIONS.

As well as following the step-by-step route, you can use the menus at the top of the main window. Whenever you can get at the menus, you can use them; any menu item can be used at any time you can "pull down" the menus with a mouse click (that's most of the time, but some of the dialogue boxes take precedence), and when the name of the item is visible, and black rather than grey.

A complete list of the menu items follows, with a description of their functions.

This is a prototype : not all the items work, some might disappear, and others can be added as desired. Suggestions are welcome. Computer interface principles are that anything useful should be available from a menu somewhere.

The *File* menu.

Items in this menu are supposed to be connected with operations which involve copying material between the programme and the disc. In some cases, instructions here could equally well be placed in some other menu item; if so, they are duplicated.

The early Macintosh guidelines required that certain standard items should appear in the *File* menu. The aim of presenting a standard interface was laudable, but they seem to have forgotten that not all programmes are simple. Mapwalker uses two different sorts of file, and there are correspondingly two different sorts of operation you can do with a file; particularly in the case of the map file, these are more naturally grouped under a *Map* menu. But, given that you have both menus, when you want to load a map file, do you look at the *File* menu or the *Map* menu ? The items should therefore appear in both – which violates another early Macintosh canon (this one disappeared rather soon from their list of recommendations), to provide only one way to perform an action. In consequence of my efforts to follow both principles, the current version of Mapwalker follows neither consistently. I'll sort that out eventually.

New

(This item is permanently grey; it does nothing yet. It is intended to provide a means to make new maps and edit old ones.)

Most of the code exists; I've inherited it from the students. It's mainly a matter of time. Perhaps this choice should not be available during serious experiments.

Open a Definitions File ...

If you choose this menu item, you can open a definitions file. (The item *Open a Definitions File ...* in the *Set up* menu is identical.)

Open a Map File ...

If you choose this menu item, you can open a map file. (The item *Open a Map file ...* in the *Map* menu is identical.)

Save

(This item is permanently grey; it does nothing yet. It is intended for use to save a map file after it has been changed.)

Save as ...

(This item is permanently grey; it does nothing yet. It resembles the previous item, but lets you save the map under a different name.)

Print

(This item is permanently grey; it does nothing yet. It is intended to let you print a copy of the map.)

The last three items should probably be moved into, or at least duplicated in, the *Map* menu.

Exit

If you choose this item, the Mapwalker session is stopped. The log file is preserved on the disc.

The *Edit* menu.

Items in the *Edit* menu are conventionally for use when changing the file used by the programme. Mapwalker doesn't change either of its files as yet, but is eventually intended to do so. The menu is therefore of no use at all at the moment.

Undo

(This item is permanently grey; it does nothing yet. It is intended for use with map editing operations.)

The *Set up* menu.

Items in this menu are used to determine how the programme will run. They usually set switches or parameter values in the internal tables, which might or might not result in externally obvious consequences.

Parameters

(This item is permanently grey; it does nothing yet. It is intended as a way into a single display with which many requirements for the system configuration can be registered. Perhaps some of the other *Set up* items will find their way into this display when it eventually turns up.)

Open a Definitions File ...

If you choose this menu item, you can open a definitions file. (The item *Open a Definitions File ...* in the *File* menu is identical.)

Show the Log

If you choose this item, the log display will become visible; if the log display wasn't in use before, it will be opened. Even when open, you don't have to keep the display on the screen permanently − after you choose *Show the Log,* it turns into *Hide the Log,* which has the obvious meaning. (It doesn't stop the logging, just stops displaying the log text.) It's sometimes useful to be able to hide

the log if you like to have it available for inspection but find that its continual activity is distracting.

Hide the Log

Removes the log display from the screen; see *Show the Log*.

The *Map* menu.

For obvious reasons, the city map is a very important component of Mapwalker. The operations you can perform on the map are collected together in this menu.

Show the Map

This item is initially grey, and does nothing. It becomes black and active when you open a map file. Once it is active, choosing the item causes the current map to be displayed on the screen. As with the log display, if the main widow has not been resized, it is made a little smaller so that the map window is always visible. (In fact, unless you choose to resize the windows, all three are visible at once.) When the map is visible, the text of this menu item changes to *Hide the map*, which does so.

Being able to see the map is very useful for sighted people, but rather spoils the point if you're trying to test verbal preview. Of coure, no experimental subject would be tempted to cheat. But, just in case, should this item normally be unvailable ? This could easily be managed using the definitions file.

Hide the map

Removes the map display from the screen. See *Show the map*. After choosing this item, the text reverts to *Show the map*.

Open a map file ...

This item is identical with *Open a map file ...* in the *File* menu. It can be used to select a map file.

The *Run* menu.

The items in this menu have to do with the behaviour of the programme during a journey.

Start

Choose this item to begin the journey. It shows your initial position, and starts the clock. If you have not set up the system properly, you will be asked to complete the setting up before *Start* will let you proceed.

Give up

This item is initially grey, but becomes active while a journey is in progress. You can choose it to escape from a journey without ending the session.

Hint

This item is initially grey, and is made active during a journey. If you choose it, the programme will give some information about your current position, and repeat your destination. This is the equivalent of asking a question of someone you meet in the street (except that the answers should always be right), so you incur a penalty if you request a hint.

Different levels of hint are possible : one could separate a short statement of position ("You are on the left-hand side of a road"), a statement with street names, a full description of a junction, suggest a route to the destination, etc. Each of these corresponds to getting more

information from the environment. Would that be useful ? Most of it is in the code, but some of it is turned off.

The *Help* menu.

The items in the *Help* menu are, reasonably enough, supposed to offer help or information of various sorts. As yet, they must be regarded as failures.

Some day, perhaps !

