MENTAL PROCREATION

By Andrew Braybrook

Joyous news! Once again Andrew Braybrook's imagination is 'up the junction' . . . (pregnant pause) . . . Yes, there's another binary bun in the oven – and over the following months you will be able to witness, first hand, the agonies and the ecstasies Andrew experiences . . . From conception, through gestation, to the labours of birth. In this heart-rending first installment, Andrew ignores his maternal cravings for lumps of code and gets to grips with ante-natal depression . . .

Thursday 4th December

I've finally tidied up the loose ends on *Uridium Plus*, and the American version of *AlleyKat* and *Uridium*. This took a while because I've been installing the new anti-cartridge system. Since most of these devices are used primarily for piracy I feel that it is necessary to spend this time, although I'd rather not have to. Such cartridges have no place in the grand scheme of things – if you want the disk version then buy the disk version. If your tape breaks, send it back for replacement. If you wear it out, buy a new one. It's the same with records.

None of this would be necessary if these cartridges were individually numbered, like certain modems, and had to be in place to reload backups. I suppose manufacturers don't do this because people wouldn't spend £35-50 on a back-up device that couldn't make that money back somehow. I wonder just how many people have spent that much on replacement originals. Anyway, I hope that I don't have to spend too long protecting the new game, as it's just an annoying waste of my time.

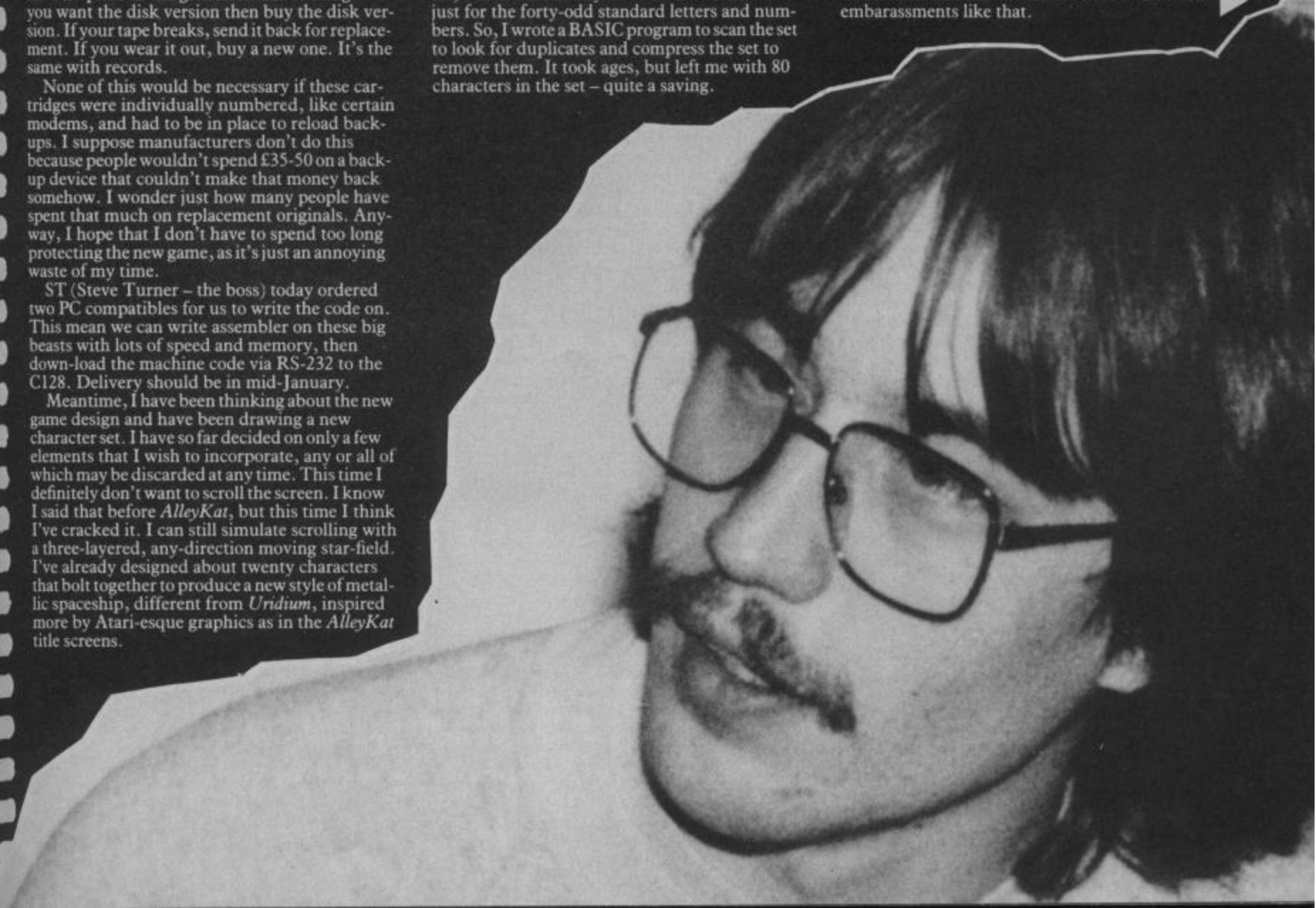
ST (Steve Turner – the boss) today ordered two PC compatibles for us to write the code on. This mean we can write assembler on these big beasts with lots of speed and memory, then down-load the machine code via RS-232 to the C128. Delivery should be in mid-January.

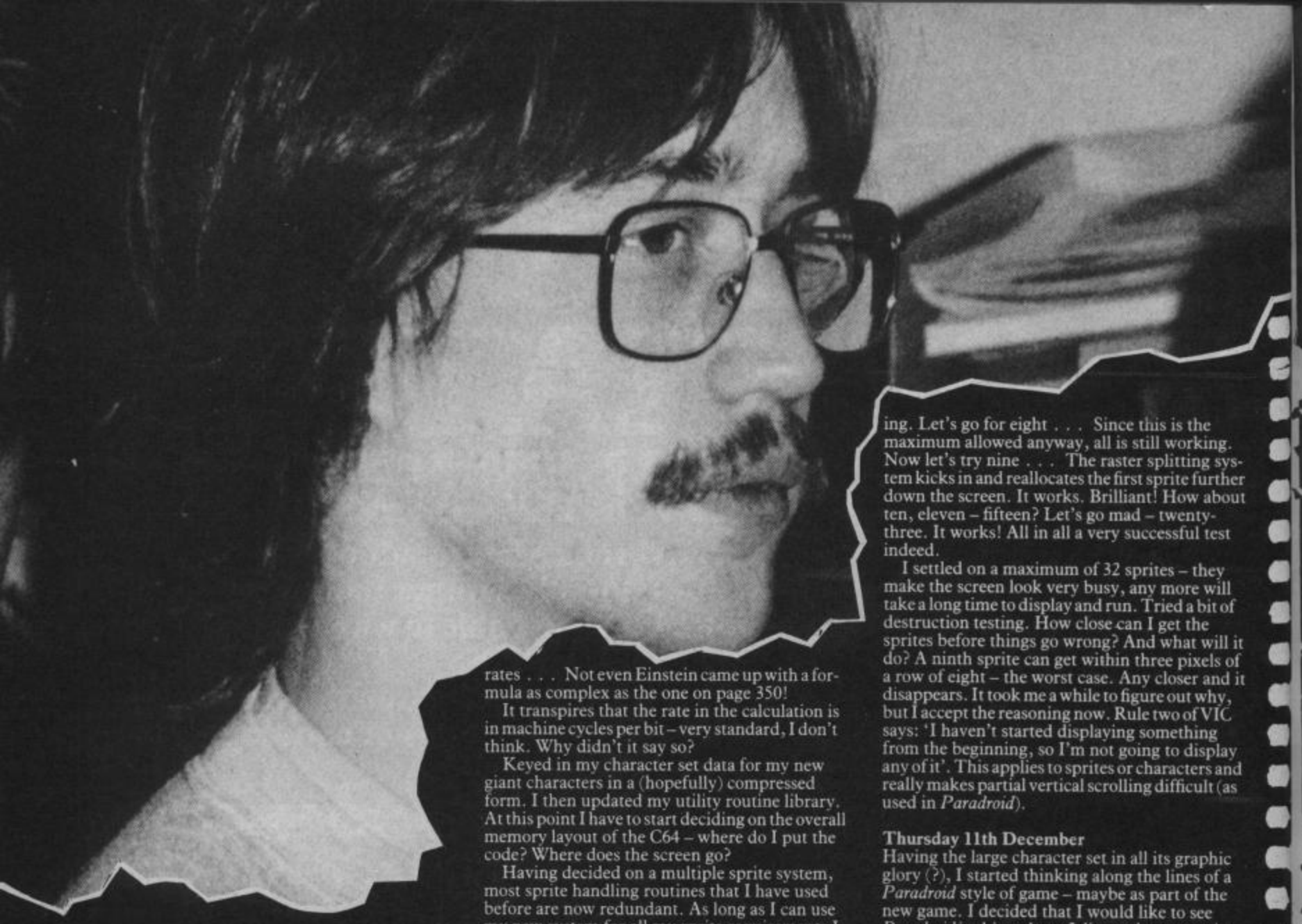
Meantime, I have been thinking about the new game design and have been drawing a new character set. I have so far decided on only a few elements that I wish to incorporate, any or all of which may be discarded at any time. This time I definitely don't want to scroll the screen. I know I said that before *AlleyKat*, but this time I think I've cracked it. I can still simulate scrolling with a three-layered, any-direction moving star-field. I've already designed about twenty characters that bolt together to produce a new style of metallic spaceship, different from *Uridium*, inspired more by Atari-esque graphics as in the *AlleyKat* title screens.

" . . . I'm thinking of running about 32 sprites simultaneously, without glitches or airborne trousers . . . "

I also designed a giant bolt-together character set, which used nearly all of the 256 characters just for the forty-odd standard letters and numbers. So, I wrote a BASIC program to scan the set to look for duplicates and compress the set to remove them. It took ages, but left me with 80 characters in the set – quite a saving.

I've started thinking about sprite usage – and since I like the use of no on-screen distracting text I don't need a raster splitting system to split the screen. The 'in' thing seems to be to use more than eight sprites on screen, so I'm thinking of running about 32 sprites simultaneously, without glitches or 'airborne trousers'. Stand on top of the first tombstone in *G 'n' G* and wait for a couple of minutes for the latter effect! I think I can avoid embarassments like that.





Friday 5th December
Got to grips with the thirty-two sprite system. It has to reassign sprites to new objects after it has finished displaying the old object. This requires some fancy raster splitting. Such problems as 'What if two sprites need assigning at the same time?', or 'What happens if more than eight sprites need displaying on the same line?', or 'What happens if I move an object while it's displaying it?', or 'What about different sprite colours and display modes?' all spring to mind.

I've spent most of the day writing this on paper, refining it on paper, realising it won't work, redesigning it and mentally testing it. I think I've now got a Mk 1 system that is adequate for testing the theory. Unfortunately I need quite a hefty system in the C128 just to test these three pages of code. Things like a character set, monitor, and text routines are all required right at the start. With the new large letters I need to rewrite my character and text handling routines. I also need a test-bed to get the screen in the right place and mode, preserve the monitor's variables, fire up the interrupts, return to the monitor when required and display diagnostics. This is all before I get to even do any actual game processing! I'd better start thinking of a game name too, as it's very important to get the right name, create a good atmosphere and get things moving in the right direction. After all, it's very difficult designing a title screen when you don't know what to put on it!

Monday 8th December
Spent some of the morning trying to decipher the RS-232 section in the Programmers Reference Guide. All the decent baud rates are marked NI

"Not even Einstein came up with a formula as complex as the one on page 350!"

which we believe is Not Implemented. Thanks, Commodore. I'm not waiting four minutes for any program to arrive from twelve inches away at the PC, so we investigated the user baud rates . . . Not even Einstein came up with a formula as complex as the one on page 350!

It transpires that the rate in the calculation is in machine cycles per bit – very standard, I don't think. Why didn't it say so?

Keyed in my character set data for my new giant characters in a (hopefully) compressed form. I then updated my utility routine library. At this point I have to start deciding on the overall memory layout of the C64 – where do I put the code? Where does the screen go?

Having decided on a multiple sprite system, most sprite handling routines that I have used before are now redundant. As long as I can use my new system for all my sprite requirements I shall be alright.

Tuesday 9th December
Read up a bit more on RS-232, including how to get it talking to PC compatibles. Now I'm even more confused. A 25-pin connector, ideal for parallel data transfers at high speed, and how many wires do they actually use? Three! Brilliant!

Keyed in enough code to get a test-bed running just enough to see the screen, colour it, write on it, and return to the monitor. Forgot to unblank the screen. Twice! The giant character set looks great, no mistakes, just ABMON printing two non-existent characters.

In this game I need to print a 'null' character, ie: nothing at all. This is important. In every game I've done so far I needed to do this, but never got round to it. It may seem a waste of time printing nothing, but it is in fact very useful. For example: imagine the line '3 ships left'. When there's only one ship left it looks stupid saying '1 ships left', and I hate '1 ship(s) left' as it's a lazy cop-out. So, you need to be able to eliminate the 's' on the end of 'ships'. You could blank it out with a space, but then it looks untidy because there are two spaces instead of one. So, you need to substitute a null character.

Following so far? Yes, I am very fussy, but I take great pride in my work – not a pixel out of place, nor a line of text uncentred.

Wednesday 10th December
Today's the day – take the plunge and key in the multiple sprite routine. I always write the complicated routines on paper first. This gives me time to think of all the things that might be wrong. Thus, a routine is fairly well debugged by the time I type it in. This does result in a very untidy-looking piece of paper though – keying-in time is a time for final mental debugging.

I fired up the test-bed with no sprites to start with – let's not get too ambitious. The system is running OK, so any crashes now will be caused by the multi-sprite system. I set up some sprite positions in the table and switched on a sprite . . . It appeared. Wonders will never cease! Try two sprites . . . three, four. All working. Let's go for eight . . . Since this is the maximum allowed anyway, all is still working. Now let's try nine . . . The raster splitting system kicks in and reallocates the first sprite further down the screen. It works. Brilliant! How about ten, eleven – fifteen? Let's go mad – twenty-three. It works! All in all a very successful test indeed.

I settled on a maximum of 32 sprites – they make the screen look very busy, any more will take a long time to display and run. Tried a bit of destruction testing. How close can I get the sprites before things go wrong? And what will it do? A ninth sprite can get within three pixels of a row of eight – the worst case. Any closer and it disappears. It took me a while to figure out why, but I accept the reasoning now. Rule two of VIC says: 'I haven't started displaying something from the beginning, so I'm not going to display any of it'. This applies to sprites or characters and really makes partial vertical scrolling difficult (as used in *Paradroid*).

Thursday 11th December
Having the large character set in all its graphic glory (?), I started thinking along the lines of a *Paradroid* style of game – maybe as part of the new game. I decided that I would like to see *Paradroid* in this style, so I diverted my attention to dusting off the old *Paradroid* source files. This turned out to be a lengthy job as I started thinking of all sorts of 'What if . . . 's and did about eight half-hour assemblies. At one point I had *Paradroid* running at 50 game cycles per second (three times faster than normal), the same speed as *Uridium*. It took a while to familiarise myself with the memory layout, as it's rarely the same in two consecutive games.

"My disk drive 'machine-gunned' twice and ruined two consecutive assemblies right at the end. AB is not impressed."

Took *Paradroid* home to work on it. My disk drive 'machine-gunned' twice and ruined two consecutive assemblies right at the end. AB is not impressed. The red light flashes a bit on loading, but it still works. It's getting unreliable though. That's all I need . . .

Friday 12 December
Spent most of the day redrawing the *Paradroid* graphics in the new style. After correcting the colour table, which displayed the most grotesque green selection anyone has seen since the Dragon 32, it gives a new lease of life to the game. It's a pity that it's too late to use it in the Christmas double-pack, but may be just as well – I'll let my new game show the graphical advance. After all, the object of this exercise was only to evaluate the new graphics properly.

Saturday 13th December
Completed work on the new graphics as inserted into *Paradroid*. It looks very neat and proves that the new graphics work, although finding colour schemes that fit is more difficult. They represent a step forwards from the *Paradroid/Uridium* style but retain the shiny metallic look.

Monday 15th and Tuesday 16th December
Went to ZZAP! Towers for the Newsfield Christ-

mas Party. Watched certain notables consuming silly expensive drinks, and generally had a good time. Stavros Fasoulas and I decided that redefinable keys are a pain and tried to find the name of the only Commodore owner without a joystick – which should be easy as he'll be the only one still playing *The Sentinel*.

Wednesday 17th December
Decided to take the multiple sprite system one stage further to get it running more or less as it will in the final game. This requires strobing between two position tables, the interrupts picking up the sprite positions and displaying them from one table, while the main program works out their next position in the other table . . . Well it seems pretty basic to me. Fired it up to be greeted by a blank black screen. No return to monitor is possible. The code all seems intact but the interrupts aren't running anymore. I haven't a clue as to why not – it's not as if I've altered a lot in that department.

Thursday 18th December
Found an error in the sprite assignment system which meant it was trying to update a large-numbered sprite's position. This is more than eight anyway, so some important video chip registers were getting corrupted. This resulted in, amongst other things, a blank screen.

Tried to get it running moving sprites on the screen, but it seems that a routine that puts the objects in the correct sequence ready for sprite assignment isn't doing its job properly – it's doing a *Commando*, ie: the display glitches every couple of seconds and objects disappear. Unlike *Commando* however, this bug *will* be fixed!

Still putting together ideas about running the objects in various tables, I have an object table of 'n' elements, some of which will be given sprites because they are near or on screen. These will go into one or two alternating (or strobing) tables. The active table of these two is then partially copied out into another table for displaying every 50th of a second. The active table then swaps places with its passive sister table for the next 50th of a second. You probably won't be surprised to learn that co-ordinating all of this gives me quite a headache!

I also want to use Polar vectors for the ships, not X-Y vectors for the movement. I think it'll give them more realistic movements. I may use 'bradians', which involves doing nasty things to radians to get 256 degrees in a circle rather than 360. See Gary Liddon for further information on 'bradians'.

Friday 19th December
Updating all these objects seems to take up a large amount of processor time. It seems I've only got time to update about 22 simple moving objects every 50th of a second. I'll have to make the code more efficient to run any more than this.

I did manage to get different coloured squares floating quite merrily around the screen, and it does look very busy. Rearranged a few instructions to stop it from glitching.

Monday 22nd December
Got rid of the last screen glitch and refined the object handler. It can still only manage 24 objects comfortably. I may have to run the objects every 25th of a second. I'll have to experiment.

Scribbled notes on running polar vectors. These are not distant cousins of polar bears but a way of representing movements in terms of an angle and a velocity, rather than an X speed and a Y speed as I've always done before. The latter approach tends to give unnatural diagonal movements. I shall still use X and Y co-ordinates to log positions, but I will have a giant sine table to look up angular movement speeds. Calculating them would be far too slow. Steve Turner wishes it to be noted that he doesn't hold out much hope for this. He only wants to say 'I told you so.' in a couple of months time!

Tuesday 23rd December
Winding down for Christmas. Had the Company Christmas lunch up at Chequers, and very nice it was too. Tried to get the C64 talking to the Spectrum via RS-232, but they weren't listening to each other.

Wednesday 24th December
Mickey Mouse award of the month goes to the inventor of RS-232: the standard that doesn't have a standard. Let's have an expensive 25-pin connector and only use eight of them. Let's use cables with only eight wires to make future

"Anyone know how to connect an Opus PC-II to a C64 via a parallel data link?"

expansion impossible. Let's use sockets on some devices and plugs on others so that people need a whole host of different leads. And to cap it all the Spectrum has its own standard four-pin connection! Sparse documentation on the subject gives the impression that no-one really understands it anyway. Anyone know how to connect an Opus PC-II to a C64 via a parallel data link?

Monday 5th January
Over-slept due to being out of the habit of getting up much before mid-day. Had lengthy discussion with ST about how bad RS-232 is (still) then decided to abandon it. Looked up the specifications of Centronics interface and compared these to the specifications of the C64 user port. Looks pretty similar to me – and it's parallel. A whole byte at once! Pretty amazing!

Everything we've read implies that it is possible to connect the two. Why has this idea been kept from us? Why does everyone rave over RS-232 when it just doesn't work?

Revised my sprite sorting ideas after discussion with Paul Hughes and Gary Liddon. I will get 32 sprites running with plenty of time to spare. I'll try a faster sort first, and if that doesn't speed it up enough I'll have to rethink the whole system.

Tuesday 6th January
Battled with the new sort routine some more. Got a couple of BSS errors – the ubiquitous 'Blank Screen and Stop'. Finally sussed out what was causing it, but not necessarily why! It makes a dog's dinner out of sorting the sprites, and causes the them to flicker occasionally. Yeuck. It doesn't half fry the old grey matter when you have lists of pointers pointing to other pointers in the same list, and also to other tables. Reading what I've just written, it now dawns on me what I've been doing wrong. All that sorting by Y co-ordinates is undone by the final extractions routine that unsorts everything back into jumbled up sequence (!). Brilliant!

Wednesday 7th January
Stayed up 'til two o'clock in the morning battling with the sprite sorter. Finally got it to display 32 sprites on screen at once, providing the first one appears last on the screen. Very peculiar. Unfortunately it doesn't leave much raster time to do anything else, such as run the game – in about an inch of raster time on 14" portable!

