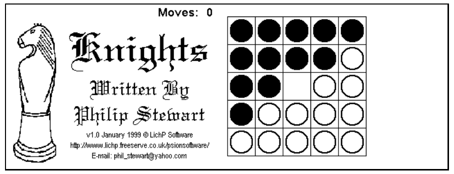
A Knightly Pursuit

A review of Philip Stewart's Knights puzzle, by Damian Walker


Moves: 0
Knights
Written By
Philip Stewart
v1.0 January 1999 © LichP Software
http://www.lichp.freeserve.co.uk/psionsoftware/
E-mail: phil_stewart@yahoo.com

Knights is a traditional puzzle game, the type at which handheld computers seem to excel. This solitaire-type puzzle is quite popular as a desktop toy, with its simple rules and pleasing appearance.

The puzzle consists of a 5×5 grid of playing spaces, and 24 pieces, twelve of each colour. These start in a particular configuration, with the white pieces clustered near one side and the black near the other. The central space starts empty. The object of the game is to swap the positions of these pieces. The pieces move, one at a time, using the knight's move from chess. Unlike solitaire, which is always solved in the same number of jumps, knights does not feature capture and therefore the pieces can jump back and forth as many times as they please. There is therefore an extra element of play, in trying to use the least number of moves possible.

The graphics are clear and serviceable, if not exciting. The screen is decorated with a hand-drawn chess knight, and some of the text is in an old-fashioned black letter font. But the board itself is a plain black and white grid of squares, and the pieces are plain flat circles. While there are clarity issues with the Series 5 screen, it would have been possible to be a bit more ambitious with the graphics. Nevertheless, they are very clear and do the job.

The user interface is very straightforward. It would be difficult not to be: the game is very simple, and there are no sensible variations, so there are few features to implement. The layout of the board and the rules of the puzzle are implemented exactly as in the traditional variant of the puzzle.

Knights is quite entertaining. It will appeal to all those who like the traditional peg solitaire game, or my own Pebbles on the move, with which it is perhaps more similar. Like Solitaire and unlike Pebbles, the solution is reasonably complex enough that it isn't easy to memorise after a single play. However, I think replay value would have gained benefit from an option to start the puzzle from a random position.

The quality of the program is very reasonable. It is let down by its lack of compatibility with anything other than a Series 5-sized device. While Geofox and netBook owners can play the game in letterbox mode, owners of the Revo and Osaris cannot play the game at all. The Ten Dan series of games, mentioned in the Osaris article in EPOC Entertainer issue 4, show how simple flat graphics of the type used in this game can easily be scaled to take advantage of various screen sizes. My other minor gripes with the quality are that the game doesn't allow itself to be closed from the system screen, and that the amount of memory it uses when running—368K, not including disk space—is a little excessive for the simplicity of the game. But other aspects of the quality of the game, such as speed and reliability, are absolutely perfect.

This game will appeal to owners of the Series 5 and 5mx, who are fans of traditional types of puzzle and board game. If you like the various Solitaire puzzles, or my own Pebbles on the move, then Knights will certainly appeal to you, and is certainly worth a try.

By	Philip Stewart
URL	psion.snigfarp.karoo.net
Licence	Shareware
Compatibility	Series 5/5mx
Rating	☆☆☆

EPOC ENTERTAINER
Issue 9: August 2008
Edited by Damian Walker
This month... SimCity

Welcome to another issue of *EPOC Entertainer!* This month there are two more reviews, *SimCity Classic* and *Knights*, both absorbing games. Files relating to the current issue are now available at the *EPOC Entertainer* web site.

Aspriring programmers will be pleased to see the continued *Taking Control* tutorial, all about scanning the keyboard.

Any comments, suggestions or contributions are welcome, as ever, at the usual address.

entertainer@snigfarp.karoo.co.uk

Urban Legend

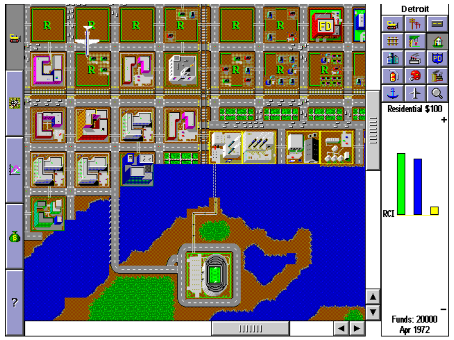
A review of Atelier's SimCity Classic, by Damian Walker.

Most computer games players are familiar with the game of SimCity. It first appeared at the end of the 1980s, on desktop machines like the Amiga and the PC. I first played it on a 4.77 MHz IBM PC, and it ran at a surprisingly smooth rate. When the Psion Series 5 appeared, the now defunct company Atelier ported the game to EPOC machines as a commercial offering, SimCity Classic. The game is still commercial, and therefore not available for download, but at least one Psion supplier still lists it as being available for purchase. I got my copy from eBay.

Unlike many EPOC versions of desktop games, this one is not cut down at all. That is as it should be: if the game ran on a machine with a 4.77 MHz processor and 512K of RAM, it should run without simplification on an 18 MHz machine with 8 MB. All the features of the original SimCity are present, and game play is absolutely identical.

The idea of the game is to build a city from scratch, and run it successfully using all the tools available to the town planner. You lay out zones for residential, commercial and industrial buildings, building roads, rails and power infrastructure, while the sims, your city's inhabitants, build their houses, shops and factories on the land you've set aside for them. As your city builds up, you have to deal with problems like heavy traffic, pollution and crime, and the sims occasionally demand facilities like a stadium or a sea port. They also rate your performance from one year to the next, and tell you what problems irk them the most. The game is open-ended, so there is no victory; the object of the game is the building and continued running of a successful city.