(This item is permanently grey; it does nothing yet. When something else is available, the item will disappear; it's just my way of apologising.)

About Mapwalker ....

Choosing this item displays a panel showing a very abbreviated version of the history of Mapwalker. Click the close box (or press Alt+F4) to dispose of the panel.

The Log file.

The log file is primarily designed as a record of your journey which can be analysed later to see how long it took and how well you were able to remember the navigation details. It should record all significant events in the journey, including such details as the names of the files you use, certain errors and warnings, and requests for hints. The log is always produced, and is always called **logfile**; any existing logfile is destroyed at the beginning of a session.

The log window is provided for your convenience and interest; it isn't an essential part of the system, but sometimes – for error messages, for example – it's useful to be able to see it. Log messages are displayed in the log window from the time when you open the window to the time when you close it; hiding and showing the window don't interrupt the display. If you close the window by clicking the close box or pressing Alt+F4, the display stops, and starts again from scratch if you reopen the window from the menu.

INSTRUCTIONS FOR THE EXPERIMENTER.

The experimenter must determine the parameters of the experiment. The city must be designed and constructed, and the nature of the subject's interaction with the model must be determined. (Details remain to be defined : obvious items for definition are the set of operations available to the subject, and the extent of feedback provided by the model.)

> These instructions and descriptions are illustrative rather than authoritative. Some of them are from my original specification, but might not be implemented as I've described (or even at all), others are guesses from inspecting the rather tortuous code and observing the system's behaviour. It will get better if there's any point in making it better.

THE CITY.

The city is composed of street segments and junctions.

- Each street segment runs from one junction to another; that defines the direction of the segment. Every segment is a member of exactly one street.

- A street segment has a length, a width, and a shape (which for the moment at least is always linear). The length is taken to be the length of its centre line.

- A street segment may, or may not, have footpaths on each side. (For the moment, all street segments have footpaths on both sides.)

- A junction may be ordinary or special. An ordinary junction is the meeting point of at least three street segments. A special junction may be a point at the edge of the map at which a street escapes from the city, or the end of a cul-de-sac.

  The coordinates (in effect, latitude and longitude) of the centre points of all junctions are defined. The centre lines of the street segments which meet at a junction are deemed to stop at the centre point of the junction. The geographical positions of the street segments are inferred from their junctions and other characteristics.

  While the street segments continue in principle to the centre of the junction, it is necessary to identify the effective junction so that the progress of the walker may be more realistically represented. For this purpose, the junction is deemed to be the polygon which connects the points of intersection of the edges of the roads which meet at the junction.

- A street is a list of one or more street segments. Each street has a name, a first segment, and a last segment. A segment is identified by its street name and its serial number (starting at 1) in the street. The directions of the segments of a street are consistent in the obvious way.

THE MAP FILE.

The city is defined in a map file. It contains information about each street segment in the city. Here is an example, taken from the file **grid.map** :

```
Street name:    111 Ave
Street width:   8
Footpath width: 3
Segment length: 124
Node 1 pos:     157 276
Node 2 pos:     157 152

Street name:    111 Ave
Street width:   8
Footpath width: 3
Segment length: 93
Node 1 pos:     157  59
Node 2 pos:     157 152
```

```
Street name:    AAA St
Street width:   8
Footpath width: 3
Segment length: 123
Node 1 pos:     401 152
Node 2 pos:     278 152
```

The footpath width is not used in the software as yet, and perhaps adds little of use to the simulation. The street width is used to count paces across the road when necessary. The units are arbitrary, but must be consistent with those used in the definitions file.

The format is self-explanatory, and must be adhered to rigidly. (The parser isn't clever.) The whole city is described in terms of street segments; there is no separate definition of the junctions, so all junction properties must be inferred from the street segment list.

> This is not very satisfactory, if only because it's impossible to name a junction. A much better format would be a list of nodes with positions and other attributes (names, for example, if required), with a list of streets as node pairs. If the project continues, I'll want to move it in that direction.

At present, if you want a new city you have to construct the map file by hand. A graphical editor of some sort was constructed, but isn't in action at the moment.

THE DEFINITIONS FILE.

The definitions file contains values for some parameters. This is **bigstrid.def** :

```
150
4
36
540
simcity
1
```

The parameters are, in order :

`stepSize`	:	The length of each step.
`stepDelay`	:	The simulator delay for a step, in milliseconds.
`StepTime`	:	The "real" time assumed to be required for a step, in milliseconds.
`hintPenalty`	:	A time penalty for requesting a hint, in milliseconds.
`setupPass`	:	A password, required if you want to change the parameters (not used now).
`soundOn`	:	A switch to turn on the sound effects (not used now).

Notes :

- The uncharacteristically long stride (150 units) is set longer than any street segment to give the efffect of moving from junction to junction without any intervening steps. (Note the file name.) A more conventional setting would be 1.

- The password is ineffective, and I don't think it's worth having. The intention is to prevent people from "cheating" changing the parameters of the system, but owes more to the students' preference for playing it as a game than to experimental design.

- The students attempted to provide sound effects, including crossing signals and traffic noise. They added a certain amount of realism, and acted as a distracting influence making it harder to remember the instructions. They were quite effective. I haven't put any work into them, but I've left them there in case they might be useful.

REFERENCES.

1 : Chat H.-W., Wan W.-H. : *Walk-along-the-road simulator* (Auckland University Computer Science Department, 07.473 Assignment 2 Report, 1993).

2 : C. Miller, R. Kuipers : *Simcity* (Auckland University Computer Science Department, 07.473 Assignment 1 Report, 1994).

2 : C. Miller, R. Kuipers : *Simcity − the road walking simulator* (Auckland University Computer Science Department, 07.473 Assignment 2 Report, 1994).