"No bubbles, no insert sorts, no Shell Metzner (who?), no McPherson Struts . . . no matter – my logical chain sort isn't quite working anyway!"

Got a 'phone call with all the answers at about 11 o'clock this morning. "Why don't you try this . . . " were the pearls of wisdom. Eventually ascertained what was going on, and it doesn't involve a sort routine at all. No bubbles, no insert sorts, no Shell Metzner (who?), no McPherson Struts . . . no matter – my logical chain sort isn't quite working anyway!

Rearranged the program a lot and then rearranged it again after realising that the first method was very silly indeed. It was setting up a new table on top of an old table, which the interrupts were still trying to read! This is what causes what IBM call 'unpredictable results'. Personally, I find the results very predictable – it always goes wrong!

Now I've got my 32 sprites whizzing around the screen looking like an abstract asteroids game, I've got at least four inches of raster time to spare, and I'm very happy. With a bit of optimisation I'll get some more time back and I can start on the moving background system.

Thursday 8th January
Went to Hewson's New Year launch in Picadilly and met some of the Press. Finished the day with a visit to a couple of arcades in London to see what's new on the 'real' machines. There are some great games about with superb graphics like *Slapfight*, *XX Mission*, and *Salamander*. It really makes us weep to see what these machines can do. Saw the new hydraulic-seated driving game with probably the same video chip as *Space Harrier*. This produces a very convincing environment, mostly with giant sprites. The gap between arcade technology and home micro technology is widening.

Friday 9th January
Our new Opus PC compatibles have arrived! ST has one of them working and is trying out some of the software. They're quite large and have a rather loud fan inside for cooling purposes. There's also plenty of reading matter to plough through before we can get working on them. On the game front, I've optimised the sprite code a bit more and it's all running smoothly with no apparent hiccups.

Monday 12th January
Snowed in at home. Luckily I have my working disks at home so I can carry on – but without my notes. Ah well.

To speed things up even more I had a quiet think about sprites. My current system sets position, colour, sprite image and mode in either hires or multi-colour. I realised that any one sprite image will always be in the same mode, and usually the same colour. Thus, it is necessary for the object mover to worry about colour and mode – the display routine can just pick these up from a table. I can update this table if an object needs to change colour.

Tuesday 13th January
Still snowed in. I still haven't drawn any sprites that I need – I'm not even sure *what* I need! I want to steer clear of space-ships and robots as these have been used to death on the 64 over this last year.

Had a lengthy session on the sprite editor, just doodling. I know what I want but I can't really draw it. The editor doesn't quite reflect what my game needs, ie: a different colour for each sprite, which makes it's difficult to check animation.

Wednesday 14th January
Been thinking about the background. Again, I know what I want, but I have to think of a different and plausible scenario. As this type of display hasn't been used before, to my knowledge, I don't even know if the game design will work! If I go ahead it could be two or three months before I know for sure either way! This requires more thought, and perhaps some discussion.

Thursday 15th January
Had the brilliant idea of using a bit-map screen to get more colour out of the machine – three independent colours per 4 x 8 pixel square. This is appealing, but it costs 9K in the 16K video bank. That leaves 7K for sprites, or 112 images. In character mode it would only use 3K, leaving 13K for sprites – or 208 images. Do I really need the extra colours? Looks like I'll have to live without them.

ST phoned in a jubilant mood in the evening to say that the Spectrum and C64 are now chatting to each other – both ways, in parallel *and* reliably! This bodes well for connecting the Opuses (Opae?) to the target machines. Bye bye RS-232 . . .

To be continued . . .

Will Andrew and Steve ever get their 64 freely conversing with an Opus? Or are they both mute? Find out next month . . .

MENTAL PROCREATION

By Andrew Braybrook

Like all good prospective fathers, Andrew Braybrook finally decides upon a name for the new offspring. Uncle Steve continues counselling the Opus and the 64 in the hope that they will start talking to each other, and a program to double the speed of the C64 is written. There's more drama to writing a game than you find in an average episode of EASTENDERS, as the second part of Andrew Braybrook's diary reveals . . .

Friday 16th January
ST (Steve Turner) has been setting up the operating system on the Opus (called MSDOS) ready for our cross-assemblers, which assemble either Z80 or 6502 on a machine not running under those chips. Having obtained various library books on 8086 (the PC processor) and communications, we are hopeful that communications between our computers will be easy to set up. We just need to set up some leads from the Opus Centronics printer output (non-standard socket of course, courtesy of IBM), to the C64 and Spectrum. Two PCs running at the same time make a lot of noise with their cooling fans.

Monday 19th January
Did a nice sequence of eight sprites which fit on top of each other. They look very swish when colours are faded through them. Bit expensive on sprite usage though, maybe I can come up with a way of running something similar with characters.

Having seen Gary Liddon's sprite multiplexor on Saturday, I can't understand why his runs faster. Committed a no-no by using his method of building a table in the system stack so I can use faster instructions to read the table. It would have been very messy if it hadn't worked. Don't tell ST, I'd get the sack for misuse of the computer. It still isn't as fast as Mr Liddon's system, but his isn't doing as much yet. . . . I hope!

Tuesday 20th January
Discussed the possibilities of a game where the enemy show a bit of evolutionary behaviour, as the weaker ones get destroyed they are never replaced, but ones that get away grow stronger and randomly mutate, as featured on Horizon yesterday. Maybe even the graphics could be generated too, giving every game a different look, depending on the performance of the player in the particular game.

Scribbled some code for the moving background – looks like a lot of code with not much time to execute it in. I may have to use a 24 line screen rather than 25 to buy myself some extra time with the raster off the screen.

Also keyed in a machine code Centronics output routine for faster communication to the Spectrum. We're still awaiting the data sheets for the chip in the Spectrum interface, so it wasn't working too well – the Spectrum's a little slow at noticing that a byte has arrived.

" . . . it worked almost perfectly, except one set of lumps kept stepping backwards every now and again and some others left trails of debris behind them . . . "

Wednesday 21st January
Keyed in the moving background system and it worked almost perfectly, except one set of lumps kept stepping backwards every now and again and some others left trails of debris behind them. Didn't take long to fix though. This screen updating of thirty objects just about takes place while the raster is off-screen, so no flicker can possibly happen.

I've also been wondering whether to display the score on-screen and if so, how. Then I realised that I could just write the score in standard characters – I've always assumed that I'd have to do it with Sprites. Sprites may still cross the score but they will be on the move. I'm unhappy about using sprites in the border as it could be exploiting a video-chip bug. This may not work on some machines or it could cause damage.

Thursday 22nd January
Flushed with the success of yesterday's background routine, I set about defining what I want to see on the screen, the control mode required to run it, and how I'm going to achieve this. I need to be able to move in all directions, fire in all directions, dematerialise, and select quickly which of these I want to do, all from an eight-directional joystick and one fire button. I shall attempt to put this multitude of functions into the game without the use of windows, icons or mice, and only quietly muttering at the lack of foresight to put two independent buttons on a joystick.

Designed some more of the character set to show these various systems, I also think I've come up with the game name. As I was flipping through the dictionary for inspiration I came across 'Morpheus – God of Dreams'. It's about the right length, sounds nice, easy for advertisers to misspell and generally mysterious, so that's what the game will be called, **Morpheus**.

I'm also conscious that the on-screen display must be kept moving and changing to maintain interest. I think I'll need a more complex than average character update system to animate parts of the display with lots of frames of animation.

Friday 23rd January
No progress on **Morpheus** today, battled with US *Uridium*.

Monday 26th January
Spent much of the weekend working on US *Uridium*, and today I'm getting *Uridium+* ready for going to the US too: I'm sure it'll enjoy it there, it could do with a holiday.

It looks like the designers of the 8086 chip in the PCs are going for the January Mickey Mouse award, with its sixteen byte segmented memory arrangement. Any byte in memory can have thousands of different addresses. Very confusing.

Tuesday 27th January
Back to the real business of **Morpheus**. I want lots of different 'meanies', whatever they turn out to be, animal vegetable, mineral or atomic. What I want to do is build objects from lots of source objects, like building a sprite house from sprite bricks and sprite windows. By using X and Y reflections and specifying an artificial 'depth' or 'priority' for each image, I should be able to get proper animation with objects passing behind others in the same sprite. It will have to build the images in advance, say between levels rather than immediately as required – as combining up to eight images per frame of a sixteen frame sequence could take a lot of CPU wellie.

" . . . With the aid of a mega-soldering iron, pliers and our frustration brick we cracked it . . . "

Wednesday 28th January
Been thinking about yesterday's sprite combining system. It's important to think through the limitations of a system before writing it, in case it won't do what you want. It also gives me an opportunity to expand on the idea to explore what the system will do. I'll have to resurrect my sprite reversal routine from *Paradroid* which produces the large robot pictures. Reflecting multi-coloured images is not as straightforward as reflecting hi-res sprites.

ST and I battled with some parallel communications this afternoon to get the Opus talking to the C64. We had a rare collection of errors, including the C64 missing out bytes because it wasn't fast enough, the C64 duplicating bytes because it was too fast, total lock-outs where both machines were waiting for each other, and the C64 only receiving one byte per millenium. With the aid of a mega-soldering iron, pliers and our frustration brick we cracked it. We can transfer data at about one and a half kilobytes per second, which is the fastest the Opus will currently do. We should get it faster with our own dedicated routine, if ST wants to learn 8086, I don't.

Apparently on the Opus you can print in foreground (ie everything stops until it's finished) or in background, that is you can carry on as normal and the interrupts routines just get on with it. MSDOS takes the January Mickey Mouse award for its idea of background printing. it accepts keyboard input at a magnificent rate of one character every three seconds while background printing. Not what I'd call user transparent.

Thursday 29th January
Wrote a utility to read data from the Opus in ASCII format and convert it to machine-code as it reads it. Since it keeps getting checksum errors because it is missing bytes, I can only assume that I'm not getting back to read the next byte quickly enough. The Opus is like a machine gun and I don't have time to catch each bullet, polish it and place it neatly, I'll have to catch them all in a bucket and tidy them up while it's reloading. Most frustrating because it very nearly works.

Friday 30th January
Day off.

" . . . Further investigations reveal that a certain C64 operating system delights in polling the keyboard every 60th of a second, despite being told not to by me . . . "

Monday 2nd February
Re-arranged my Centronics receive code to read a series of bytes at a time, stop transmission and decode them. It still occasionally 'misses' six or seven bytes. It either works perfectly or misses a whole lot, nothing in between – this is suspicious. Further investigations reveal that a certain C64 operating system delights in polling the keyboard every 60th of a second, despite being told not to by me. This has only started since I've been calling the write character to screen kernal routine, CHROUT, which must be enabling the interrupts again. How good of it to take this decision all by itself! Thus my receiving success or failure depends on whether I get interrupted while reading data. The interrupt lasts for so long that I lose about six bytes. With a very small test file it seems about an evens chance, but a larger file would never get across intact. I can soon fix this, just tell the timer chip (the CIA) to stop screaming blue murder every 60th of a second, rather than just sticking cheese in the system's ears to stop it hearing and responding.

Now that it appears to be working it also explains why it didn't work last Thursday, which was a better system. Now I've changed it I'm not going back. Just like my macro assembler loaders, it'll type out full-stops as it loads, except mine will allow loading in RAM under the ROMs and protect against overwriting certain delicate areas in the machine.

The Opus is now playing up, it refuses to link some Z80 code for my test file: it just crashes the whole machine. Perhaps it knows that the C64 wouldn't know what to do with it anyway. That's it, the Opus is emulating the C64 perfectly, it crashes! I only wanted to use Z80 because ST has already keyed in some Spectrum routines.

Tuesday 3rd February
Completed work on my parallel data receiving program and tested it out by passing a test file down from the Opus. All appears good. Right, time to take the plunge, retire the old C64, to be replaced by the new Opus PC, (I've been using ST's Opus thus far), and connect it to the C128. The MPS801 printer will have to go as well since there just isn't room, Opae are big beasties indeed. The 1570 disk drive can sit on top of the Opus, connect it all together, light the blue touch paper and bingo, a Graftgold Development System.

Got a quick tutorial from ST on using the text editor and MSDOS in general. So begins the task of rekeying all that I've done on **Morpheus** so far into the Opus. It's nice talking to a man's computer where software is written with no space constraints – a mere 492K spare for editing files, another 392K on the RAM disk, two disk drives with about 360K each, and an operating system that does what you want, when you want – with no fuss! Brings back memories of the old IBM mainframe (when it was running full tilt after five o'clock with decent IBM terminals and CMS).

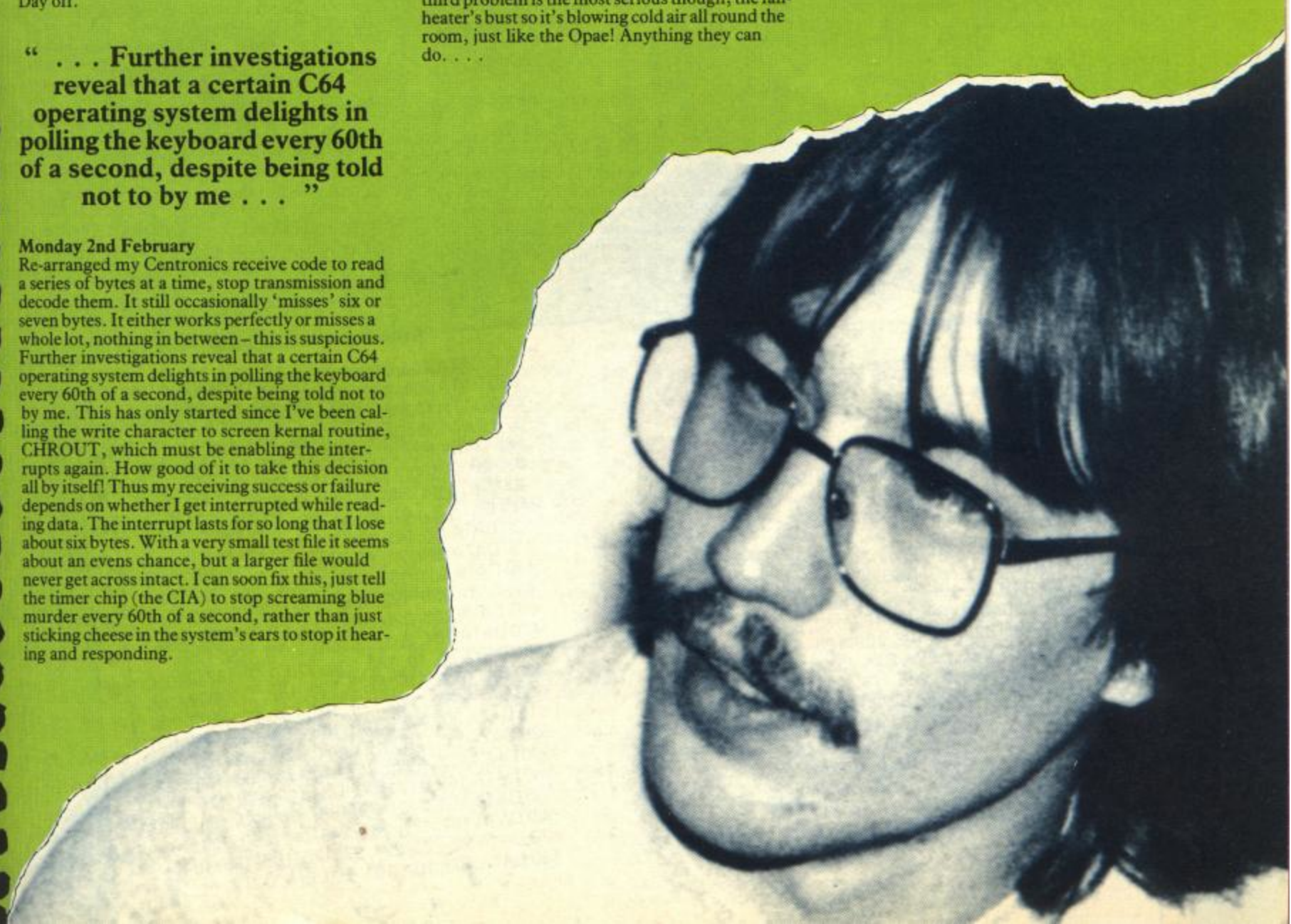
Three problems have since reared their ugly heads, *Easyscript* won't load, so I can't see what I'm going to type in on the Opus, and the old 1541 won't read any disks at all. It knows it's being retired to the great disk drive home in the sky, to become just another ordinary breeze-block. The third problem is the most serious though, the fan-heater's bust so it's blowing cold air all round the room, just like the Opae! Anything they can do. . . .

" . . . if God had intended us to use serial data, he'd have only put one bit in every byte . . . "

Wednesday 4th February
One quick amendment to the receiver system, don't allow it to load data over itself and crash, I've done that with my C64 assembler loaders a few times this week.

Some people may be saying "How can it take these people so long just to connect two computers together?" Whilst appreciating that it's not exactly on the forefront of technology and one of the selling points of computers is the ability to interface them to anything – just you try and find someone who'll stand up and admit that they've connected two different computers together in parallel. Many people 'in the business' seem to use serial links, but if God had intended us to use serial data, he'd have only put one bit in every byte. As we've already discovered, serial linkage is no easier than parallel, is slower, and has a worse-defined standard. We have connected an Opus PC-II to a C128 in parallel using only information in the Programmers Reference Guide, some timing and circuit diagrams of a Centronics printer and applied logic. We've now probably got one of the more advanced development kits in the country for micro software.

Thursday 5th February
Continued keying in my source code to the new assembler. It's mostly much more versatile then the CBM one, but it does have one or two inefficiencies. Whereas I used to type < or > signs for low or high bytes, I now have to key 'Low' or 'High' in full, not even 'Lo' or 'Hi' will do. I'm also learning to use the new proper full-screen editor. Imagine a delete key that actually deletes the character under the cursor, not the one before it: how revolutionary, it'll never catch on! I remember the IBM mainframe that I used to work on did that – a real home from home this ▶



is. Finally got my utilities all keyed in and my variables and macros. I made some improvements to some tacky bits of code whilst rekeying it so I wouldn't be able to check it against the original version when it's done, very clever I don't think. If anything goes wrong it'll be harder to find the errors. It's great being able to put huge comments in the code – on an 80 column screen it still looks tidy. It was impractical to get carried away on the C64 as it hampered loading time and took up too much space. All the source for one game used to take over 500 blocks on a disk, or about 120K.

Initial timing tests show that to assemble and download my utilities takes about half a minute, not fast by any means, but compared to about five minutes on the C64 it's great. It'll be interesting to see how long it takes on a finished program, which used to take the best part of half an hour on the C64. If it's still ten times faster it'll only take three minutes.

One thing about the assembler is that it really goes to town if it finds an error. It usually manages to find at least five things wrong on the line, just from one letter being mis-keyed. If it only printed four error messages it'd probably print a fifth moaning that it could only find four things wrong!

A final piece of good news, the fan heater is working again.

" . . . due to a certain wire not being connected at the Spectrum end, the Opus thought that the Spectrum had run out of paper . . . "

Friday 6th February
Feeling rather ill today, but it only hurts when I move. Struggled in to work and keyed in all the remaining source code to the Opus, assembled it, corrected all the typing errors, assembled it again and booted it down to the C128. It all ticks over very nicely.

Spent much of the afternoon looking for the bug that caused the sprites to flicker horrendously. Finally found it, I'd just mis-read one character from one screen while typing on the other.

ST found out why the Spectrum won't listen to the Opus whereas the C128 will, it's a good one this. Since the Opus thinks it is printing to a printer, and our micros are impersonating printers, due to a certain wire not being connected at the Spectrum end, the Opus thought that the Spectrum had run out of paper! Thus it wasn't sending data across. A simple soldering job cured that and the two are now happily communicating.

Hopefully by now some of you will have played the Competition Edition of *Paradroid* in the Christmas double pack. Being 50% faster than the standard edition, it certainly plays a lot more furiously. I discovered how to make *Paradroid* run fast after I finished *Alleykat*. I was experimenting with the double-speed CPU in the C128 which is an adaptation of the 6510, called the 8510. It can run at a clock speed of 2MHZ if required, but normally it runs at 1MHz for C64 compatibility.

Upon further investigation, I realised that the 6510 present in all C64s could also be encouraged to run at this higher speed by use of software. I have produced a BASIC listing to allow you to apply this discovery to your own software. Just key in the program and save it to tape or disk. RUN it, and then NEW it. Finally load in any game from cassette. Once loaded, it will run at up to double normal speed, prepare for a mega-fast game! Since the 6510 Accelerator upsets disk loading speeds, it cannot be used to speed up disk-based software yet, but I am working on a solution to this.

By the way, line 90 contains some cursor commands in quotes. The PRINTed line is 3 cursor downs, then NEW, then three cursor ups to put the word NEW on the CURRENT line just to remind you to NEW the program. All you do is press ENTER.

Monday 9th February.
It's been a slow day. Began by writing some more notes on the sprite combining system. I suppose the idea came from reading about 'Blitter' chips, the way you just point them at up to three sources of data, one destination for the finished data and tell it what logical operations to perform, and it does it. It can do anything from simple data copies to complex object plotting. Wish there was a blitter in the C64.

Now that I have a game name I can begin thinking about title screens. I've re-arranged some of my large character set to allow it to be used with my moving background system. I shall also need a snazzy high-score table of some kind.

