known as MBM files), not the Sketch file itself.

First we'll draw a single tile for the floor. Our program will copy this across the whole screen, so only one tile is necessary. Set the sketch size to 16×16 pixels using the Resize Sketch option on the Tools menu, and draw the floor tile graphic as illustrated. Feel free to embellish it if you like, or to add colour if you have a Series 7, but don't make it too intricate as it's going to be copied a few hundred times across the screen. We're not trying to excite our viewers, but we don't want to give them a headache either! Once the tile is drawn, use the Export... option from the File menu (under More), to export the picture as Floor.mbm.

The ball is a bit more complicated. We want it to bounce, and to do this, we'll draw three separate frames of animation. The ball will grow bigger and smaller as it bounces, as if we were watching it from above. At its highest and biggest, the ball will be 16×16 pixels, which is handy as our sketch is currently that size after drawing the floor tile. So use the Delete all option from the Edit menu to get rid of the floor tile, and draw the ball outline as shown. Export the picture as Ball1.mbm. Now we need to draw a silhouette of the ball, the use of which I'll explain afterwards. Draw a filled circle in place of the hollow one and export that as Mask1.mbm.

The next frame of animation will be slightly smaller. Reduce the sketch size to 14×14, and clear the sketch using Delete all. Draw another ball as before, exporting it as Ball2.mbm, and a silhouette exported as Mask2.mbm. Finally, create a smaller frame at size 12×12, exporting as Ball3.mbm and Mask3.mbm. With all these frames you can make the ball as pretty as you like, but the silhouette should be pure black and white.

In the next issue, we'll start on the programming side of things, and create a display using the bitmaps we've drawn.

Illustration 1: an example of the floor tile graphic.

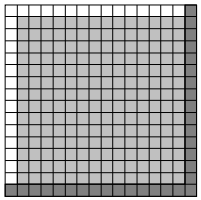


Illustration 2: the largest ball frame of size 16×16 pixels.

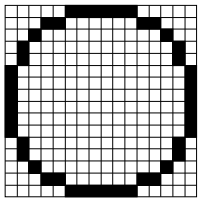
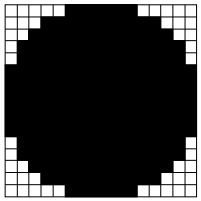


Illustration 3: the silhouette mask of the largest ball.



Issue 1: December 2007

Edited by Damian Walker

EPOC entertainer

Welcome!

Welcome to the first issue of Epoc Entertainer, the new monthly online publication EPOC32 owners who want to play! We'll be covering every EPOC32 machine from the Osaris, the Revo and the Series 5 through to the Series 7 and netBook. You might ask the question: why? Well, maybe we'll figure that one out as we go along.

While the EPOC32 platform's glory days may be well in the past, there are still a lot of people who use these machines for business and pleasure. They're still in demand in places like eBay, with the 5mx and netBook selling for more than pocket money prices. Past publications have often focussed on the businesslike qualities of these machines, but now is the time to let these little computers kick back and relax.

Articles planned include head-to-head comparisons of similar games, spotlights on the games of some prolific authors and publishers, as well as reviews of individual games that we'll single out for attention. There will also be games programming tutorials—remember, with their built-in OPL programming language, most EPOC machines are still better than modern PDAs for getting into a bit of programming.

As this is a new publication, you may well see the style and format change over the next few months, as we settle in. Your suggestions are always welcome!

In this issue we'll start off with a review of the excellent and popular Chain Reaction, and start the first installment of a tutorial on animated graphics in OPL. We hope you enjoy this early Christmas present!

entertainer@snigfarp.karoo.co.uk

Chain Reaction

A Review by Damian Walker

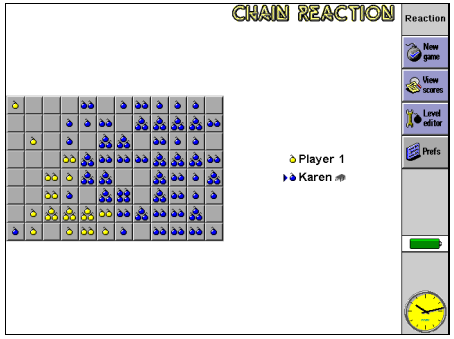
Chain Reaction is a game by Adam Dawes, from Neuon. It resembles traditional board games like chess, draughts or reversi, in that it is played on a square grid onto which pieces—in this case, bombs—are placed. As the squares fill up with bombs, explosions occur, and bombs change ownership. The object of the game is to blast all of your enemy's bombs off the board.

