The Right Connections

Five Connect 4 Games go head to head in this new series, by Damian Walker

An old game called The Captain's Mistress, said to be named from the amount of time Captain Cook spent playing the game in his cabin, is better known by its modern commercial name Connect 4. Players drop balls or counters of their own colour into a vertical grid, in turn, each player aiming to be the first to form a line of four.

The EPOC32 platform is very well served with traditional board games, and Connect 4 is one of the best represented. There are five of these games to my knowledge: 4Play, Connect 4, Four in a Line, Power4 and Puissance4. In this series of articles I will be taking a look at them all, in an attempt to rank them in order of preference.

4Play

The first game in our line-up is 4-Play, a freeware release by Palmanac. 4Play alone is purely a Series 7/netBook game, so users of monochrome machines will have no interest in it. This game is one of the simplest on offer, and the rules are completely faithful to the original board game. The board has the standard seven slots each accommodating six pieces, and in style is obviously influenced by the commercial Connect 4 game.

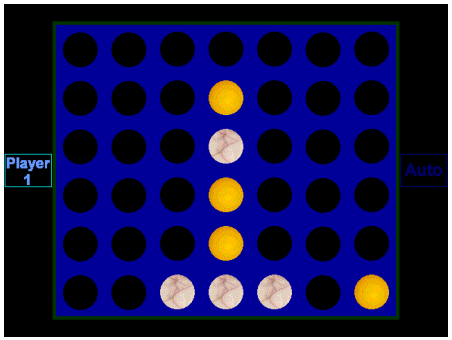
The graphics are attractive in a spartan kind of way. The pieces themselves do not resemble the plastic of the originals, but instead use marble and other textures. Sound is reasonably pleasant and appropriate, with a sliding noise and a click as each piece drops into place.

Despite its simplicity 4Play has one feature not universal in all of the games in this round up: the ability to decide who goes first. Row games tend to feel very different depending on whether you're moving first or not, and the option to let the computer go first can increase the variety of game play.

Unfortunately the game is marred by a very significant bug. If you switch to another application, and then return to 4Play, all the pieces disappear from the board! The pieces still play their part in the game, forming rows and stopping other pieces falling to the bottom of the grid, but they are completely invisible. This may not matter if you're always guaranteed an uninterrupted game, but it put an end to my ideas of a tournament between 4Play and Four In A Line.

This might be a good choice for casual play for Series 7 and Netbook owners, but for me the display bug is a significant black mark against this game.

In the next issue I'll take a look at Ben Vaughan's Connect 4.



EPOC ENTERTAINER
Issue 3: February 2008
Edited by Damian Walker
This month... Mille Bornes

As I promised in last month's issue of *EPOC Entertainer,* I have continued to play with the presentation. With hindsight, last month's issue looked a little bit dull to me, so hopefully the changes will make this month's issue more visually appealing. Let me know if you like the result.

As for content: the popular Animating OPL series continues. I have included a review of one of my favourite games, Mille Bornes by Fred Botton. I hope the fact it has four stars, not five, is some evidence of my objectivity in reviewing it. And finally, in this issue starts the first series of head-to-head game comparisons, this one about Connect 4 games.

I hope you enjoy the issue. As always, feedback is welcome, so let me know what you think of this issue by emailing me at the address below.

entertainer@snigfarp.karoo.co.uk

Animating OPL

In the latest part of this programming tutorial, Damian Walker introduces the concepts of sprites and masks.

In drawing the floor, we used a simple technique, of copying bitmaps. This is fine for static graphics like the floor, but it can be quite cumbersome for graphics which will be moving around. For that purpose, OPL provides *sprites*. Sprites aren't part of the OPL language itself, but are provided as an OPL Extension, or OPX. An OPX is a collection of procedures, usually written in another language like C++. The Bmp OPX contains the procedures for sprite handling, and to use them you need to at the following line to the top of your OPL program:

```
INCLUDE "Bmp.oxh"
```

Now that technicality is out of the way, we can turn our attention to sprites themselves. What exactly are they? Sprites in OPL are a development of the bitmap technique. Unlike a bitmap, a sprite exists only at one position on the screen. It cannot be copied like a bitmap, but it can instead be moved. In addition to this, a sprite can be animated, in that it can vary its appearance as time passes. In our demonstration we'll be using this capability to make the ball appear to bounce up and down.

Another feature of sprites is transparency. All bitmaps are rectangular, but we want our ball to be round. To see

the implications of this, add the following procedure temporarily to our program, just before the Bouncer procedure:

```
PROC Temp:
  LOCAL ball%
  DrawFloor:
  ball%=gLOADBIT
        ↗("\Bouncer\Ball1.mbm")
  gUSE 1
  gAT gWIDTH/2-8,gHEIGHT/2-8
  gCOPY ball%,0,0,16,16,3
  gCLOSE ball%
  DO UNTIL GET=27
ENDP
```

When you translate and run this, the ugliness of the inappropriate white box is plain to see. gCOPY does allow transparency in a rudimentary way: just try changing the final 3 parameter to 0 and see what happens when you translate and run again. The white box is gone, sure enough, but the whole ball is now transparent. We want something a bit more sophisticated than this, but what?