Tuesday 10th February
Got to grips with the multi-layered grid that moves at different speeds, done with characters. Sadly it doesn't have CPU time to build and display it in real time, and not even at half-speed, doing half of the grid on each cycle. It is a nice effect though, so I'll have to work out another way of displaying it. It's a lot faster to debug with the source code in front of you on one screen and the game running on another.

Redrew the small-lettered character set and the game logo that will appear on-screen during the game to make them easier to read. It's the first multi-colour resolution character set that I've ever done, *Alleykat* was in hi-res but with a fancy Atari-emulation system to make it look like more colours.

" . . . I hate line numbers! They were only invented for punch cards so that if somebody shuffled them you could sort them out again . . . "

Wednesday 11th February
Re-organised yesterday's grid-drawing routine so that it calculates all the animation frames in advance and can then display them at any speed. Thus it takes very little CPU time to display and gives me time to run the moving blobfield and thirty-two sprites. Of course it does take an extra 4K of memory – you don't get something for nothing. The constant battle is always time against memory; you can write a routine that's fast but takes a lot of memory, or write it to take less memory but it executes a lot slower.

Re-arranged the character set (again) to group together all the pieces that are similar in what they will do. This is so that anything such as collision detection needs only to check a range of codes, not a series of individual codes dotted around all over the place.

Tried to think of a neat way of dematerialising the character set that hasn't been used . . . failed. I could switch in a new character set, or a new screen, or both. Maybe a *Uridium* dreadnought dissolve would look nice.

One thing about our new Opae, the editor has no line numbers. Great. I hate line numbers! They were only invented for punch cards so that if somebody shuffled them you could sort them out again. This is the 1980s, and we don't use punch cards on the C64, so why do we have to suffer line numbers? Mickey Mouse award for the Thirteenth Century goes to the inventor of line numbers, especially in BASIC programs.

Thursday 12th February
Wrote (on paper) and keyed in an object block plotter, which plots blocks of characters on screen with which I shall build a sort of mini-dreadnought. This is the ship that the player will fly. At biggest it could be twenty-seven characters wide by n-n-n-nineteen high! Should give the player a feeling of superiority. I've got the 'Player X' words and the game logo put on screen using this block plotter, as well. Looking at it now I reckon I should put the moving grids up using the same plotter, which will save on code. However it can only handle the first 128 characters, because I use the top bit as an end-of-column flag. The grid uses higher character codes.

Took some screen shots this week, but the processor always takes at least a week to do black and white so I'll send them to ZZAP! next month.

Since our printer has been disconnected for taking up too much room on the desk, this diary will have to go home for printing.

To be continued . . .

```
6510 ACCELERATOR LISTING
10 B=0
20 FOR X=49152 TO 49463
30 READ C
40 B=B+C
50 POKE X,C
60 NEXT X
70 IF B<>44878 THEN PRINT "ERROR": END
80 SYS 49152
90 PRINT "<CD> <CD> <CD> NEW <CU> <CU> <CU>"
100 END
200 DATA 174, 48, 3, 172, 49, 3, 142, 46
210 DATA 3, 140, 47, 3, 162, 74, 160, 192
220 DATA 142, 48, 3, 140, 49, 3, 162, 245
230 DATA 160, 192, 32, 188, 192, 162, 6, 160
240 DATA 193, 32, 188, 192, 169, 128, 141, 225
250 DATA 192, 160, 3, 202, 208, 253, 136, 208
260 DATA 250, 206, 32, 208, 173, , 225, 192, 56
270 DATA 233, 1, 72, 238, 32, 208, 104, 141
280 DATA 225, 192, 208, 229, 169, 37, 141, 5
290 DATA 220, 96, 201, 0, 208, 8, 165, 186
300 DATA 201, 1, 240, 5, 169, 0, 108, 46
310 DATA 3, 32, 23, 248, 162, 26, 160, 193
320 DATA 32, 188, 192, 169, 11, 141, 17, 208
330 DATA 169, 48, 141, 225, 192, 160, 0, 202
340 DATA 208, 253, 136, 208, 250, 206, 225, 192
350 DATA 208, 245, 169, 34, 133, 192, 169, 55
360 DATA 133, 1, 169, 27, 141, 17, 208, 173
370 DATA 134, 2, 41, 15, 168, 172, 134, 2
380 DATA 185, 36, 193, 141, 243, 192, 162, 226
390 DATA 160, 192, 32, 188, 192, 165, 197, 201
400 DATA 64, 240, 250, 174, 46, 3, 172, 47
410 DATA 3, 142, 48, 3, 140, 49, 3, 169
420 DATA 27, 141, 17, 208, 169, 64, 141, 5
430 DATA 220, 76, 116, 164, 142, 203, 192, 140
440 DATA 204, 192, 160, 0, 140, 225, 192, 172
450 DATA 225, 192, 185, 226, 192, 73, 255, 72
460 DATA 238, 225, 192, 240, 11, 32, 210, 255
470 DATA 104, 201, 13, 240, 3, 76, 199, 192
480 DATA 96, 19, 185, 176, 170, 177, 187, 223
490 DATA 250, 190, 175, 173, 182, 179, 223, 185
500 DATA 176, 176, 179, 101, 242, 201, 202, 206
510 DATA 207, 223, 190, 188, 188, 186, 179, 186
520 DATA 173, 190, 171, 176, 173, 242, 189, 166
530 DATA 223, 190, 177, 187, 173, 186, 168, 223
540 DATA 189, 173, 190, 166, 189, 173, 176, 176
550 DATA 180, 242, 172, 186, 190, 173, 188, 183
560 DATA 182, 177, 184, 242, 111, 250, 227, 96
570 DATA 99, 225, 224, 97, 126, 106, 105, 104
580 DATA 103, 102, 101, 100, 0, 0, 0, 0
```

MENTAL PROCREATION

Part Three

By Andrew Braybrook

Between eye operations and Trans-Atlantic jaunts to oversee the adoption of his third child, Father-To-Be, Andrew Braybrook, misses his ante-natal classes, and throws up ideas in the early hours of the morning . . .

"It's purely for medicinal purposes, don't you know . . . "' Down at the Ante-Natal Clinic, Andrew is dressed to impress and make new friends (don't worry kids – it's only coloured water) . . .

Friday 13th February

Yet another "Why don't you . . . " from Mr Liddon, wanting a rotating starfield rather than a vector one. This could make the control mode rather more interesting, giving speed and rotation instead of X and Y speeds. Altered the starfield coding accordingly and instead of the stars rotating around the ship, they veered away from it. This looked very peculiar, like anti-gravity shields. Aha! Reverse the vectors and try again. Same thing happens. How about reversing only one vector? That's better, it doesn't even matter which one, except that the direction of rotation changes. Unfortunately the calculation isn't quite accurate, as the stars slowly spiral outwards and collect in the top left corner. More accurate

" . . . I don't recall ever being taught the application of controlling moving starfields!"

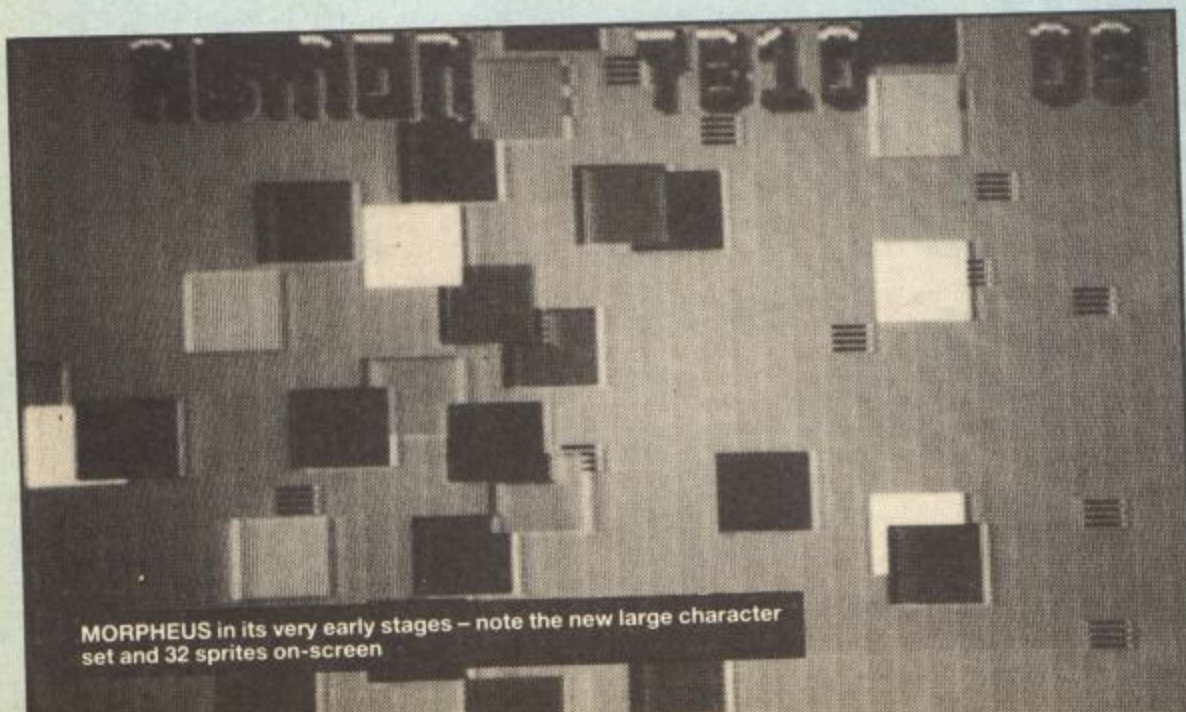
calculations are required. Nevertheless it proves that ST and I still remember a bit of basic maths, although I don't recall ever being taught the application of controlling moving starfields!

Monday 16th February

Increased the accuracy of the star rotation system to a 65536th of a pixel which holds things steadier, although the lack of subtlety involved in the calculation method still causes some inaccuracy. I left it running and leaving trails over lunch, and returned to a screen full of dotty circles, the stars had again crept outwards. As long as the game itself tends to discourage just sitting still and spinning then I see no problem. I suppose now that you know that you'll all try it.

To stop things moving in an oval orbit, (I'd used slightly incompatible co-ordinate systems for X and Y), I reduced the resolution of star movement vertically onto alternate pixels only. This has the beneficial side-effect that it frees 48 characters that were reserved for those star positions, and allows me to run the full quota of stars while docking. Previously the moving grid system used 32 of the star characters.

I'm also bursting to tell everyone that I've got an Amiga at last, and they're wonderful! Got it second-hand from Mr Liddon (would you buy a used computer from this man?) with an instant software collection, which is the only way to get one until the A500 comes out. I've been using the Deluxe Paint utility on it to do screen mock-ups of *Morpheus*. It's very useful for experimenting with graphics designs.



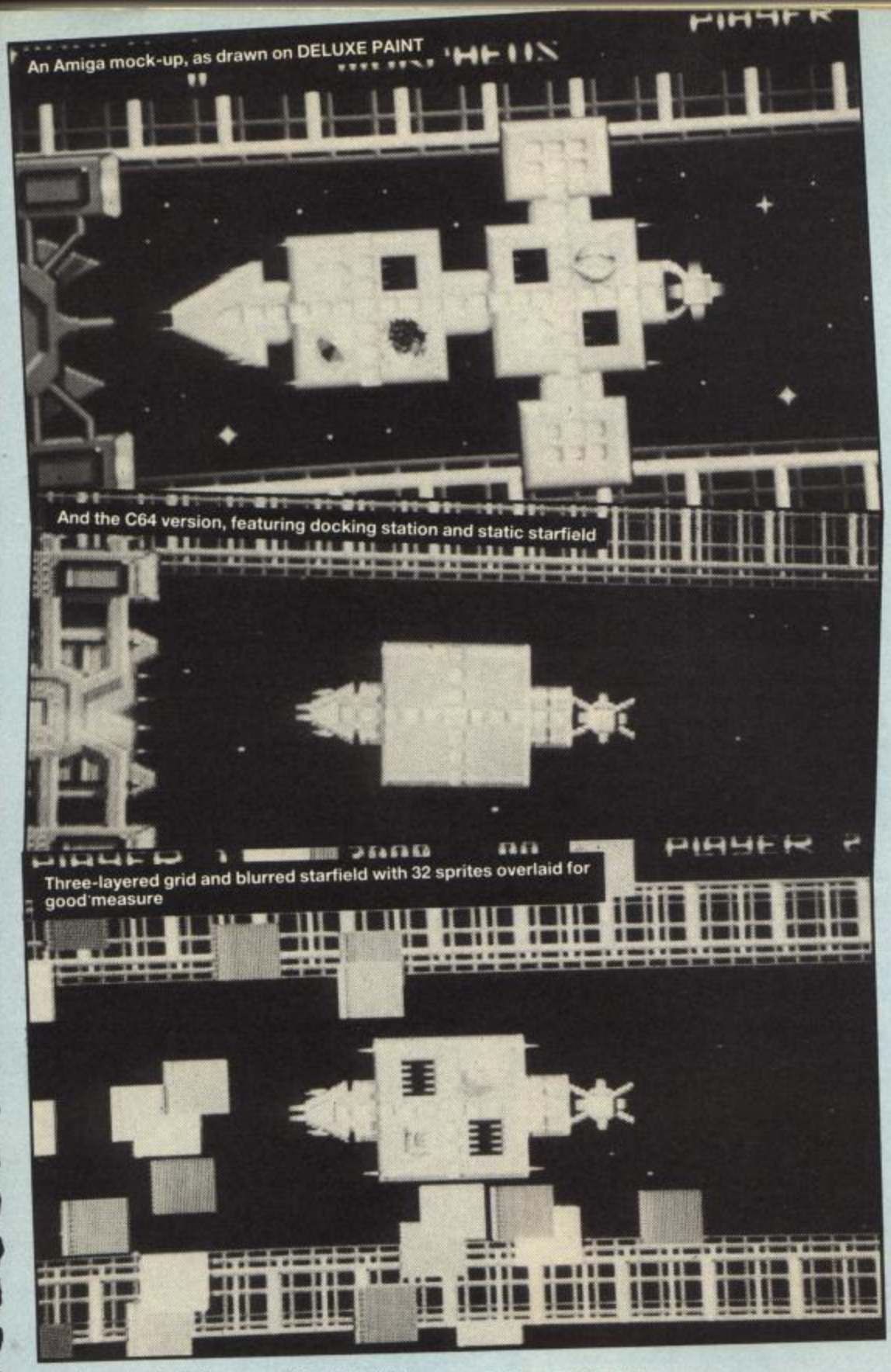
MORPHEUS in its very early stages – note the new large character set and 32 sprites on-screen

Tuesday 17th February

Off to hospital for a quick eye operation – normal service will be resumed as soon as I can see again!

Thursday 26th February

Back after a week of convalescing with an eye that feels as if Frank Bruno hit it, and being unable to sit in front of a TV for long, let alone a computer screen. I've been thinking about the game scenario and have nearly sorted out the gameplay – but not quite. So, I decided to design some sprites instead! For a docking-type sequence I want to suggest a giant platform at the side of the



screen by only displaying some of it. This will be built of sprites to give me another movement speed and colour set. I have already designed something similar on the Amiga, so I started drawing the sprites on the sprite editor. It's quite difficult to check that they knit together so I also wrote a small BASIC program to display all the sprites together as they will be. This worked fine until I needed more than eight sprites. Whilst *Morpheus* is capable of displaying more sprites, my sprite editor isn't – and considering the trouble CBM BASIC has to even display one sprite, I didn't hold out much hope for writing a sprite multiplexor in BASIC. I designed the sprites as best I could and then put them straight into the game. By directly accessing the sprite display positions while the game was running, I could manoeuvre the blocks into their correct positions. I used 19 sprites in all for this. I didn't make too many graphical errors, although one pixel took a while to correct – it turned out to be a speck of dust on the screen.

" . . . one pixel took a while to correct – it turned out to be a speck of dust on the screen . . . "

This docking station can now be a different colour scheme from the moving grids, and will slide onto the side of the screen independently.

Friday 27th February

Gone to Peterborough to try to get a passport in a hurry – that sounds a bit suspicious, doesn't it? I've been invited to Chicago to help start the *Uridium* conversion to the PC, Apple, ST and Amiga. Who am I to argue? Pity though, I wanted to do the Amiga version myself, but at least I'll be able to help out, and ensure that a first class mega-wonderful job is done.

Scribbled further notes on the scenario whilst on the train. By just following through various arguments some theories show up as glaringly impossible, whilst new ideas are revealed which can be handled by existing theories. What I need is some way of showing two different factions of meanies. Colour is normally used but I don't think that will be possible as my sprite system is speeded up by the fact that any one sprite frame will always be the same colour. It also annoys the players with black and white TVs which I'd rather not do.

I'm also again wondering about a two player option where the players play simultaneously. This invariably eats more CPU time and complicates matters because I have to effectively design two games, for the single or dual player options. Both games must be similar and equally as good.

Monday 2nd March

Morpheus unfortunately has to take a back seat again as I have to prepare some disks for taking to America – assembler source files, graphics data, development versions that allows perusal of levels – the usual stuff. It would be rather embarrassing to arrive with only half the material – you can't just nip back for anything you've forgotten!

Tuesday 3rd March

Began writing a thesis on *Uridium*, going through it all, making notes on how it all works. Now it may look very simple to you out there, but half the reason for that is that the program really does go out of its way to be friendly and do whatever you want. I figure that the moment the player gets irritated by the program not appearing totally transparent then the magic is lost. How many times have you been thumping fire to play another game when all the program will do is play a stupid jingle or show you the high score table? Never in *Uridium*. The game must appear real, and the moment manky programming rears its ugly head, the game is destroyed. Not many games are up to this standard. The player should never by limited by the programmer's inability.

Wednesday 4th March

Off to the American Embassy to get a Visa, apparently my Barclaycard won't do nicely enough!

Thursday 5th March

Back to the thesis on *Uridium*. I never knew there was so much in it. I really get involved in the current game that I'm working on, to the exclusion of everything I've written before, so looking back at the code is really weird, almost like it was written by someone else. It takes me ages to fathom out what it's doing in places, the control mode is really complex to get it really working smoothly. It may appear simple to some reviewers but I assure you that appearances are deceptive. I have to remember the sequence in which I developed various features to work out why it's doing some things, as I'm bound to get asked questions, and there's nothing worse than not being able to explain your own work.

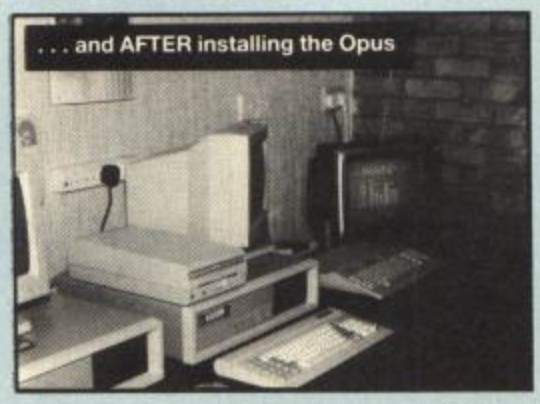
Friday 6th March

Still on the thesis. I'll have to finish it over the weekend.

Monday 9th – Tuesday 17th March

Off to Chicago. I'll have to get down to some real graft to make up the time, otherwise I'll be overtaken by the unscrupulous programmers who are already cloning *Morpheus*!

To Be Continued . . .



MENTAL PROCREATION

By Andrew Braybrook

After four months of hard labour (ouch), ante-natal depression has reared its ugly head – and a huffy Andrew Braybrook relates his experiences with soft and hardware problems, piracy, and computer magazines' reviews . . .

Wednesday 18th March

Got back from Chicago yesterday (place dropper) so it's back to *Morpheus* today. To compact my sprites and still be able to use them I have to write a de-compact sprite routine for the game, create the sprites I want and compact them in the first place. The de-compaction was easy, it just converts the data for one sprite into its real image. Since the images will be small, they will be a small clump of data with zeroes before and after it. I shall convert the leading zeroes to a one-byte count, and specify the size of the central data clump. This should convert each 64 byte sprite down to a more-manageable 15 or so – quite a saving.

Confession: I don't actually have *any* sprites for this system drawn up yet, I've not been able to draw anything that I regard as suitable. To get the data compacted I decided to write a BASIC program. It is easier to stop and can report errors in a more friendly fashion than just setting the border colour. Of course it will be a lot slower, but I won't be running it too often – I can survive on test data for a long time.

I've been thinking about the sprite combining system and have decided that the X and Y reflections are unsuitable, as images will be drawn with light coming from one side so any reflection will cause an incorrect image. I will have to produce any reflected images myself.

The materialisation sequence is also causing some concern, but with some fancy raster splitting I may be able to draw up a second character set with a see-through ship and combine it with the current ship, then fade it out by converting the data to run through the grey shades.