The graphics are workmanlike: not flashy, but they are clear and do the job. The sound is also limited; you hear clicks when bombs are placed, and brief explosions when appropriate. The sounds is also well behaved, and can be switched down or off *before* the first noise is played.

The game is not particularly addictive, in that it's easy to put down when you've won or lost a game. But like all board games, it has replay value, in this case hampered only by the lack of configurable difficulty levels for the computer player. It would be well described as a time filler for idle moments, rather than something where you're desperate for just one more go.

One thing that impressed me is that the game runs on all EPOC machines: the three versions cover all machines from the Osaris to the Netbook. I was a little disappointed by the graphics on my Series 7, though. The colour graphics are the same size as on the Series 5, so the board looks a little lost on the big screen.

The game is fast, reliable and kind to your computer. The computer players make their



move very quickly, and the graphics always draw instantly. The game takes up very little memory, and responds properly to system events, like closing down from the main screen. The user interface is familiar, having a toolbar and standard menus with standard keys. The only minor niggle I have is that some options call up two dialogs in a row (e.g. Player Settings), when it would be more straightforward for each dialog to have its own menu option.

The program is laden with features. Different board layouts are available, including special squares that affect how the bombs behave. There's even a level editor to allow you to design your own board. The game also includes a help file, and has reasonable access to preference settings.

All in all this is a good, playable game, that I can recommend to anyone who likes the board game genre. Though I couldn't see myself playing it all afternoon, or even for an hour at a stretch, I do tend to load it up often enough that it has earned a permanent place on my CF disk.

Chain Reaction
Shareware
Adam Dawes/Neuon,
http://www.skankee.com/neuon
☆☆☆☆

Animating OPL

A series by Damian Walker

This is the first in a multi-part series on programming animated graphics in OPL, the built-in programming language of EPOC. The tutorial introduces the practical topics of creating your own graphics, and using them in your programs. It also discusses some more advanced techniques for creating good game graphics, and some ideas about designing games to fit within the limitations of OPL and the machines it runs on.

To follow this tutorial you'll need to understand some OPL already. You'll need to understand procedures, loops and variables. If you could write a simple game of guess-my-number, then you'll be fine. If not, then the Basics chapter of the OPL manual will bring you up to speed. You'll also need the Program and Sketch applications installed. Most EPOC machines have them as standard, but if you have a Revo, you'll need to install the freely available SIS install files yourself.

The Project

Back in the days of the Series 3, the Psion programming manual used to include a bouncing ball animation. This was very rudimentary, with the ball being represented by a letter “O” being alternately printed on the screen and overwritten with a space character. It would roll around the screen, bouncing off the edges as it hit them, until the viewer got bored of watching and stopped the program.

We can do better than that now—indeed, the Series 3 could do better back then. We'll be using proper graphics for our demonstration. Our ball won't appear on a blank screen, but will be on a tiled floor. And it won't roll around, but will bounce up and down on that floor as well as bouncing off the edges of the screen. While this still isn't particularly exciting, it does let us focus on the animation techniques rather than on impressing an audience. There'll be plenty of time for excitement when you're happy with the graphics.

Creating the Bitmaps

There are two elements to our demonstration: the ball and the tiled floor. Before we start programming, we'll get the drawing out of the way, and this is where Sketch comes in. Before starting Sketch, create a folder \Bouncer on your preferred drive. This is where we'll store everything associated with the project. Create a Sketch file, using whatever name you please. The name doesn't matter, as OPL will be using OPL Picture files that Sketch exports (also