Presentation of the EPOC game is very good. You have a choice of grey scale or pure black and white graphics on monochrome machines, and colour on the Series 7 and



netBook. I find the pure black and white graphics best on a Series 5, as not only are they clearer than the grey scale ones, but they appear to have more detail. Sound is limited, as in the desktop versions. One thing lacking is the helicopter, which used to announce “heavy traffic reported,” but given that heavy traffic is one of the most frequent challenges of the game, this is probably a blessing. The general user interface is very good, and uses the familiar EPOC menus, buttons and dialogue boxes. This is essential, as the game is very detailed, and there are a lot of reports, maps and graphs that you need to help you in the running of the city.

The game is very addictive, especially as it encourages the more creative players among us. The city becomes your baby, and the urge to stick with it, tinker with it and watch it grow is extraordinary. If one city fails or becomes stagnant, the desire to have another try is sometimes overpowering. For those who like a real challenge, there are several historical and fictional scenarios, in which you take an existing city and try and solve its problems. One example is San Francisco after the earthquake of 1906.

The game is compatible across the entire EPOC range. It was originally designed for the Series 5 Classic, but doesn't suffer from being run on faster machines. The game automatically scales itself for any size of screen from the Osaris to the netBook, though some screen sizes suffer from glitches, like the scroll bars on the Osaris which are not quite at the edge of the playing area. There is also a colour patch available, so that the game can be played in full colour on machines that support this. The game has an adjustable speed option, which appears to work well on all machines.

There is full support for the EPOC operating system. Each city you create has its own saved game file, which appears as a document icon on the System Screen. You can create new cities or load existing ones from the System Screen, just as you would a Word document, and the program will close down from the System Screen if requested.

I have spotted a couple of minor problems with reliability, unfortunately. These is the cosmetic bug on the Osaris, as I mentioned above. There is also a more annoying bug, which appears only on my Series 7. When trying to build a road (or railway or power line) by dragging the stylus across the screen, the play area will jump about, resulting in some road sections being built at random in inappropriate places. The solution is to plot out the road by tapping on each individual spot where you want to build a road section. The bug doesn't seem to affect the 5mx.

These minor niggles aren't enough to spoil the game, by any means. As one might expect with a commercial franchise of one of the world's most successful computer games, SimCity Classic is a real gem, and has used up hours of time and battery life, especially on weekends when it has prevented me from getting up until well after my accustomed hour. It may take some finding, but I think that the effort will be well rewarded.

By	Atelier
URL	www.ebay.com
Licence	Commercial
Compatibility	Osaris, Revo, S5/5mx, Geofox, S7/nB
Rating	☆☆☆☆☆

Taking Control

The second article in the tutorial series on keyboard control, by Damian Walker

In the last issue we prepared the *Bouncer* program so that we can add keyboard input. The *movex%* and *movey%* variables that formerly controlled the ball's movement are now left at 0, so the ball doesn't move at all. Now we can add code to the program that will set these variables according to keyboard input.

In the *Animating OPL* tutorial the *KEY* function was introduced to scan the keyboard for the Escape key. *KEY* returns the ASCII code of any key that is pressed, or 0 if no key is pressed. There is an associated function *GET*, which will wait indefinitely for a key press. But that wouldn't be appropriate for this tutorial, because we have a 60 second timer to watch, too. In fact, using GET in a program prevents anything else happening while the program waits for a key (except the constant animation of the sprite), so GET is inappropriate for most games. For now, we'll stick with *KEY*.

As well as returning ASCII codes, *KEY* is capable of returning Psion's extended codes for such things as the *Menu* key and the cursor keys. In our bouncing ball example we'll use the cursor keys to move the ball around the screen in a predictable way. Firstly add the following line to the start of the *MoveBall* procedure:

```
LOCAL keypress%
```

Then add the following lines after *start&=DTNow&:*

```
movex%=0
movey%=0
keypress%=KEY
IF keypress%=256
  movey%=-4
ELSEIF keypress%=257
  movey%=4
ELSEIF keypress%=258
  movex%=4
ELSEIF keypress%=259
  movex%=-4
ENDIF
```

Upon translating and running the program, you can now move the ball around the screen for 60 seconds, using the cursor keys. How does this work?

The first two lines of our inserted code still the movement of the ball each time they're run. Without these lines, the ball would continue to travel in its chosen direction even after you let go of the key—try omitting them to see. The effect of continuous travel might be useful in some types of game, but for this demonstration we want the ball to come to a standstill, so make sure you reinstate the two lines if you've tried the experiment of removing them.

The next line scans the keyboard using the *KEY* function, and stores its code in *keypress%*. The checks on the remaining lines look at this value and set the directions of movement accordingly: 256 to 259 are the values for up, down, right and left respectively.

There are problems with this approach. The main problem when using *KEY* is that only keyboard events can be scanned. Perhaps all you want in your game are keyboard events, but a well-behaved application should at least scan for system events, such as a request from the system screen to close the program. You might want your program to pause itself when it's moved to the background. And if you want to scan for pen events as well as keyboard usage, then *KEY* is absolutely useless.

In the next issue we'll be reversing the changes made this month, and adding alternative code that can scan for system and pen events as well as looking at the keyboard.