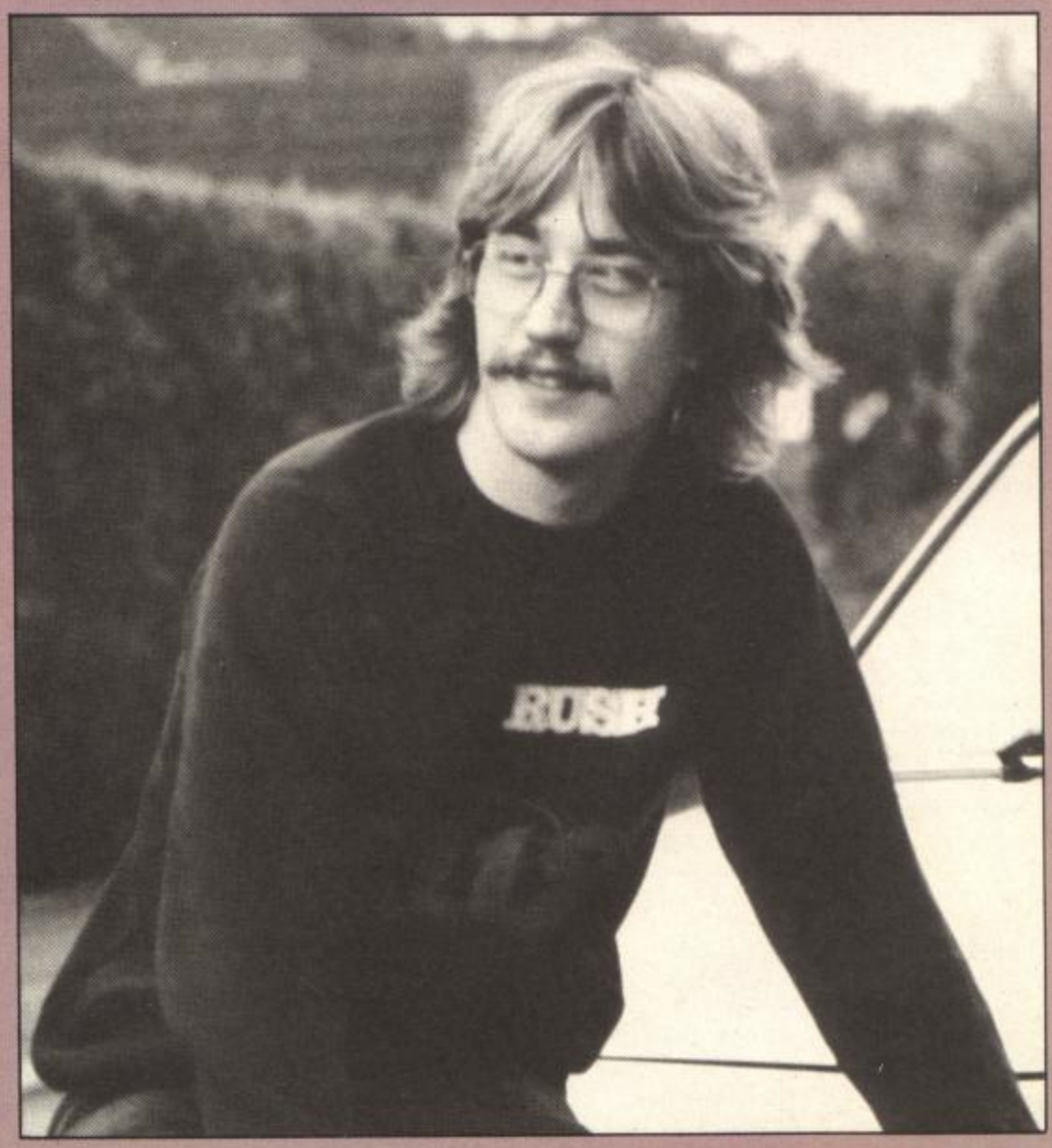
Thursday 19th March

This dematerialisation thing is all round the wrong way. I've been looking at it from a camera's point of view. The film director would watch the ship disappear from the docking bay, fade the picture, fade back in on another part of space, then rematerialise the ship. However I think I should be looking at this from the ship's viewpoint, or at least a remote camera associated with the ship. Thus the grid and stars would fade into nothing, leaving the ship, then the new stars would fade back in. My big problem has been separating the ship from the grid, I can't just split the screen since the ship can overlay the grid by up to two characters, and both are built from the same character set. I've sort of boxed myself into a corner by attempting to extend what I'm achieving on the C64. I do still have all the sprites free so I may be able to second them for some evil purpose.

I think the reason why there has been a spate of vertical or horizontal scrolling games is that those particular formats are best when using most of the C64s capabilities. It's nice to extend the playing arena for the game, and the C64 hardware was designed to help this. In *Morpheus* I'm attempting not to use the smooth scrolling facilities which make up a large part of that distinctive Commodore look. I therefore have to push further in other departments to end up with a satisfying product, like the gameplay.

I'm not having much luck designing any suitable sprites. I had a quick thrash on the sprite editor but didn't produce anything particularly

All in a Rush (double ouch) . . . a pensive moment, in which Andrew ponders on the meaning of life, the universe, and Hewson's deadlines . . .



inspiring. Sometimes it gets like that.

Friday 20th March
Day off.

Monday 23rd March
Tried to get some sprites drawn on the sprite editor. I's difficult to visualise the game when you don't know what half the graphics look like. I remember *Paradroid* went through a stage like this. I still want to avoid the same old spaceship designs, and with a sixteen-frame animation system I ought to be able to come up with something . . .

After a few hours I gave up, having produced nothing of any use to man or beast. I brought in the Amiga today so I decided to fire up Deluxe Paint and try to draw some sprites with that. I enhanced my *Morpheus* mock-up picture for inspiration, but no sprites were forthcoming.

Tuesday 24th March
Went to Hewson's to discuss life, the Universe and everything. Apparently I have to finish *Morpheus* this year!

"ST was trying to communicate with extra-terrestrials by waving an RS-232 lead in a skywards direction, which was causing pretty patterns to appear on the Spectrum . . . "

Wednesday 25th March
A day of judgement. I think I may have an overall sprite style. I need to show which of two factions each sprite belongs to, its strength, and they have to be able to move in any direction without looking as if they're going backwards. The game scenario is also evolving. In the beginning there was a pretty star in the middle of nowhere. A small reaction in the heart of the star produced two split particles which split from the star in opposite directions. These each split into two and each one split again. These particles then begin to drift towards their respective partners, and when they meet, fireworks. Your mission Jim, should you choose to accept it, is to bring the particles together more gently by capturing particles from one charge centre and deploying them at the other. Two particles of equal but opposite charge will cancel each other out, but two like particles will combine and become stronger. I also hope to introduce neutral charges and photons to upset the situation.

On the coding front I'm not happy with the star rotation, it's slow and heavy on CPU usage. There's enough work to do running 32 moving objects, so the star rotation is out. I've coded the grid fade-out routine which took all of 15 minutes to write as an existing routine was easily modified to carry out this task. Now I can fade out the multi-layered grid slowly or quickly, and while it is moving to choreograph the docking sequence using existing routines.

ST meanwhile, having written Amstrad *Ranarama* in two weeks flat, was trying to communicate with extra-terrestrials by waving an RS-232 lead in a skywards direction, which was causing pretty patterns to appear on the Spectrum. Why bother to build an interplanetary craft when you can use RS-232 . . . Aaarrgh!

Thursday 26th March
I've actually managed to rustle up a few sprites – hang out the flags! I needed an animation sequence to show two objects cancelling each other out, so I drew an implosion sequence which involves a circle of dots spiralling inwards, followed by a twinkle. This takes fourteen frames of animation in total. I also enhanced the docking bay sprites slightly and put them all together ready for inclusion in the game. I'm getting a better idea of memory usage now, so I can so I can put various chunks of data in their final places.

I then set up the colour table for the sprites that I have drawn, each sprite image always being the same colour. I can modify this table 'in-flight' to cause glowing effects and such-like, so it's not too restricting, and saves me a lot of time not having to worry about getting the colours right during the game.

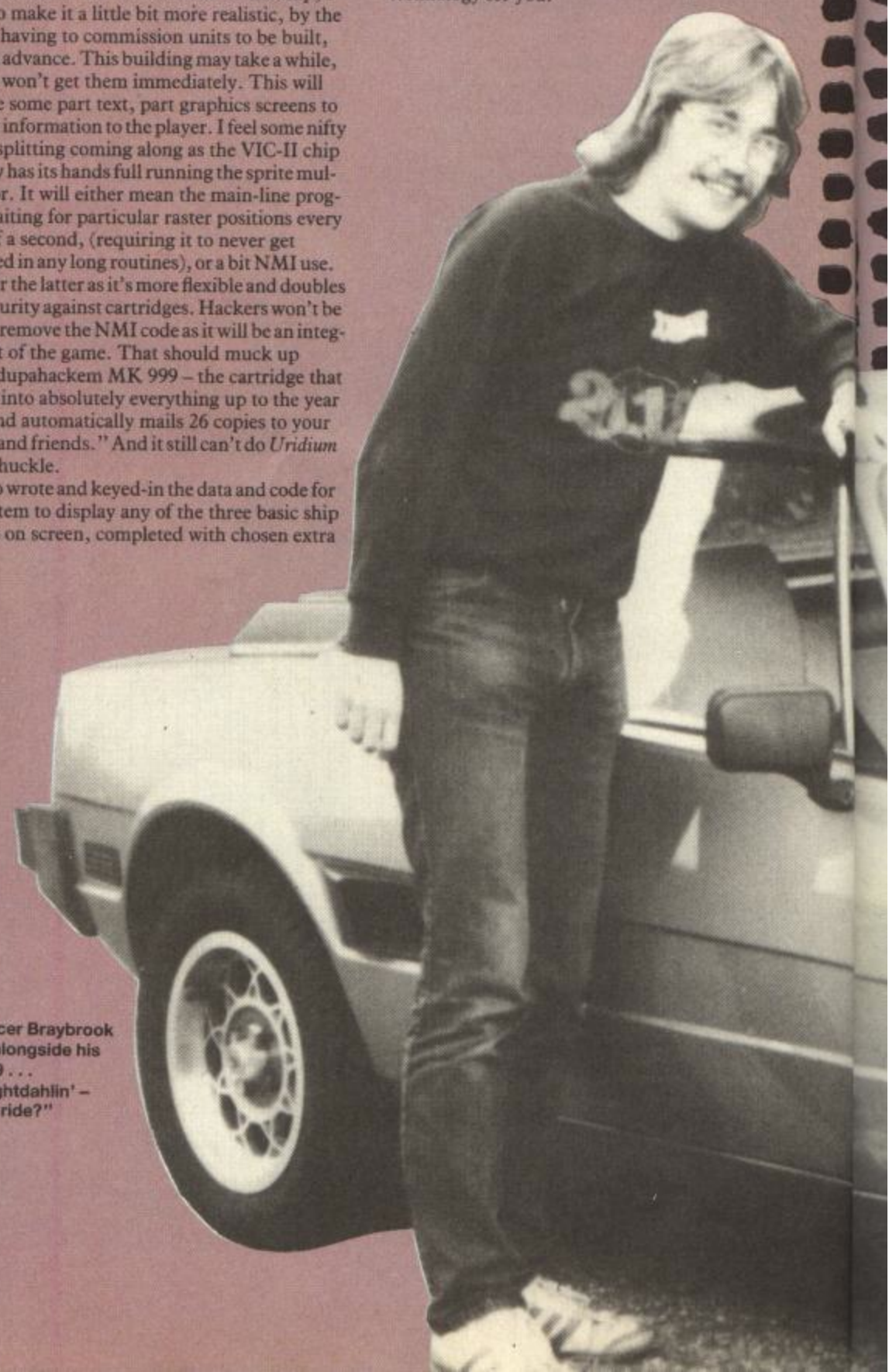
We're having minor equipment troubles, the 1570 needs a dab of glue as it's getting a bit tetchy, the 1541 won't read anything, ST has blown up the C64 and the 1541 analyser disk has a read error on it! I'm also still trying to soak out the coffee from my Amiga keyboard which causes some keys to stick, so it's not been our day. ST is keying in the sound routine on the PC and will be making further enhancements to it, probably in a mixture of 6502 and Z80!

Friday 27th March
Started to sort out the game structure. The game will involve frequent returns to the docking bay for information updates and ship repairs. Rather than just obtain new units to bolt to the ship, I want to make it a little bit more realistic, by the player having to commission units to be built, cash in advance. This building may take a while, so you won't get them immediately. This will require some part text, part graphics screens to supply information to the player. I feel some nifty raster splitting coming along as the VIC-II chip already has its hands full running the sprite multiplexor. It will either mean the main-line program waiting for particular raster positions every 50th of a second, (requiring it to never get involved in any long routines), or a bit NMI use. I favour the latter as it's more flexible and doubles as a security against cartridges. Hackers won't be able to remove the NMI code as it will be an integral part of the game. That should muck up "Supadupahackem MK 999 – the cartridge that breaks into absolutely everything up to the year 1997 and automatically mails 26 copies to your family and friends." And it still can't do *Uridium Plus*, chuckle.

I also wrote and keyed-in the data and code for the system to display any of the three basic ship layouts on screen, completed with chosen extra weapons and systems. Now that I can use local labels with the cross-assembler I don't have to wrack my brain coming up with six-letter unique names like RFLOP3, I can just put . . . LOOP everywhere. I bet I'll regret it when I try to understand this code in a year's time.

"My Amiga appears to be decaffeinated too!"

Monday 30th March
Had a long duel with the cross-assembler today, cross being the operative word. I wanted to do the equivalent of a GO TO DEPENDING ON using an assembler macro, which is basically a way of inventing your own assembler commands. The assembler however insisted on producing 27 error messages, all for one line, where I had to use this new command called JUMPY. No clues were given, just a bundle of error messages like "Labels not allowed here." Where, you stubborn goat of an assembler? Give me a clue. Finally we decided that the assembler has a bug in it as it doesn't evaluate parameters properly in a FOR-NEXT loop. Thus I had to key in the same thing nine times, twice. There's modern technology for you!



Boy Racer Braybrook poses alongside his Fiat X19 . . . "Aw Irightdahlin' – fancy a ride?"

Some technical problems resolved, the 1570 only won't read disks when the PC is switched on. It's getting zapped by magnetic radiation. The solution? Always switch off the PC when loading from disk. Probably not, move the 1570 to pastures new. This means that I can screw the top back on it, as currently I have the cabriolet model 1570 Ghia for easy maintenance!

ST now thinks that the C64 that we sent for repair isn't blown up after all, it wouldn't load because the C2N had thrown a wobbly. That's now all crated up ready for repair. Wonder if they'll find anything wrong with the C64? My Amiga appears to be de-caffeinated too!

As far as *Morpheus* is concerned I roughly know what the control mode is supposed to do, but I'm giving the player more freedom so it's going to be a little more cumbersome to learn fully. I shall try to introduce the new features during the course of the game rather than have them all available at the start. The game has to be instantly accessible.

Tuesday 31st March

Tested the new routines to display the three configurations of ship. I only noticed after a couple of tests that no ship at all had appeared. This was simply because I hadn't told it which of the three ships I wanted. I was lucky that it didn't crash as it had been reading random data, but unfortunately my routines tend to have plenty of 'exit on error' clauses to prevent such catastrophes. After correcting a couple of blocks with the wrong numbers in, I am now the proud owner of a rounded-metal ship construction kit.

ST and I had a discussion on whether I should set up the ship with the front facing left or right, I wanted it to be left as it would be different from most other games. Had you noticed that more games play facing right? For example: *Scramble*, *Nemesis*. I can't think of any that only play to the left. This is partly because we read things from left to right, our brains are more accustomed to the movement. If I set my default direction the other way round it may subconsciously jar with the brain, putting people off the game. So, even though my ship will move in all directions (not all at once, incidentally, unless you overheat the engines) I shall configure it to face to the right. This means that two of my thus far created eleven blocks are already redundant as they face left only, I guess some prototypes never make it. It does free up some more graphics space though.

"I hope I never get stuck in a lift with any miserable types who didn't find it amusing!"

Wednesday 1st April

I've got three different sized ships on screen and this morning I thought of another configuration, so that makes four, which is a nice number for a programmer to deal with. The system that ignores or displays extra weapons pods on the ship is also working. This system can also display the landing pad for the remote ship to be launched from.

The game can now display the docking bay of its own free will. I no longer have to set its variables up using the monitor. It's all starting to fit together. I happened to notice that not many stars were being displayed in the bottom half of the screen. Upon checking I discovered a remnant from the rotating starfield days. A quick cut and thrust of the delete key and it's gone.

I also pottered about with the character graphics to design some new pieces for the ship, and keyed in polar to X-Y vector conversion table, 512 bytes of juicy hex to allow eight speeds in 32 directions. I should now be able to generate better circular movements.

I gather that some people have tried my little 6510 accelerator. Okay so it doesn't quite speed up the C64 to mega speeds but you have to laugh. It seems to have taken anything from fifteen minutes to two hours to key in depending on manual dexterity. Lucky I didn't pad it out too much, eh?

It's interesting to study people's reaction to the hoax, one or two people were apparently very grumpy indeed, hurling abuse in all directions. I reckon that just shows the Human animal for what it is. Greed had set in at the thought of gaining extra CPU wellie for nothing and then it was snatched away. I hope I never get stuck in a lift with any miserable types who didn't find it amusing!

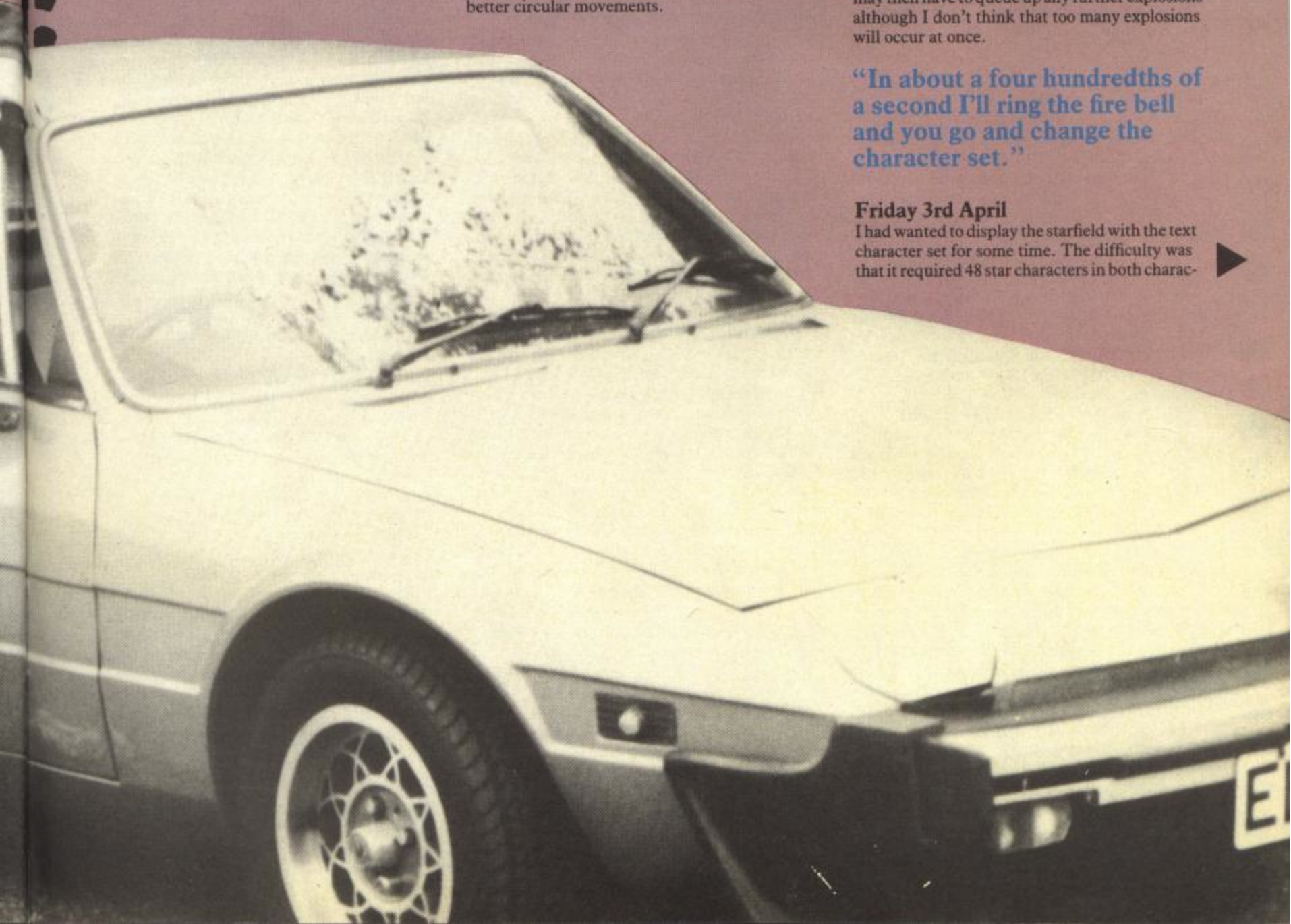
Thursday 2nd April

Designed some graphics for what I'll call the system blocks. Each ship configuration can carry a number of these, two on the smallest ship, up to eight on the largest. These system blocks will be for additional 'passive' systems on the ship, no weapons, but additional energy, charge holders, tractor beams and the like. These will be prime targets for the enemy and may be destroyed. Up to 32 system blocks are being budgeted for, each with its own colour scheme. The energy display block consists of eight frames of animation and its speed of rotation will show the ships energy. I will need a system to plot these blocks into the character set. Only eight blocks are needed on the biggest ship at four character a piece, so I have reserved 32 (that magic number again) characters for these. I couldn't afford the space for 32 different blocks in the character set as there are only 256 available, so plotting these on screen will involve updating the character set in-flight. I don't want to push the timing so I'll have to devise a routine to plot explosion frames only on cycles where the energy display isn't being plotted. I may then have to queue up any further explosions although I don't think that too many explosions will occur at once.

"In about a four hundredths of a second I'll ring the fire bell and you go and change the character set."