Proper transparency is achieved using the technique of masking. A mask is a silhouette of a shape, drawn in black on a white background. Sprites use these automatically, in such a way that everything that's white on the mask is treated as transparent, and everything that's black on the mask is drawn. This explains the purpose of the Ball and Mask pairs you drew earlier in Sketch. You can see how this works in a simplistic fashion by changing the Temp procedure to read as follows:

```
PROC Temp:
  LOCAL mask%,ball%
```

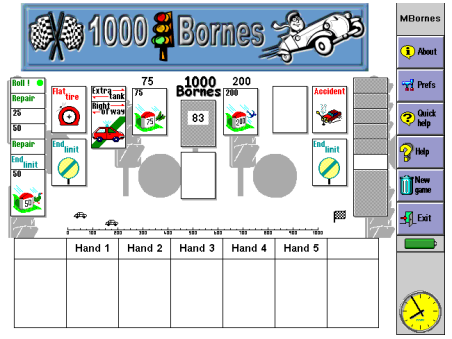
The diagram on the left shows the ball drawn without a mask. The diagram on the right is drawn with a mask.

```
  DrawFloor:
  mask%=gLOADBIT
        ↗("\Bouncer\Mask1.mbm")
  ball%=gLOADBIT
        ↗("\Bouncer\Ball1.mbm")
  gUSE 1
  gAT gWIDTH/2-8,gHEIGHT/2-8
  gCOPY mask%,0,0,16,16,1
  gCOPY ball%,0,0,16,16,0
  gCLOSE mask%
  gCLOSE ball%
  DO UNTIL GET=27
ENDP
```

If you want to see properly how this works, stick a GET statement between the two gCOPY statements. The first gCOPY uses the 1 parameter at the end to *erase* pixels, using the silhouette mask. This leaves a circular white hole in the tiled floor. The second gCOPY draws the ball, with *all* white pixels treated as transparent. The effect of these two operations means that any pixel that is white on the main bitmap *and* on the mask is left alone when the image is copied. Now that we've seen transparency at work in a static image, you can delete the Temp procedure, and next month we'll return to sprites.

Hit the Road

Damian Walker's review of Fred Botton's card game, Mille Bornes.



In the 1960s a card game called *Mille Bornes* became popular in France. It used a customised deck, featuring a motoring theme, and the object of the game was to travel 1000 kilometres before your opponent. The French equivalent of milestones are called bornes, hence the name.

There are a number of types of card in the deck. Distance cards have a number on them, from 25 to 200, and to play one of these simply adds to the number of kilometres you have travelled. Hazards such as *Flat Tyre* stop your opponent. Remedies, like *Spare Tyre*, get you going after suffering a hazard. Safeties, such as *Puncture Proof*, prevent your opponent from playing a particular hazard on you.

Certain actions in the game, such as finishing the 1000 mile trip, playing safeties at the right time, or finishing without playing a 200, earn more points to your score. Fred Botton has brought the two-player version of this game to the EPOC32 platform.

The first thing I like about this game is that it works on nearly all EPOC machines. This is a collaborative effort: different authors have released different versions for the Revo, Series 5 and Series 7 sized screens. While this is not as clever a solution as scaling a single version to suit all screens, it does the job. Only the Osaris is left out; I have no Geofox to test on but assume that it will run the Series 5 version without a problem.

The rules of the game have been slightly altered from the original card game. Whether this is at the author's whim, or whether variations of the card game were sold, I do not know. For instance, the original two-player game usually stops after 700 miles, and the scores for playing safeties at the right times differ between the EPOC and original versions.

Despite these differences the game is very addictive. A match is a group of five games, and I find it hard to tear myself away in the middle of a match. I find myself going back to the game time and again.

Graphics are clear and attractive, though no more exciting than any card game. The Series 7 version has the addition of a score board and title plaque, but a score card is brought up at the end of each round on the smaller machines. The cards are quite clear, using colour on the Series 7 and shading on the other machines to distinguish between one type of card and another.

The interface is clear and easy to use, though I don't like being presented with a dialog box on start-up; I'd prefer to select Start Game from a menu. But overall, I find this game very enjoyable and recommend it to anyone to try.

By	Fred Botton
URL	www.pscience5.net/LFGames.htm
Licence	Freeware
Compatibility	Revo, Series 5/5mx, Series 7
Rating	☆☆☆☆