Friday 3rd April

I had wanted to display the starfield with the text character set for some time. The difficulty was that it required 48 star characters in both charac-



ter sets. Having accepted this as a necessary overhead I wanted to use this fact to combine text and graphics on screen simultaneously. Not just a simple split screen but more integrated.

This involves raster splitting and created the dilemma mentioned earlier (27th March). I can't use full blown raster splitting for screen changes as the VIC chip is already using these. Thus I must try using the NMI time which is like an alarm clock that tips the bed over, ie: you *do* wake up immediately. I couldn't quite see how to use only one set of interrupt routines to run either a straight graphics screen or a mixed screen. I then worked out how much extra coding is required to split the screen, and it's quite minimal. So one set of interrupts runs all the time, but in graphics-only mode the splitting just gets the same character set instead of a different one. The game has to function correctly in either mode anyway so it's no real overhead running a small quantity of unnecessary code all the time. It's a case of by the time you've worked out which function to carry out, you could have done them both anyway.

Of course the structured approach would be to just duplicate one and a half Kilobytes of code to create two interrupt systems and change one instruction, however, on a limited memory system this is not particularly practical. Anyway I implemented the NMI timer system which takes its cue form the top raster interrupt that sets the first sprites. This starts a timer which basically says:

"In about a four hundredths of a second I'll ring the fire bell and you go and change the character set."

This routine then sets the timer for the next split, four times in all down the screen. This also has the beneficial side-effect that it screws up ALL cartridges, including the ones that claim to the contrary (they just don't test them on *AlleyKat*, *Uridium Plus*, *Terra Cresta*, etc).

"Sorry couldn't find them in the shop." Well, they were looking in the local greengrocers! Very exhaustive test I must say. My name's Ben Elton, goodnight!

Monday 6th April

I found it quite disturbing to read one of Mr Mangram's replies in the May ZZAP! Rrap regarding reviews, which posed:

"Which would you prefer, an earlier black and white review . . . or a full colour review a whole month later?" This has been echoed by Commodore User recently claiming to be first with this review and first with that review.

Well, since I'm on the receiving end of the review I figure that having spent six months producing a game, it deserves an accurate review, not to be treated like another lump of nothing on a production line. How can it possibly get an accurate review if the reviewers are simply trying to be first to get to print?

The game should be played on and off for a couple of weeks, how else can you test lastability? Surely you'd rather read an accurate appraisal of a game, so what if it is a month later than a bad one. If you're like me you wait for *all* the reviews anyway, so whoever got the review out first will be forgotten!

Also, since the magazines get early review copies of the game then the review could well appear before the game is available anyway. That just frustrates you, upsets the shops and wastes everybody's time. Applying this to *Morpheus*, I hope to finish it by the end of June, but it is unlikely to be released before September as the packaging and advertising will then go into full swing.

I think it's ultimately up to you, the games players, to write to the magazines, especially if the standard of review is slipping. Tell them you want accuracy of facts with plenty of detail. Colour is obviously preferable as this is how most of you will see the game, and you don't just want regurgitated instructions, anybody can do that, you want an objective review of what the game is like to play, what new original features the game contains. Am I right or am I right?

I put together most of the undocking sequence today. The ship now accelerates to the right, the stars and grids moving in the opposite direction. The grid then fades out and the stars change. The stars changing gives me a sneaky chance to build the required sprites over the old grid data. The only thing not yet in is the docking bay relative movement, it should gracefully scroll off, but at present it just sits there. This will all be tied-in to the sprite movement system, so I'll leave that for now.

It's a long time since I've had the grid off the screen. It needs the full screen size to give the 3D effect more impact, and it gives me a better idea of the playing area size. With a large ship with all its extensions it's nearly the full screen height.

Tuesday 7th April

I've rooted out a major stumbling block in the theory of the gameplay. I wanted two particles or charge centres to split from a star. These would be of opposite polarity, I then wanted them to split again into two particles of opposite polarity, and finally split again. This raised the problem that an already negative particle can't really split into a negative and a positive one. It would have to split in another 'dimension', not necessarily a physical one, but a mathematical one, for example: matter/anti-matter, positive/negative clockwise spin/anti-clockwise spin, red/blue, whatever, provided the two opposites cancel each other out, the last example simply being an arbitrary rule, which is all that a game really is, a collection of arbitrary rules. Whether you match these to a real sport, or a simulation of some physical process or something totally abstract is entirely up to the programmer. I think that may define the originality of a game or a clone, whereby a clone game has attempted to copy the arbitrary rules of another, it's just worse if they nick the graphics and sound as well! But back to splitting particles. I can't really justify multiple splitting, it just complicates the situation. I've resolved this by just having one split from a central star, although into more than two parts.

In experimenting with the polar vectors I've begun to compose the title screen sequence, using routines that will be used in the game too. I now have a many-coloured central star which can randomly spit out particles, in circles or in spirals. This also allowed me to check the polar vector table that I keyed in recently. I told the particles to stop moving when they left the screen, sensible enough I thought, I'd rather they didn't float about through memory. Instead the ones that left off the top bounced back and rained down the screen. This gave rise to the programmer's war-cry of: "That can't possibly happen!" Quite often bugs cause things to occur that haven't been programmed in yet, or things occur that you wouldn't be able to program in a month of Sundays, if you tried. This one turned out to be miskeyed data in the polar vector table on the one number out of 512 that I happened to set the particles to as they left the screen to stop them. Glad I found it at this stage rather than when the data will be used for sprite movements.

" . . . we decided that the only course of action was to operate . . . "

Wednesday 8th April

The Opus has been causing some concern as it has not been reading from 'Drive A' properly, which is a blow because it reads all of its boot-up instructions from it, so we can't run anything. Upon the advice of the suppliers we decided that the only course of action was to operate, just an exploratory to discover whether the drive or the disk controller was at fault. We swapped the leads from the two drives and it booted up from 'Drive B' so we swapped them back and it booted from 'Drive A', problem solved, it was probably a loose connection. Saves sending it back anyway.

Further organisation of my ship set-up data means that I might be able to actually interpret it in a useful manner. It's all very well being able to set up these different ships, but the game will need to reference this data later, so it needs various pointers into all the tables to be able to get at the right information quickly.

One thing that has puzzled me, is that all the sprites disappear in the changeover from the title screen to the main game. I wanted this to happen anyway, but I didn't tell it to do that. I realised that this is because the sprite multiplexor clears out the sprite table ready for the next position, which don't arrive if it's busy setting up the screen – so what am I to do in the pause mode? All the sprites will disappear. Any offers?

Thursday 9th April

Took the bull by the horns and began work on the control mode. From a programming point of view this is the most complex one that I've done yet. It is arguably the most important part of the program, as it is the interface between the player and the game. It must therefore be very carefully tuned up to allow the player as much freedom as possible, to make his or her own mistakes and not be able to blame the game afterwards!

It took a while to actually get the thing started as I'll be using the fire button to move from one section of the ship to another, but I was also using this to return to the monitor so every time I tried to move on the screen it returned to that. A quick alteration put that right, but the flashing blue square representing the current position refused to move. This was because it thought it was in two different places at the same time. Realising my error I fixed the initial start position routine that always locates the engine room. I began by moving at rocket speed. Moving one character position every cycle is too fast. I gave it a delay of three cycles in any square but controllable movement. Having rigged the hold down fire-button to disengage from a particular system and put in the movement, I needed the routine to re-engage into another system and identify it so that the proper things occur when using it. It's no good sitting in the landing bay being able to fire rockets. That's where I needed to access into the ship setup table.

Got that system up and running too, and implemented my first use of my JUMPY macro, the wonderful GOTO DEPENDING ON.

Friday 10th April

Got to thinking today that I may move the ship a few characters to the left so that the primary play area is on the right-hand side of the screen. This will have the effect that the ship will actually touch the docking bay structure, which can't be a bad thing. Maybe I'll try it. I'm also planning the title sequence and considering the possibilities of a demo mode, but I've never been to fond of them. It's quite hard to demonstrate a fairly complex control mode well.

ST has been finishing work on the new sound routine for the C64. He's upgrading it for more complex sounds and music. He did try to explain some of the new ways of producing sound effects to me but it's really hard to relate a bunch of numbers to a sound effect. You just have to experiment with it. I won't be putting the sound module in until the last few weeks, so maybe ST will send me on an IBM sound effects course in Florida!

To Be Continued . . .

MENTAL PROCREATION

By Andrew Braybrook

Monday 13th April
It's about time I had a proper pause mode in the game, so that I can return to the title screen, or quit to the monitor properly. All I have to do is round up the routines that ought to be left running in pause mode (colour updates for instance), and ensure that no routines involving object update are called. As mentioned last month the sprite display system (or multi-plexor) expects new sprite positions to have been calculated every cycle to ensure smooth movement. During pause mode everything should stay still (partly so that magazines can take screen shots), so the easiest way to achieve this is not to call the movement routines. Trouble is, the sprite system then receives no information regarding sprite positions and shuts down, so they all disappear – not good.

Tuesday 14th April
The interrupt-driven sprite system was slightly amended to tell it to use the previous sprite positions if no new ones arrive. I can therefore now pause the game or carry out mega-long calculations in the knowledge that all the sprites will remain on screen. They may not move a lot, but it's better than disappearing completely. Thus the pause mode is now implemented. Work has also continued on the control mode to allow the ship to be steered whilst operating its engines. The three parallax star-fields all move at different speeds and can be made to spin quite merrily. The ship's speed will vary depending on configuration and engine mounted, but I don't want any sluggish combinations. I hate games where the control mode prevents you from doing the job in hand.

I also enhanced the object handler to ignore non-existent objects rather than process them anyway. I used to have to hide them off-screen! I have created my first three classes of object:

1. dead objects.
2. fixed on-screen objects.
3. docking bay sprites that move away from the ship.

This completes the un-docking sequence so the main port slides off the screen as you leave it – it's better than having it following you around.

One design change now has the ship positioned on screen left to give a larger area in front of the tractor beam unit at the ship's right.

Thursday 16th April
Put in some of the text for the title screen and high score table. With the giant-sized text and only two lines of characters available I shall only be able to print one line of the Hall of Fame at a time. This does have the advantage that I don't have to fit the whole table on one screen. I will allow for ten entries for now, eight digits per score and three letters of initials. There's not room on one line for a full name.

In preparation for the title screen I had to define some more characters: the joysticks and player symbols and a double width space. I also put in the colour and monochrome symbols, but I just remembered that I won't be needing them because there won't be a colour option this time – it isn't necessary. Carefully planned painting and decorating will make the game work on either type of TV.

I also put in the tractor beam firing mechanism which spits out a sort of cloud, but I don't like the graphics. I may change the whole routine to run that if the graphics require it. It's no good just putting up with a routine because you sweated blood over it – if it doesn't work then it has to go.

Tuesday 21st April
Put in some more text for the title screen and a high-score decode table. I may need to put in a save to disk routine and it's best to keep the amount to be saved as small as possible – we all know how long the breeze block takes. As long as I can decode this data ready for screen printing fairly quickly I can put in ten entries quite happily, maybe even 20. I like full zero-suppression on scores, it gives a more professional look – better than hundreds of leading zeroes. After all, calculators stopped displaying leading zeroes soon after the Sinclair Cambridge, and that was back in the 14th Century!

I've also put in the redocking sequence except that the docking bay sprites that scrolled off so nicely arrive back on screen in a state of minor disarray. Well, all right – a complete shambles! This smells of an index to the individual parts getting corrupted somehow.

Wednesday 22nd April
Yesterday's corrupted docking bay was caused by what we technically refer to as 'raster overrun'. Put in layman's terms that means 'trying to do too much in too short a time'. I always wondered what would happen if I ran out of CPU time. This was caused by attempting to fade out the grids while running 21 sprites. I don't actually need to do this as the sprites are held off-screen anyway, so I just set them up after the fade-in instead of before.

Tried putting in larger front-layer stars, but they looked too square. Changed the tractor beam to be more of a *Defender*-type laser. This required a total rewrite of the firing routine but it looks much better. The process involved required first working out the routine, testing it and finally tuning in – that is, slowing down some animation to alternate game cycles so that it is more obvious what is happening. I considered making it a continuous beam for repeated presses but it loses its effect. I rigged the control mode to fire the tractor beam from either the unit itself or from the engines section, which removes the necessity to keep running across the ship all the time.

I also had to increase the time taken to initiate the dematerialisation, as it was triggering too easily by accident. There's nothing worse than dematerialising to another part of the universe just as you are nipping out for a cuppa. There are quite a number of functions allowed to the joystick while in the engine section, about the same as in *Paradroid* (move, transfer and fire).

Thursday 23rd April
Did some sprites for the photon weaponry. I think I'll use three different brightnesses or colours and a number of different sizes and sound effects for them. This will enhance the difference between the weapons system. The enemy will be using similar weaponry.

To implement the weapons I need to set them up on the ship, so I've been working out what sort of a system I need to select new lumps for the ship. I'll need an installation system, a 'commission parts to be built' system, and a 'scrap existing parts' system.

Monday 27th April
Started putting in the self-fired weaponry. There are three stages of initialisation to co-ordinate. First, the original ship layout set-up which decides what weapons are available, then the individual gun details have to be accessed when the weapon is activated, and finally the actual firing of the weapon to initiate a bullet. This caused some confusion as the chances of a game still working is inversely proportional to the number of instructions added. So I decided to forego the first stage of weapon set-up, as this can be rigged to work using the monitor.

Had a long duel with the assembler again when it decided that I wasn't allowed to modify some code at a Procedure Name. All subsequent labels were apparently not in the same places on the second pass. How does it know? And why is it looking anyway? ST and I decided that the assembler is suffering from bad programming. I put in a second label at the same place as the first and now it's happy. I reckon that will probably earn the assembler the Mickey Mouse award for April.

Tried out the weapons system. Bit of space station drifted out of the guns in a fairly random direction. Two different guns produced different effects . . . all wrong, but still . . . Decided that it's time to print off a few routines for reference. Going grey waiting for printout.

Tuesday 28th April
Fixed the bugs in the bullet routine. It's much easier when there's a listing in front of you. The sequential firing gun is a bit of a disappointment – it works so well on the title screen, releasing particles one or two cycles apart in a circular motion . . . but the simultaneous firing makes up for it. Four or eight bullets released in a ring for mucho devastation.

By dinner-time the fundamental flaw had reared its ugly head. The control mode is basically unwieldy and generally all-round difficult to use. A quick pint down at The Chequers soon sorted this out. The problem was that controlling a weapon system means leaving the engine section ('Och Cap'n'). Thus you're left as a sitting duck on the screen just hitting fire for all you're worth, with no means of movement, just relying on the meanies walking into your line of fire – not particularly exciting, I trust you'll agree.

The solution is to allow movement of the ship from any weapon system. This allows movement while not firing, and temporarily locks out movement while giving the enemy a pasting. The latter ensures that you don't fire a joystick-directed bullet and immediately hare after it. By locking out the movement I only mean that it won't allow alterations to the current direction, not that the ship will stop! This also frees up the functions on the

engine unit, which is now only used to dematerialise back to the docking bay. I can also remove the fire tractor beam from the engine room routine which was attempting to cover up the inadequacies of the control mode.

The assembler definitely gets the April Mickey Mouse award for producing buckets of error messages on lines of perfectly good code, just for missing out a simple directive on a preceding line which it could have assumed to be present if it had any idea about user-friendliness. Even the COBOL compilers on the IBM mainframe could cope with this one, and they just love producing error messages like: 'Full stop missing, assumed present', followed rapidly by: 'Superfluous full stop encountered, assumed not present!'

Remind me to write the next game in COBOL – it's wonderful!

Wednesday 29th April

I put in the system block display routine today. There can be up to eight system blocks on the ship's hull which carry out various functions and enhance the ships capabilities. There's a twirly energy display and a current charge indicator. This routine also handles system block explosion animation, and runs at one frame every four cycles. So, by doing the animation on cycles based on the block number, it won't have to run more than two explosions at once. It doesn't seem to eat much CPU time anyway. If I ran it on interrupts it wouldn't take any time at all because as every programmer knows, the way to save CPU time is to run everything on interrupt. This takes no time at all because interrupts are free. As soon as an IRQ is detected the CPU immediately switches to 20 GigaHertz mode. Only kidding . . . I'm just bored of people telling me that they 'run the scrolling on interrupts', or 'poll the joystick on interrupts' – it means nothing but it sounds impressive.

Back to the system blocks . . . they can have different colours, which requires colour RAM update, so the addresses of these have to be noted by a previously executed routine which set up the ship's data. Colour RAM and screen RAM locations are mathematically related, and the assembler is awfully good at subtraction so this is no problem.

I had to optimise the grid fade-out routine to make it run faster, the docking sequence was running out of time, as indicated by my coloured order stripes. By optimise I mean fiddle it to execute quicker, which usually involves doubling the amount of code or using self-modifying code or both! I wish to admit here and now that self-modifying code is disgusting and very difficult to debug, but a good assembler makes mistakes less likely. There's nothing like a good assembler, and my assembler is nothing like a good assembler!

Rounded off the day by amending the sprite editor. It has a really . . . what's the word . . . 'unusual' selection of default colours – white purple and cyan. It also uses the border to indicate which colour you're painting with. This is rather silly, as a bright border completely alters the on-screen colours. These problems were fairly easy to solve as they all use the VIC video registers, so I just had to look for where it set the border colour and tell it to use black. The editor also has an annoying habit of printing in black, regardless of background colour. On a black background this is no use to man or beast – hack!

Friday 1st May

Started putting in the weapon and system selection routine which includes total ship replacement, commissioning, installing and scrapping of weapons and systems. I will need a means of regulating real time so that units can't be commissioned quickly by cheating. The selection mechanism must cross-check various lists of parts available, with parts being built and spaces on the ship to fit all these goodies.

I got out the *Paradroid* listing to see how the console logging-on routines worked. *Morpheus* will be allowing updating of data, so it must be done carefully. Maybe it should just produce update records for the overnight batch to run?

I'll confine all input to joystick only, and without an on-screen pointer. Although I am more sympathetic to mice and icons these days, there's a time and place for everything. I wouldn't use them just because they're fashionable – they must be functional.

I've got it displaying the right headings so far, and for most of the categories it studies the appropriate data and reports whether there are systems or weapons available or present. It's taken me a long time to figure out how to clear the sprites from the previous screen. After all the trouble with them disappearing when there wasn't time to update them, now I can't get rid of them! Even brute force and ignorance didn't work the first time. Got rid of them in the end though – I'll not be outwitted by a couple of sprites.

Tuesday 5th May

Chief test pilot and critic Robert Orchard had his first look, the atmosphere was not entirely unlike a Del Monte advert. De Man from Del Monte, he say: 'control mode eez not working wery well.'

We also decided that the ship is too wide. Suggestions of allowing it to flip onto its side to fit through narrow gaps were severely dealt with. The weapons are too near the edge to see the full effect, and meanies approaching from the top do not give enough warning.

So today is shrink the ship day. I removed the connecting tunnel to the weapon stations and cut down their width. The only reason they were so big to start with was to allow me to separate the harsh white highlight along the top edge from the coloured weapon ports. In the event I made the weapons ports white as well so I can combine the two in the same graphics space. A full-width ship is now reduced to 13 characters high from 19. It looks a little more streamlined now. One of the effects was that the ship passed over the grid, which it doesn't do in its present form, as it doesn't reach. I moved the grids in slightly to restore that effect. This made the docking port sprites look overly large so I cut them down too. I had wanted to reduce the number of sprites on-screen anyway. The game has to run on an NTSC C64 as used in the good ole US of A. Their C64s run about 16% slower on raster-synchronised games because their TV sets have to spit out 60 frames per second as opposed to our 50. Offset by this is the fact that their C64s actually run slightly faster (not a lot of people know that). Thus everything I run in the game for the European version has to have 16% spare time at least. I can use that time for increasing sundry items like the number of background stars, anything that can be altered once the game has loaded in and had a chance to sniff around and adjust to its surroundings.

I've also been tuning up the control mode. It's now a maze of delays and decisions, all in an attempt to make it transparent to the player. I rigged the weapons to remember the last-fired direction to reduce joystick-bashing. It also helps to differentiate between the functions of firing or leaving the weapon. I sometimes wonder whether I should keep a badly tuned version around to emphasise the fact that the game has actually been tuned up properly.

I managed to wreck the starfield completely in an attempt to improve its randomness – all the stars went and hid off-screen! Further adjustments put all to rights, so now the stars are tied to the screen with no extra off-screen areas. Any star leaving on the left will arrive on the right, but at a different height. Similarly, stars are re-shuffled vertically. The only purpose of the starfield is to give an indication of speed and direction, which it does admirably. Having done this I can reduce the number of on-screen stars from 30 to 18 and it still looks like the original number.

ST and I have devised the ultimate cheat mode to incorporate. The most comprehensive POKE of all time. Just hit any key on the keyboard, the score will set to all nine and it will print 'GAME OVER'. Just what is the point of an infinite lives POKE? The game is supposed to be challenging entertainment, not a boring wander through the graphics. If you want graphics, go and watch a film. It's not winning that counts, it's how you play the game.

Wednesday 6th May

Adjusted the starfield again to produce more distant particles and less close ones. This should enhance the effect of depth. I couldn't really spot the difference, but at least I feel better knowing that it's doing it properly.

I've put in the hull selection system but left out the bit where I have to pay for it. This makes the testing easier. The actual gameplay mechanism just gets in the way of program debugging.

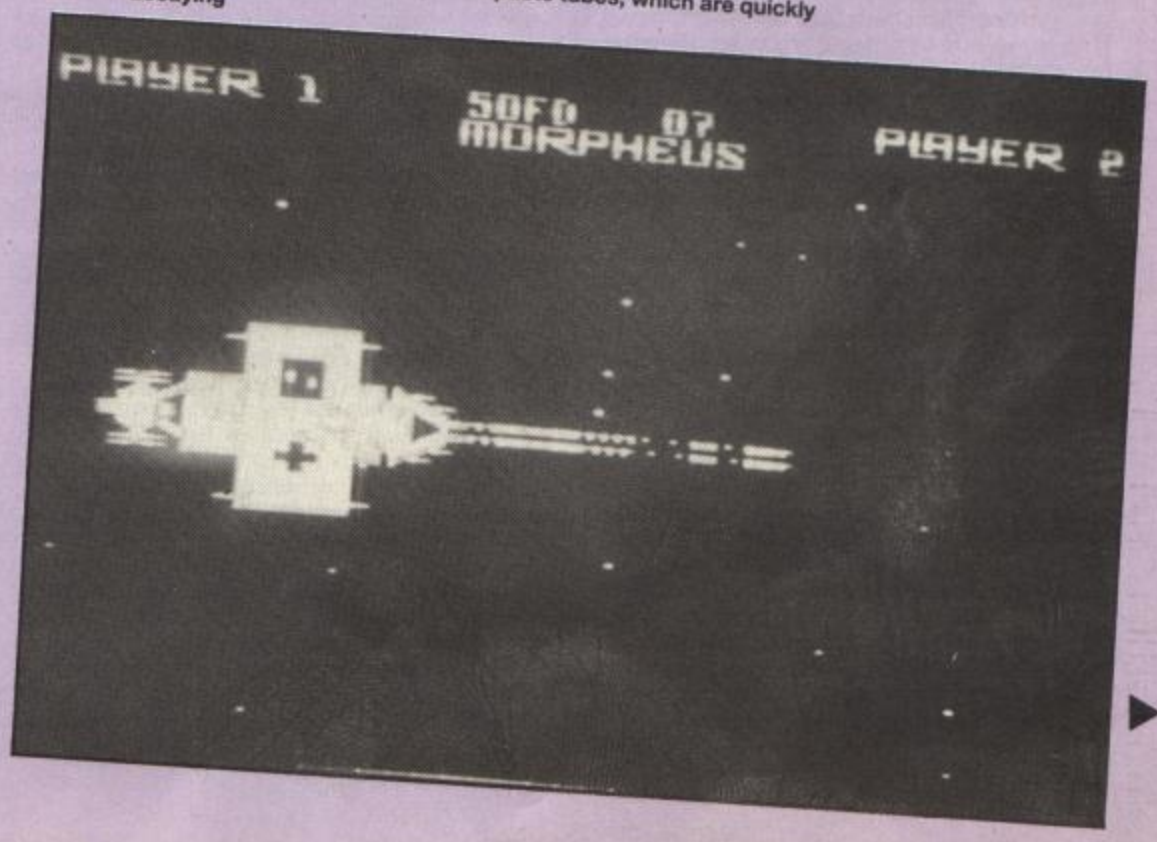
Thursday 7th May

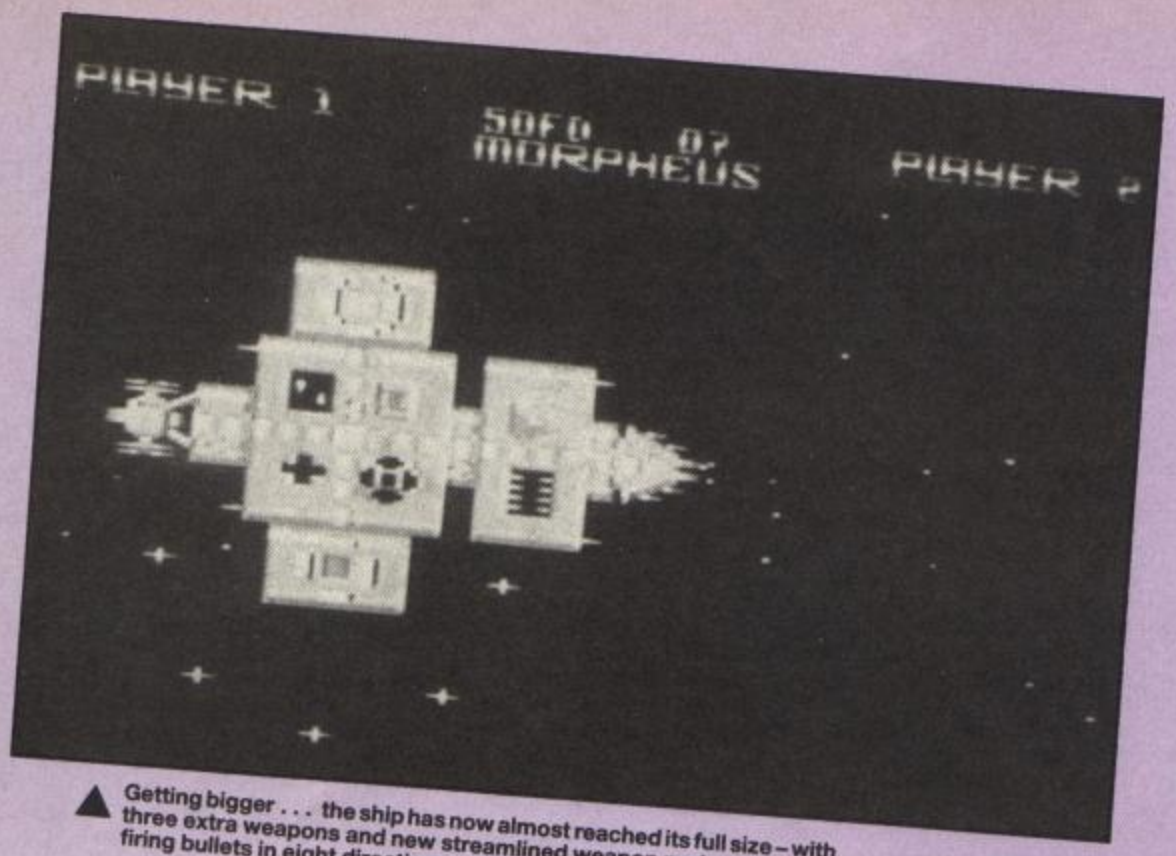
Put in some more code for upgrading the ship, namely selection, installation and scrapping of on-board systems. This appears to be working after a fashion. It did hiccup a couple of times and actually crashed the machine, something that rarely happens these days.

Eventually disaster struck. I filled up the 384K RAM disk on the Opus with assembler files, so the marvellous assembler just quietly gives up. It doesn't actually report an error that it couldn't finish writing a file, it waits until the next stage falls over when it unexpectedly reaches the end of the file. A massive re-organisation followed to split the main code into two smaller lumps. This involved isolating the code that is unlikely to change from the stuff that I'm still working on. Of course, saying that a piece of code is finished is a guarantee that it'll need changing tomorrow at the latest, but it's time to be brave. This makes the assembly time shorter as well – it was getting boring, taking over two minutes.

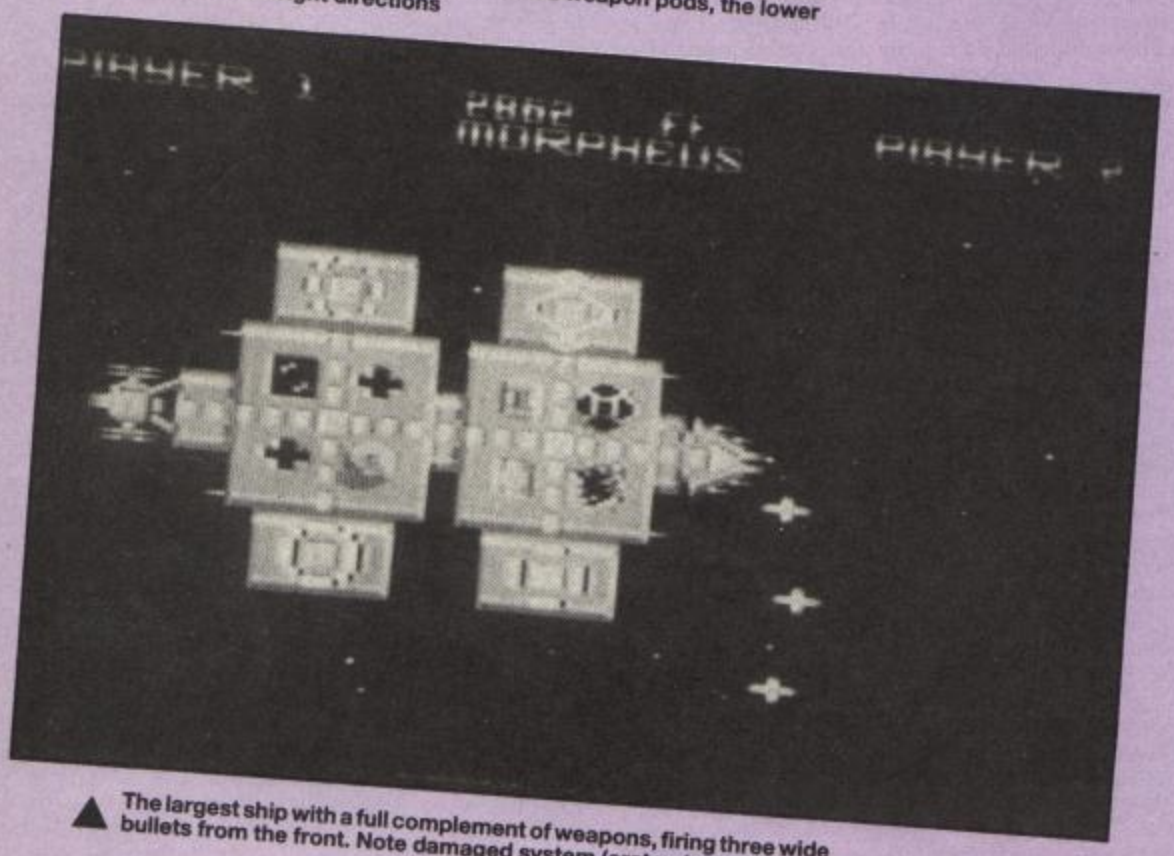
Having a lot of trouble linking the two modules together. All my routines have gone undefined, even though I can see them in front of me. Come seven in the evening, ST and I suss out what it is. In order to not have to key in all the routine names to make them known to other modules you can

▼ A small ship fires two deadly toothpaste tubes, which are quickly decaying





▲ Getting bigger . . . the ship has now almost reached its full size – with three extra weapons and new streamlined weapon pods, the lower firing bullets in eight directions



▲ The largest ship with a full complement of weapons, firing three wide bullets from the front. Note damaged system (crater, lower right)

just put 'ALLPUBLIC' which says that all the names encountered will be made available. However, this marvellous option doesn't pick up procedure names, which is what most of my routines are. Apparently sometimes procedure names are labels, and sometimes they're not. You can JUMP forwards to them but you can't forward reference writes to them. You can make them PUBLIC by naming them one by one, but not by declaring them public all at once. This makes the Avocet AVMAC65 assembler a dead-cert for the May Mickey Mouse Award, the first to hold it for two months in a row. I can't see anything beating this!

Friday 8th May
This scrap weapon facility is causing an embarassment. It's elevated itself to a 'scrap whole program' facility! I'm not entirely sure why this is, I'll need to look at a listing.

Meantime the select system facility requires that I display an individual system. I can only display eight at once and some configurations require that I display nine on screen, so I'll have to convert the character data into sprite data. The way I've organised it requires colour changes by adding one to each bit pair in the byte with no overflow, for the multi-colour data. Rather than painstakingly separate each bit pair in turn, I worked out the logical operations to achieve this in one go.

Andrew Hewson came along to discuss business and declared *Morpheus* to be 'Eccentric'. I've no idea whether that's good or bad!

Monday 11th May
Discovery of the day: my new programmer's calculator has a delete key on it, so no longer do you have to re-key the whole ten-digit number if you make a mistake – brilliant. How come it's taken this long?

Mr Penn himself rolled up yesterday with Mr Liddon but I didn't have *Morpheus* at home so they didn't see it. It can be very dangerous showing the press very early versions, even if he was on holiday. The freshness of a game wears off, I never did get my Gold Medal for *Uridium*! Nevertheless Mr Penn suggested a recoiling gun at the front, while Mr Liddon suggested finding some food! I'm all in favour of recoiling guns, so 20 minutes of updating the code and the recoil mechanism is in and working. It's quite a small gun anyway but it comes under the 'nice touch' heading.

Found the bugs in the install and scrap sections. I was tying myself in knots trying to ensure that the right messages appear on screen, supressing later ones in favour of important early ones. This resulted in sometimes not resetting internal pointers, so having installed the last weapon it still thought there was another one left, so it picked up rogue data and fell base over apex.

Tuesday 12th May
Finally completed the set of ship enhancement features. The game now allows all functions to be used without having to fiddle anything. I haven't yet put any restrictions on commissioning new ship parts. If I put in the bit where you have to pay for them, I'll need some money, which is derived from points scored – but there's no way of scoring points at the moment!

I keep running out of RAM disk space again on the Opus, it's just not big enough for a growing program, and certainly not enough for a grossly inefficient assembler.

I had to move the text bars further apart to allow more of the weapons stations to be visible.

In order to control the time taken for units to be built, the game notes how long you've been flying the ship. Pausing *will* stop the clock, so it's no good leaving it paused overnight to get the extra weapons built! I'm just wondering what units of time I should use – days I suppose. I'd rather use something more mystical, like the Dalek's 'Rels' – they use them to measure everything: time, distance, energy, mass, the lot. It must make Dalek Physics lessons much shorter.

I'd just like to add my two penn'orth to the 'which 16-bit machine should I buy to play games on?' argument. Let's get some facts sorted out.

1. There are as many arcade games out for the Amiga as for the ST, mainly due to the higher numbers of Amigas in the US. At the moment you just have to search harder to find the Amiga software.
2. The Amiga hardware was designed primarily by the designer of the Atari 8-bit hardware (Jay Miner, a very clever guy), and is aimed at doing all the things that games need to do, smooth scrolling, manipulating large graphic images quickly and producing high quality sounds.
3. The Atari ST contains just enough hardware to make it graphically superior to most 8-bits. It has a large colour palette, but cannot smooth scroll using hardware. The only smooth scrolling possible will thus be limited in colour, vertical only, on a small area or very s-l-o-w. Sonically it has a very old chip indeed, as used in the Amstrad CPCs and Spectrum 128s, not as powerful even as old SID.
4. Despite what Commodore *still* maintain, the Amiga chips were designed with games in mind and will be used as such. How much business software uses four voice stereo sound? Or smooth scrolling?

There will be some great games appearing on both machines, some already have, but anything the ST can do, the Amiga can do equally well, usually better. The reverse will never be true, provided programming standards are similar. In conclusion I would say that the Amiga to the ST is like the C64 to the Spectrum.

Wednesday 13th May
Added one or two new graphics to the set of on-ship systems which look quite nice. It's difficult to come up with a whole set of graphics which all look different but have to blend in with the same surroundings, but I'm sure you don't want to know about that! I've started to organise the 'ecosystem' within the game for the meanies. Each level will be played over two alternating phases with the ship able to commute between the two via the docking bay. I want to create different atomic-style structures of a central charged impenetrable nucleus surrounded by charged particles that have to be recharged by a charge-carrier that shuttles between them and the nucleus (phew). Each particle decays with time and must be revisited periodically. The main ship gun, nicknamed the toothpaste tube, can suck out and store this charge. This may cause other things to occur, and you may not be the only one trying to steal the charge!

Richard Groome paid us a visit and he may be able to help us with a little surprise on the music front, we'll have to do some feasibility studies, but we're hopeful. Can't say any more about that yet, very hush-hush, top secret and all that.

To Be Continued . . .

MENTAL PROCREATION

By Andrew Braybrook

Thursday May 14th

Today saw the inclusion of another of Mr Penn's little suggestions, a second high score table. One will be for the all-time greatest scores which will be saved back to disk. This will be a facility only available to disk owners as it's too much hassle trying to talk to a tape deck, what with switching tapes half-way through and allowing the C64 operating system to talk to the screen, it means that I have to be careful not to mangle the system variables, which I always do! Disk I/0 is much simpler, just squirt a few bytes down the serial bus and let the 1541 stir itself into action. Sorry tape owners, but cassettes were never designed for computer use, tape decks are even dying out on mainframe systems now. Still, you won't miss what you never had!

Found a neater way of indicating which system or weapon to scrap by use of an orange and black pointer moving around the screen, sounds familiar, and it is. I know I said last month that I wouldn't resort to icons or pointers, well I fibbed. Just one teensy-weensy pointer is necessary, but definitely no icons, never. Anyway this pointer is a lot better than the old Epyx White hand which rears its ugly fingers in Star Raiders II.

Friday May 15th

Rearranged the game's memory considerably to remove the need for my secondary set of 48 stars. I'd duplicated the starfield characters for use in two sets, as I'm doing some screen splitting where stars could cross from one set to another. This would look very untidy if they suddenly changed to a completely different graphic. However I recently tuned up the raster split timing so finely that the split occurs off the right end of the screen at exactly the right lines where the set changes. This splitting appears to be stable, even when sprites cross the boundary, so the stars are redundant from one set.

Minor screen glitches usually appear on screen when the programmer can't be bothered to introduce a very short delay to the screen change routine to wait until the raster gets to the side border. Thus all display changes are carried out while the raster is off screen. Changes to the display mode or colours while the raster is on screen invariably cause white flickers and the infamous glitch (more so on C128s). Sprites can cause the screen interrupt to be delayed slightly, so if all eight sprites line up over a screen change the glitch can move by a few inches – one has to take this into account, says he, attempting to excuse the glitch in GDO, which he spent ages getting rid of in Gribbly's Special Day Out. Of course all of this doesn't excuse the 'mega canyon glitch' as made famous by Wizardry . . . remember that? Some games seem to be emulating this glitch even today, and I did tell The Edge how to get rid of it.

Anyway all these rearrangements of memory give me another 1K in the video bank, room for another 16 sprites.

Monday May 18th

Had a quick perusal through all the sprites that I've created so far and not used. There turned out to be 140 of them, only a few up to scratch. I drew a few more and came up with a design for the remote drone, a six-frame animated vehicle. I can display this in characters while it's on the launching pad, then run it as a sprite when it's in-flight. This gets me out of wasting four sprites all the time, as the ship could be configured with four landing pads with remotes on.

I want to set up a series of charge orbitals in a pattern in the play arena. I was using the star explosion routine with predetermined start speeds to initially show their positions on a smaller scale. This didn't work too well, as some parts flew off screen at breakneck speed while others moved very slowly. Tomorrow I'll try another method. All I want to do is display a pattern rather like electron shells around the nucleus. The positions generated by this sequence will decide the positions of the charge orbitals in the game proper. I will be using sound to indicate proximity to the centre of the pattern.

Thursday May 19th

The patterned charge centre didn't work out. There were too many concentric rings with too few parts in each so it looked like a mass of dots (even though they were in a pattern). I decided to use the existing title screen explosion generator to produce the charge orbitals, which it does admirably. They could appear in one or two rings, a spiral or randomly distributed. I had to slow down the particles to get them to stay near the screen centre. I also fixed the bug in the position calculation routine so that the particles are moving at a fourth parallax speed in space. I've worked out that there will only ever be one of 32 orbitting particles or the nucleus on screen at any one time, so I need only assign one sprite to this function. This led me to define the remaining objects. Eight are for the ship's bullets, one for the remote, one for the orbitals, two for charge rejuvenators, two for charge supervisors, three for meanies' bullets, two for roamers, and five for other assorted bad guys. This keeps the top limit to 24, although I don't expect all of these to be on screen at once. My top limit of 32 is safe, and keeping the limit of 24 should safeguard the NTSC version which requires that CPU usage be kept down.

Roamers will be antagonisers, just wanderers that keep annoying the player in otherwise non-busy moments. Charge rejuvenators will carry charge from the nucleus to the steadily decaying orbitals –because of this decay, the rejuvenators will have to visit all orbitals periodically. The charge supervisors will shuttle around the orbitals ensuring that all is well.

Wednesday May 20th

Put in some of the coding to run the remote drone vehicle. This will sit on the landing pad until activated, and then buzz round the screen with its own gun mounted on top. Control of the main ship will temporarily be lost while flying the remote. I anticipate being able to leave the remote out in space near the ship as a decoy if required. I also think that it will become permanently lost if it strays too far from the ship. I would also like different drones to have different handling capabilities.

During launch the remote has to switch from being characters to a sprite so that multiple remotes can be carried without using too many sprites. Only one remote will be operable at a time, naturally, as one only has one joystick.

Thursday May 21st

Spent the day at the Institute of Directors discussing the ways of the World with Andrew Hewson.

Friday May 22nd

Worked out yesterday that the playing area of Morpheus is in fact 256 by 256 screens, per level, some 65,000 screens big. This is probably why it is quite difficult to locate the orbitals, as even moving at more than one screen per second it would take over eleven hours to explore it all! This is quite impractical, so I decided to shrink the Universe to a more manageable 64 by 64 screens, such power! This would still take about an hour to cover, but I'll be providing sonics to guide one into the centre, it's only fair.

Tuesday May 26th

Put in the remote vehicle movement algorithm but wasn't happy with it, so I took it out again. It just felt wrong as the movement seemed to always be in straight horizontal or vertical lines, this game definitely has a more 'circular' feel to it.

The remote drone is a peculiar beast, a sort of central vertical fuselage with a hole through the middle, the purpose of which will become clear later. It then has four protrusions, two at the top and two at the bottom, each supporting a rotating triangular block, viewed end-on to allow me to show off varying grey shades as the light falls upon the surfaces. The mysterious hole through the centre is for aiding landing of the remote by being able to see the flashing centre of the landing pad through it. Thus it is possible to line up accurately, knowing that it will land if the entire hole is flashing!

Wednesday May 27th

Changed the remote to run on a polar vector system which allows it to fly in circles if required – I'm not quite sure what to do with it if it's allowed to fly miles away from the ship! I also have to cater for all combinations of ship, I'm only allowing one active remote at once, but the ship could have up to four landing pads with remotes on. I have to prevent the take-off of a second remote, but allow it if the first is later destroyed in action. I shall have to enforce the restriction that a remote must land on its own pad, not another, due to the way the remote's firing characteristics are picked up. I know it would be nice to drive one's X-19 into next door's Porsche garage and get a Porsche 944 out the next morning but life's not like that, anyway, I've tried it and it doesn't work!

Tuesday May 28th

The remote movement system was completed by

allowing the ship to leave it in any place and fly away, the remote will keep going at the speed it was left at. It's best to leave it stationary if you want to find it again, and this enables the player to sacrifice it as a decoy. The movement turned out to be a bit tricky to tune up, so there aren't too many possibilities for good control, so I'll stick to the one set of parameters for all remotes, but still give them different weapons.

The co-ordinate system used for this is really confusing as there are two centres, one near the ship at the top left of the screen, and the other wherever the charge nucleus is. It's like drawing two pictures on two bits of tracing paper and sliding them over each other, with the additional complication that one set of co-ordinates can 'wrap around'. I just have to remember which set of positions any one object is running against.

Friday May 29th

I'm rounding off the ship improvement system by putting in the bits where you have to get involved in financial matters. Yes, I have a new routine called 'taxman', the bit where you have to pay for things. This requires setting up some prices and co-ordinating the score displays for two players. All scores accumulated in the current phase are added into the funds for buying new parts. I also have to display the current funds on the screen during the buying phase, so I persuaded the score update routine to do that for me. It didn't seem to mind too much, I just pulled the wool over its eyes and it was none the wiser.

I need a monetary unit for all this. I reckon that Intergalactic credits have been used enough, so I may well stick to good old Alleykat Guineas. They have a certain air of quality about them, like Florins, Furlongs and British Thermal Units.

Monday June 1st

Since the ship is run on energy it seems logical that the 'end of game' condition is when you run out of the stuff. Then what happens? Well normally the ship might blow up. I've never quite understood why electrical equipment has stacks of gelignite bolted into it so that the slightest fault can cause a firework display. I expect it was thought of by the same person that decided that bombing fuel dumps in scramble miraculously fills your tanks with rocket fodder. That's the excuse, now the problem. There's no practical way of blowing up a enormous ship on the screen. It was suggested by a Welsh correspondent that the ship should glow red, white out, when fade away into billions of pieces as executed so spectacularly by the Earth in the Hitch-Hiker's Guide. Well, take away the billions of pieces and that's what I've done.

Iv'e also designed a game logo on Deluxe Paint last night which may or may not be used in the official artwork. It's a full screen of logo so I can't really do it in the C64, although I may experiment with it for the tape loader.

Tuesday June 2nd

Finished off the graphics for the onboard systems and worked out how to run them all. It's all very well having a system selection utility, but it's better if the systems actually do something. I've got to take into account that some units can be fitted in more than one position to double up the effect, shields and solar cells immediately spring to mind.

Wednesday June 3rd

I have some system graphics that I don't know exactly what to do with yet, but I've put in some systems such as solar cells, battery units for energy storage, shield generators and replenishers, energy to charge convertors, charge to energy convertors, remote locators, orbital proximity detectors and . . . it even makes tea!

Some orbitals are leaving the Universe and I'm not sure what to do about them, I've randomly repositioned them at present which was fairly stupid because I now don't know the polar vector to get to it.

Thursday June 4th

Spent the day at Hewson's (does this imply that you did no work? – Ed).

Friday June 5th

Yesterday I half-inched some of John Cumming's sprites that probably won't get into Zynaps, which is a pity because they're really good. I had a good sift through them and as an experiment I changed all their colours to my three grey shades, and horrors of horrors, they all just died. This is why I've been having so much trouble designing sprites, my 'save a few cycles' method has finally caught up with me. Using the three greys doesn't offer enough contrast. I'll still use two greys but allow the third colour to be anything. I then proceeded to prove the point by designing an animated sequence of 22 sprites for the charge rejuvenator with no ensuing problems. Why has it taken me so long to discover this? Answers on a postcard to (cont page 202)

My onboard energy systems don't seem to be working some of the time – oh goody, an intermittent bug, my favourite!

Monday June 8th

A nice up to date listing has appeared on my desk courtesy of the ST Printing Co. This is good for spotting all the howlers. First off, the Universe will be variable in size, AB has decreed it. It will expand as the game goes on to increase the difficulty level. Any particle leaving this Universe will be caught and shot (or marked up as dead anyway).

The intermittent system bug was caused by the last system on the ship not being processed, hence the sequence in which the systems were installed was causing the systems to fail in different ways. Any one system would work on its own, but they wouldn't all work together.

I've added some more systems and weapons to the development sequence, which shows the current units that are available and steps through them as the game progresses. In order to make testing quicker I have a cheat system which starts me off with plenty of money and all units are built instantly.

I've also rigged the game to start on any of the first eight levels, but I'm not going to tell anyone how to select a level – other than to say that it has nothing to do with the keyboard or joystick. I figure that this will be a reviewers's initiative test, let's see who's paying attention.

I understand that a certain conversion from a certain Spectrum game by a certain Compunetter contains some Uridium sprites. I thought it was considered polite to seek permission for such a deed if this is indeed the case. Can't say as I'm particularly enthralled at having my graphics ripped off – originality it appears, has died.

Tuesday June 9th

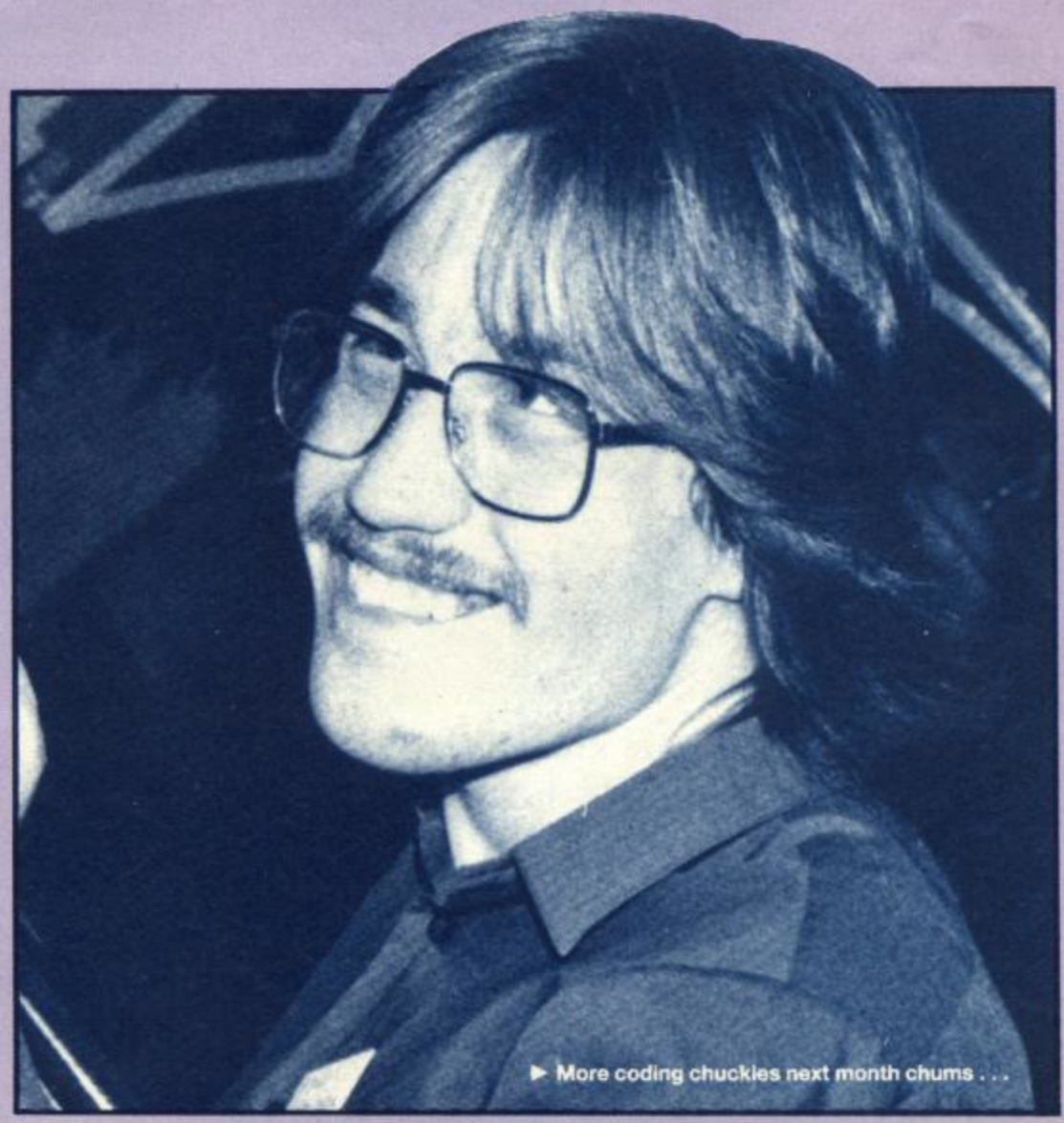
Another rearrangement of my source code occurred to move some more completed routines into a dark cupboard where I'm bound to need them again. On assembling I was left with something of a bug, the title screens were executing at breakneck speed, they decided not to bother with any of this 50 frames per second nonsense and go for about 300 instead. The explosions fair ripped out of the centre. A bit of exploration revealed that a routine had been corrupted by another routine running rogue. Lucky it didn't cause a disastrous corruption. Certain opcodes cause a total CPU shut down. After fixing the error I noticed that the docking sequence now takes much less CPU time, which is a pleasant surprise. I've no idea why, I expect it's magic.

Had to laugh at a CCI article on Andrew Hewson which said of Uridium, ' . . . too amazingly close, a Hewson rival commented, to a Sega coin-op that looked very like it but scrolled the other way.'

They were talking of Starforce which did indeed inspire the graphical style, but if they think the game is in any way similar then it's no wonder that their Uridium clones didn't sell. Apart from that it was very interesting, especially the picture of Cyborg Hewson, with a Manta perched on his shoulder, beats a parrot any day!

Wednesday June 10th – Monday June 15th

A brief interlude is to occur, whereby AB has a few days relaxation at Alton Towers to escape the election.



► More coding chuckles next month chums . . .

MENTAL PROCREATION

By Andrew Braybrook

Monday 15th June

ST had been trying to cure our data transmission problems last week. It's now quite rare that we successfully download a file for testing, and they're getting quite large so the chances of getting an error has increased. By keying in a quick analysing routine we discovered that sometimes the Opus doesn't switch its data-ready signal off very cleanly, so much so that it causes a second trigger. This only occurs briefly, but long enough to fool the C64 that another byte of data is ready. Therefore our receiving program reads in the supposed new data, and when it comes to check-for-errors time it discovers that someone has made a boo-boo. By checking the CIA latch a second time after the data is accepted, and ignoring it we have alleviated these 'ghost' images. No errors have occurred in the ten or so downloads since.

I've fixed the bug that allowed the ship to pass through charge orbitals in two directions, and I'm reasonably happy that everything is working correctly. I've put in an extra piece of information at the beginning of each phrase which shows the current level and what I'll call the current 'timeslice'. This is an indication of real time taken playing games, which is used to to derive the effectiveness of your weapons against the enemy. A weapon built in timeslice ten will be fairly ineffective by timeslice 20. Each timeslice will represent about two minutes of play.

Tuesday 16th June

Half day today. Got to grips with the compacted sprites and organised the ones that I'd drawn already. I then noticed that that many of the images were symmetrical top to bottom. This led me to try further compaction by only keeping the top half, and then reflecting the required images prior to use. The routine turned out to be quite small, and certainly simpler than reflecting left to right, which I had to do in Paradroid. Unfortunately the decompaction system went slightly wrong when the 'it was written so long ago it's bound to work' sprite header routine failed miserably! It was supposed to nip through the compacted sprites and note where each one starts, which saves me either reading through them all every time I want one header, or holding all the headers in a table all the time. However, it managed to miscount so it got out of step with the actual images, so by the time the individual sprite decompacter got there it was picking up the wrong data completely.

Wednesday 17th June

I've never written one before but it suddenly dawned on me that an automatic sprite animation system would be a good idea. A lot of space has been wasted in previous programs by objects each having their own bit of animation code, like 'every fourth cycle add one to the sprite frame and if it's bigger than 'X' then subtract seven from it.' This can be done on a similar basis to the automatic sprite colour system, which relies on any one sprite being used for one purpose only. This may require some duplicate sprites, but since they're all compacted I don't mind.

I had a discussion at the CBM show on Sunday about the merits or otherwise of high-ish level languages – mainly C. Personally I think it's a pig of a language, as it is totally wrapped up in its own syntax structures. It has two ways of specifying equals, either '=' or '==' depending on whether it's in the equivalent of a BASIC LET or IF. Every other language I've come across manages with only one symbol, they know which one you're talking about by the context of the line. C also makes you put curly brackets round multiple statements within an IF-ELSE structure, something which COBOL achieves by use of nothing more than a carefully placed full stop.

Thursday 18th June

The sprite animation system is playing up. Objects disappear, flash on and off and go through the wrong sequences – anything to get out of working properly. I checked the object handlers and the animation routine, found a few errors but all to no avail. It took until 4:30 to find the cock-up. It was the animation instructions that were wrong. Apparently eight plus three is not 83 at all! This causes sprite sequences to jump about wildly, sometimes picking 'suicide frames', causing objects to automatically delete themselves on the final frame of explosions etc. I thought it would be tidier if objects cleared themselves away.

Friday 19th June

Spent much of the day on the sprite editor. I want a design for the charge supervisors, which will travel around looking for trouble – that is, as soon as an orbital is attacked they will head towards it. However all I managed to design were a few more roamers.

I haven't had any transmission problems at all this week so it looks as if we have correctly diagnosed the fault as being cheap and untidy electronics at the PC end.

Got a rough draft of the artwork from Hewson's today. They've been looking at my Amiga artwork, so their artwork is quite closely related to what I'm intending for the game. I'm reasonably impressed by the layout, but they didn't use my logo, just some old Paradroid-style lettering. They said my logo looked like a row of coffins in space.

Monday 22nd June

Tidied up a number of loose ends and fixed all known bugs. I had to make the Universe slightly bigger as some particles were intent on leaving it. I also reduced the size of the fastest polar speeds, the fastest bullets were just flashes across the screen, not very practical for collision detection.

Went on a programmer's fitness course at the weekend, which involved transferring graphics from the C64 (downstairs) to the Amiga (upstairs) for enlargement and enhancement. This involved memorising graphics on the C64, racing upstairs, and redrawing them on the Amiga – I bet it's still faster than RS-232. the net result is that I've drawn the 16 system units in 32 glorious colours at double the size for inclusion in an accompanying booklet.

Tuesday 23rd June

Toyed with the smart bomb weapon to get it to work. It'll be a medium-term weapon rather than a once-off blast. It'll start off at maximum strength and decay to zero after firing, so it'll affect meanies arriving on the screen. Also it won't necessarily kill meanies outright, especially if the weapon is getting old.

Also put in the collision detection for my own bullets. Having a maximum of eight, and there being eight bits in a byte makes things quite simple. The bullets are small but are moving quite fast, up to eight pixels per move, so character accuracy is all that is required. Each meanie will check the character position under its centre for the presence of a bullet. I'm not using sprite to sprite hardware collision detection for a number of reasons, mainly that it isn't all that helpful in multiple collisions knowing that sprites one, two, six and seven have collided somewhere. Which one has collided with which, or have they all met in the middle? Many collisions are irrelevant and needn't be checked.

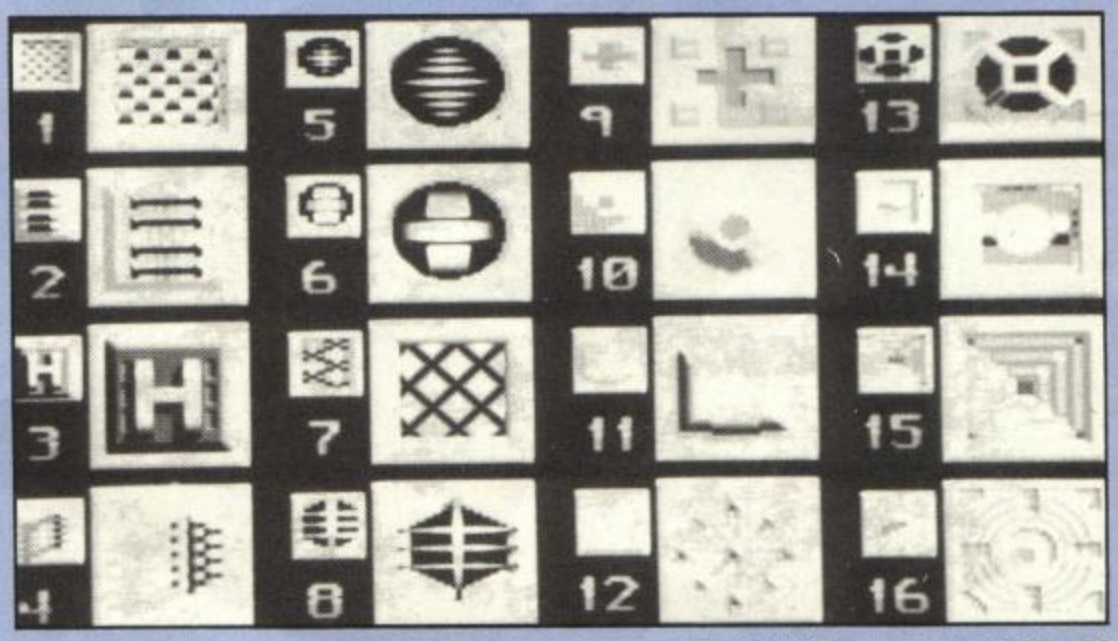
I also worked out all the data for the weapons development table, all 57 weapons. Each has its own firing type, bullet fired, reload time, construction time, graphic number, and cost to build. Any volunteers to check that they're all present and correct?

Wednesday 24th June

Been doing some random spot checks on the weapons. I'd managed to ruin the sequential fire system. It fired one bullet, waited for ages and then released the next seven in quick succession. Apparently it had waited for the gun to reload before firing the remaining bullets.

I've started putting in all the bits that nick the energy, including collisions with orbitals, and firing the guns. Yes, the guns take energy to fire, life's like that. They don't take very much though, without any energy replenishing systems the ship is still good for some 1500 bullets. This will become virtually unlimited once a solar cell or other device has been installed.

I've also improved the smart bomb system. It kept firing by accident as I left the unit, which isn't damaging with normal guns, but the smart bomb can drain up to a fifth of the total energy so something must be done. It requires a delay before



▶ The 16 systems which can be incorporated into the ship

► **The Morpheus logo as it appears in the game**

it fires, so I decided to use this delay properly, not just a delay for delay's sake. It counts a timer upwards and will fire when it reaches the bomb's preset value. However, as it counts up, it checks itself against the current energy level. On firing it will use up the amount of energy equal to the timer, so if there is insufficient energy available during build-up it will abort firing. The later smart bombs will build up quicker and thus use less energy – they'll reload quicker too. I'll use sonics to show this build-up and it'll be possible to abort firing at any time.

Thursday 25th June

Some of the charge orbitals were appearing a bit late on the screen. It seems that updating one every 32 cycles wasn't quite enough so I've doubled them up so that two are updated every 16 cycles. This seems to be running everything more smoothly, except that now they're all decaying twice as fast. This means that I have less time to find them. This won't be the case in the finished game as the charge rejuvenator will be visiting all the orbitals in turn, but he's not coded up yet. It's quite difficult to find the orbitals at the moment, so I rigged up one of the systems to glow when there's one nearby. This should be useful on later levels.

I've enhanced the shield's generators so they show their current status with colour, and you can also buy another system which shows the current status of the whole ship's shields. I have three remaining systems that don't have a current purpose.

For the first time I can now score points to earn money to buy weapons and systems. I've currently got a cheat version that gives me buckets of money anyway and allows me to build any of the weapons instantly. The systems don't decay with time, as it's really the meanie's growing immunity to the weapons which causes the weapons to fail, but the systems can be blown up if the shields collapse.

Friday 26th June

Put in a new system device – a direction to nucleus indicator. This should help navigation tremendously. Begin an eight-directional indicator, I thought I'd borrow a calculation routine from Paradroid that decides which laser bolt frames to use. One problem though – ST and I realised that it didn't work . . . well not quite anyway. We also realised that it only has to calculate four directions, it has no need to differentiate between up and down as the bolts are reversible. We therefore decided to work it all out from first principles.

At first our marvellous indicator was incapable of showing diagonals. We had used the line equations the wrong way round when working out whether a point is above or below $2X = Y$ and $2Y = X$. The direction finder could turn out to be useful, so I've made it a separate routine from the indicator so that I can use it later for other functions.

For better between-game continuity I think I'll rig it such that upon demise the player's funds remain for the next game. This will allow a quicker building up of a new ship.

Monday 29th June

Put in some close manoeuvring to stop the ship if it's moving very slowly. This helps with close positioning of the ship and also lessens the times when stars are moving very slowly. It took me a while to suss out that this was not working in the engines section, as it has a quite get-out clause normally to check for dematerialisation.

I put in a top limit for the amount of money that can be carried forward to the next game. I can just see some idiots playing level one and then quitting for half the night to build up a mega-fortune for one game. Tough luck, cheats!

Over the weekend I spotted a double star flitting between two places on the screen at high speed. I've been running the game for a while hoping to reproduce this error so that I can investigate, but will it happen again? No chance!

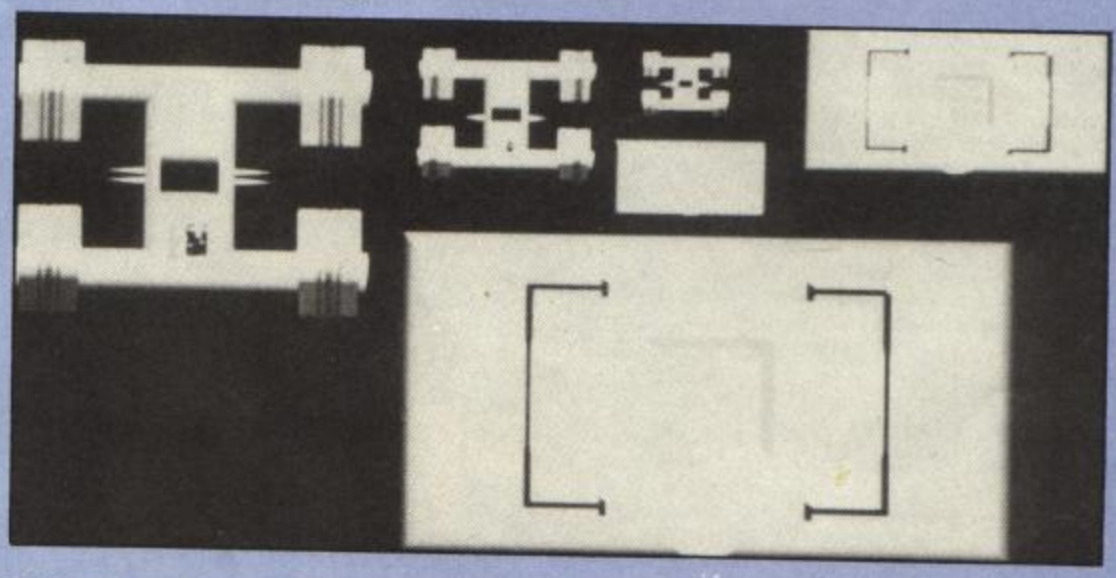
Actually got recognised in Tesco's, so to ensure that I don't get run over by a shopping trolley next time, hi to Rob and John!

Tuesday 30th June

Put down on paper all the ideas for meanie movement and initiation. This is the last big push to get the game in a playable state, the rest will just be tuning up and adding some frills. Most meanies will be generated as a result of altering the change of an orbital. I shall start them off on preset launch patterns and then switch them over to manual control where I hope they will behave with a bit of character. I want a more varied spread of speeds through the levels, and meanies will be able to generate bullets or other meanies.

Bought Slapfight last Saturday. Great game in the arcades so I had high hopes of a good game. It's beeen converted very well, good playability and visuals, well done, but what about all the program refinements? No pause mode, no quit game, and to cap it all it uses sprites in the top border for the score. Well I can't see the score on my TV set, it's off the top. Black mark for that one, why is it up so high?

► **The remote droid and landing pad**



Wednesday 1st July

First day of overtime, I was scheduled to complete Morpheus yesterday, but I've missed a number of days work for one reason or another and I really want this game to be something special – so it'll come out when it's ready. This is an artistic expression, not something off a production line.

Started coding the meanie initiator and control routines. I didn't feel like coding a lot of routines up again slightly differently, so I decided to adapt the ones that already run the bullets and the remote. This saves code and simplifies things (famous last words).

Sure enough I ended up with no bullets and an invisible remote. Haven't the faintest idea why. The objects are getting initiated in the correct places, they just die immediately. Last time anything did that it was the animator's fault, but not this time.

Thursday 2nd July

Found the no-bullets bug last night by staring at the listing. Apparently someone had put two instructions in the wrong order. Wait until I find out who that was. The disappearing remote took a while longer, but was another typing error. I'd taken the Y co-ordinate of the remote's position, added the Y movement and then stuffed the result in the X co-ordinate by mistake. When you're convinced that a piece of code is working you just read what you want to see, not what's actually there.

Continued to write the bullet and meanie initiator and handlers. Since meanies can fire other meanies instead of bullets, I can have a whole sequence of meanies. They have different conditions for generating others, randomly, only when wounded, or only when killed. Generally meanies will only take one shot to kill, but outdated weapons will be less effective. Injured meanies will have different flight patterns, usually wild retreat, but slightly scratched ones may well get vicious. The smart bomb should have an interesting effect on them, especially an outdated one.

Was interested to read in this month's ZZAP! that multi-loads are okay if they load the next level while you're playing the current one, even if it's deliberately lengthened to give the loader time! This type of loader is loading data by getting the cassette to cause interrupts rather like Novaload does. This leaves about 50% of the CPU power to the main game and no interrupt capabilities. You could get more CPU steam up by slowing down the loader, and you could try to split the screen using NMIs, but I suspect that screen splitting would be impractical as it is heavily tied into the progress of the raster, which stops for no man. So this type of loader is fine for games with no raster splitting and little CPU usage, but don't expect that sort of thing in Uridium Plus 2 and Alleykat's Revenge which would use all available CPU wellie most of the time. This said, there are moments in games where very little is happening, for example when 'Player Ready' messages appear, so short bursts of I/O are possible. Searching tapes is not really practical though, and waiting for CBM disk I/0 is like watching paint dry. I think the short-term answer is better data compaction. I could have loaded each Uridium dreadnought from disk into its 9K buffer, each is 512 characters wide by 17 deep, but by compacting this data I could specify each one in about 600 bytes. Thus I could fit 16 layouts into about 12K with overheads. If that were all decompacted at once it would take 144K, more than two C64s full!

Many multi-load games don't bother to compact data like background pictures, because they have already accepted that disk I/O is inevitable, so what's another ten seconds of load time, if they don't have to spend any time working out how to get the best use from data. If they thought about it more they could cut multi-load I/O times down by 75%, but the perceived value of a game that loads from disk is much higher because you think you're getting more for your money.

Friday 3rd July

Finished off the meanie and bullet handlers today, now all I've got to do is get them to work. I want the bullets to hit the shields and explode if the shields can take the hit, otherwise the bullets will skid across the surface of the ship doing more damage. I also want the meanies to bounce off the ship in a realistic manner, or get squashed against it if they can't get out of the way, such is the power of a large ship.

Realistic bouncing is always a problem, because although it's fairly easy to detect when the ship has been hit, it is not so easy to decide what direction to bounce off at. I got round this by defining a perpendicular direction from the face of each ship character. Any meanie approaching a block can be reflected across this perpendicular axis and pushed away. I also enhanced this by adding that if a meanie approaches from an unusual angle it will be allowed free passage.

Fired up the game after a multitude of assembly errors had been fixed. All was going well until I fired at a charge orbital which is supposed to release from one to eight meanies or bullets. This however did not happen. What did happen is that the game totally froze. Now I'll have to take out the routines one by one to find out which one caused it. This is always a problem when you add lots of inter-dependent routines at once. The code seems intact, it restarts okay after reset without reloading any files. There may well be an infinite loop coded in there.

Monday 6th July

There I was, checking all the routines for possible reasons why the machine locks up, and I came across a JSR $0000, a call to a routine at the 6510 data direction register? I think this could be the cause. The jolly old linker has left a gap in the code because it didn't know what the real address was supposed to be. It didn't bother to tell me that it didn't know because ever since day one it has whinged about not being given a transfer address. I don't even know what one of them is. I'd gladly let it have one but I don't know how to tell it either. The manual doesn't mention transfer addresses. I got so fed up with it telling me to give a transfer address that I told it to keep messages like that to itself, so it kept two unresolved labels to itself too. This is altogether more serious, it works without a transfer address, but it sure as Hell won't work with unresolved labels in it.

Tuesday 7th July

Debugged most of the meanie routines, and now I've got enough data in the game to generate several different types in a number of different ways. They are firing flak at me which can blow up onboard systems and damage the ship. Their manual movement patterns are switching in, but aren't yet positive enough to force them to move in any particular way, it's just a question of line adjustment.

The main problem that we've come across is that the game consists of moments of high activity followed by longer periods of travelling to another orbital. Finding the orbital is a bit haphazard. There are a number of systems to aid navigation, but I don't want to make them all available at the beginning. I've come to the conclusion that I need a medium range radar display.

Wednesday 8th July

I didn't want to put a radar screen in, but if the game requires one then it shall have it. Now, where shall I put it? I can't incorporate it into the main ship design, it's too big. I *refuse* to make it a sprite or two and bung it in the top border. I'll hang it below the game logo. I drew some scales round the edge to make a border for it but it looked rather sketchy. The actual coding didn't take very long but on firing up it didn't work. Not a radar plot in sight. Upon moving about I could occasionally get a dot to momentarily appear and that was all. I studied the code for ages and there was no way at all that it could possibly fail, but it did.

By 6.30 I was getting very cross indeed, the C128 nearly got thrown out of the window. It then suddenly dawned on me what I had done. Since I am plotting two orbitals every cycle it takes 16 cycles to prepare all their positions. I then spent the next 16 cycles copying one row of the radar to the screen, so this also takes 16 cycles to complete. This copying process is spread over some time to avoid doing time-consuming operations all at once – it also carries out the function of clearing out the old radar images once they have been copied to the screen. All this was jammed into the one routine, the first 16 cycles prepare the radar, the next 16 copy it to the screen. Now the problem is that this routine is called *twice* each cycle, so it copies the radar across, clears the old image, copies the now cleared image across again and finally clears it again. The set-up needs to be done twice, but the copying and clearing must only be done once. How could I be so stupid? Don't answer that.

Thursday 9th July

Changed the radar surround to a more solid border and built it up into more of a crest to fill it out. ST suggested that it be a different colour from space to distinguish it from the background so we eventually decided on blue. It looks quite neat now and serves its purpose very well.

Paul Hughes dropped by to discuss some new anti-cartridge techniques and loader. He left his Koalapad with me as I intend to use a bit-map picture as a loading screen, hopefully for the disk and tape version. I've done a mock-up on the Amiga-beast and it all looks feasible, trouble is I don't know how to use the Koalapad. I've tried turning it upside down and rolling it about on the table but I don't have big enough area.

Friday 10th July

Spent much of the day thinking about how to run the charge rejuvenators, the ships that periodically ferry charge from the central nucleus to the orbitals to counteract their decay. This involves getting the craft to the orbital (easy), carefully driving round it (not so easy) and finally docking with it from above (difficult). As the orbitals are positioned in a circle or other pattern around the nucleus then some are easy to approach from above, others are much harder and require more complex guidance.

I've changed my mind about the charge supervisors, apart from deciding that I only need one at a time. I'll make it appear in the distance and slowly approach to use the 3D depth effect a bit more. It'll appear after a while around any attacked orbital.

Monday 13th July

Put in some enhancements that I'd thought of over the last couple of days. I had to reduce the number of active meanies to six as they are eating up the CPU time. This isn't too much of a problem as I've also thought of a way of keeping them on screen without them crashing into the ship, they now run circles round it like Nigel Mansell on Silverstone.

I put in the code to run the rejuvenator, and after teaching them a bit of basic navigation they can now find their way from the nucleus to any of the orbital. They have to move quite slowly as they arrive, so that they can find their target accurately. Then I reveal my coup-de-grace, the 16 frame animation sequence to drop their charge and replenish the orbital.

I'll have to write the game instructions out soon, I've already done the page numbers, now all I have to do is fill in the rest.

► **My interpretation of the logo – now I ask you, does that look like a row of coffins?**

MENTAL PROCREATION

By Andrew Braybrook

Tuesday July 14th

Drew some new sprites, including assorted meanies and the dreaded charge supervisor – who turned out to be a weird-looking thing altogether. The sprites that I'm doing will be one of two or maybe three interchangeable sets. I was going to have one set for the positive phase and one for the negative, but I may have room for a third set after all. The animation and colour information is fixed, so corresponding elements in each set must behave similarly.

The rejuvenators have turned out to be very slow at getting to the orbitals. I have speeded up their final approach but they occasionally get a little lost.

Wednesday July 15th

Re-organised the rejuvenators's method of getting to their targets, which fortunately simplified things considerably. They now don't necessarily head straight for their destination to start with, but when they get close they're usually already above it so their final approach is much quicker.

Drew a bunch of new sprites for some more assorted meanies and bullets and put in some more data to use them during play. I've made some adjustments to the ship's slowing down mechanism that kicks in when it's moving slowly with the joystick centred to bring it to a halt. Normally the ship will drift freely in frictionless space at high speed for any-directional movement, not just eight. The slowing down system is for accurate lining up on targets which are not moving.

Thursday July 16th

The smallest ship can currently carry two systems, both of which are fixed and indestructible: the energy display and the charge indicator. This leaves no room for expansion, which is partly desirable as the newcomer to the game won't have to worry about battling with the ship modification system, but partly annoying to anyone who wants to protect their ship a little. A bit of tinkering under the bonnet has seen this altered so you still start with the same two systems, but the charge indicator is now destructible and changeable – it can now be scrapped and replaced by another system.

Thought of a brilliant new system to incorporate, an emergency dematerialise system that kicks in if the energy drops below a certain level – effectively a safety valve. Following this idea through, I wondered what would happen if there was nothing that could be done to replenish the energy before the next visit to the play arena. Well, the system would just fire off again straight away. This would result in an infinite loop. Programmer's solution? The system must self-destruct when it has been used. The excuse? The system has to supply a large surge of power to activate quickly and burns itself out. Not bad, eh?

Friday July 17th

The rejuvenators are still getting lost sometimes. They always get where they're going if I follow them, and they also succeed if I wait by the nucleus and watch them come out at regular intervals. They don't work when they have to visit the opposite side of the Universe from me because the co-ordinate system wraps around, but not very consistently amongst all the various relative distances that are used to calculate the rejuvenator's current position. Result? Confusion, for me *and* the rejuvenators!

Monday July 20th

Managed to 'mend' the rejuvenators almost totally beyond repair as they couldn't find a tree in a pine forest this morning. I sneakily followed one and it got nearly to its destination and then hared off in the opposite direction. I'd decided to set fire to the C128 if the bug didn't come out by lunch-time. It was close too, and anyway, why does the compare instruction set the negative flag? Surely all you want to know is whether 'A' is equal to, less than, or greater than 'B', not whether the difference is positive or negative. All this means is that you can now say: 'Well, not only is 'A' greater than 'B', but actually it's buckets bigger.'

I also managed to find a well dug-in bug that had been there for ages. It was in the movement routine which is an area where things are difficult to trace as it's a dynamic thing, you can't just stop things and examine them as the whole movement over a long time is being controlled. Anyway I found it, and now the meanies are following proper patterns as instructed, which is quite impressive though I say so myself. I've come up with some more movement behaviour patterns including the infamous Uridium homing mines, and some pods that shoot out, perform aerobatics and then stop. If you hit them with a weak weapon they go absolutely bananas, firing bullets and lurching around.

Tuesday July 21st

Designed most of the rest of the sprites for the positive phase set leaving only eight to do. At this point I decided to create the negative phase set where there is a one-for-one correspondence between the two. Each sprite has two images, each meanie has two appearances. In some cases it may mean reflection, in others a reversal of animation, and others a total redraw. It took about two hours to get through them all. I'd still like to design a couple of extra roamers before I'm finished.

I hope to show the meanies developing during the game through the graphics and movements and also what they fire. Some early ones may not fire, but learn how to, some may home in very badly but get better at it, one will even try to impersonate a charge orbital towards the end (and maybe even other elements in the game).

All 256 images, which would normally take 16K were slowly and carefully compacted down to under 8K. I could even put in another 256 images if I were that way inclined, but I'll only do it if I can't think of a better use for the space, like a 10K bit-map test card, much more useful!

Wednesday July 22nd

Andrew Hewson paid us a visit today, so there wasn't much progress made on Morpheus – although I did write a brief storyline for background information and I also wrote down some interesting facts about the game, such as it has 2,403 sprites all on screen at the same time, 2,395 moving stars in the background in 452 parallax layers, 72,000 colours on screen at once, a full 68-piece orchestra playing during the game at CD quality, running simultaneously with a digitised after-dinner speech by the Pope.

Thursday July 23rd

Doesn't tempus fugit? Today saw the inclusion of the roamers into the game. Piece of cake really, it turned out that the standard meanie routines were quite capable of running them with no alterations so I just had to write an initiator. This carefully attempts to place roamers, wandering meanies or maybe rocks, roughly in the ship's path to give the impression that the place is full of them. Works too!

For an encore I also decided to put in the bonus sequence whereby as the requisite number of orbitals have been de-activated the nucleus decides to shut down so all the other orbitals collapse and you have to race back to the nucleus as it release spinning 'Morpheus Symbols'. These can be destroyed by any means, fair or foul, for extra bonus points. It releases one symbol for each orbital personally destroyed, so since on level one you only need to destroy one to unbalance the system then only one symbol will be released. Come level 32 the place will be full of Morpheus symbols around the nucleus, although they are very short-lived and expire in a few seconds.

Friday July 24th

Just tidied up yesterdays's routines. It sometimes counted the orbitals wrongly, but I soon found out why. It wasn't counting the ones that the rejuvenators killed by overloading them off-screen.

I drew the last eight sprites for each phase and tidied up some others . . . so that's the graphics about finished. It took BASIC six minutes to compact them to just under half size.

I've put in most of the meanie wave data just to try out most of the manual movement modes and check that the right meanies are coming out. There are about ten different types which begin fairly stupid in different ways and each develop through the game, learning how to fire, firing better weapons, attacking less clumsily or just becoming plain nasty.

Monday July 27th

Had one or two people look the game over and we decided on a few improvements, so the radar now has a cross-hair sight to make finding the orbitals a little easier, and I've shrunk the Universe a little too. This has the knock-on effect of requiring possibly two orbitals near the screen where one was previously the limit, deliberately. I thought this would be

a toughie as many other routines assume that only one orbital can be on or near the screen at a time. In the event it didn't turn out to be too much of a problem. Sometimes I even get the impression that I understand some of these routines.

I've tried out many of the levels and it's currently rather easy early on but it gets a lot more difficult at around level 12. I'll just have to shuffle the levels around until I get a good balance.

Tuesday July 28th

Couldn't face the thought of making up and keying in buckets of data for the meanie waves that get released, so I decided to let the routine make them up itself. I still ended up keying in a large table of data but it didn't require quite as much thought!

Made up all the data for the various systems that can be selected and bolted onto the ship, including their build-times, cost and efficiency. Some will be available early on and get phased out, others will be 'invented' later on.

I like to think that these reflect a real situation. As time passes the weapons and systems get better, and usually quicker to build, maybe cheaper, with pricing wars going on between the various manufacturing companies, I'd like to write a proper history of all this.

I've been playing the game to try out the difficulty level. It's fairly easy to clear the early levels without too much hassle. I tried out level 38 and didn't last long! I've changed the level completion condition so that a maximum of the orbitals need to be destroyed rather than all 32 of them which should speed up the whole pace of the game.

ST has been developing some sound effects on the C64 and driving himself mad under the headphones. Every now and again he lets me hear one through the TV speaker and the whole office shakes!

Wednesday July 29th

Found a couple of well-embedded bugs that were so crucial that they've gone totally unnoticed since about April. The title screen had only been displaying six sprites instead of eight, but since they're all on top of each other it's difficult to tell.

The supervisor concept is not required, the meanies are quite nasty enough on their own, so I seconded the graphics for it as another meanie type. I really need to think up some names for all the inhabitants. I want to call some little spinning rings 'Ubiques', (pronounced You-Be-Kway). because they get in everywhere – it's from the Latin you know.

Now the graphics and most of the level, weapon and system data is in. ST has nearly completed the sound effects, so we're almost done.

Thursday July 30th

Started putting in the sound and fine-tuning the game. This is the bit where we spend more time playing the game than coding anything, but it's probably the most important bit, getting the playability right.

Doubtless it won't be right for everybody but as long as some find it a little easy and others a little difficult then we've pitched it about right.

Allocating sounds to the part of the program that require them is always fun, sounds go off in the wrong places, for too long, or not at all. It's just a case of knocking it into shape.

Friday July 31st

ST has completed the sound effects and has now turned to the music. He's done some really low 'sub-sonic' sound effects that'll certainly shake the dust off you TV set.

I've improved the control mode considerably to allow easier switching between weapons and I've come up with another possibility, the rapid-fire weapon. Hope I have time to code that one up.

Monday August 3rd

Been playing the game over the week-end and two playability problems loomed. One is that the game failed to kill me off after a lengthy game, which is curable by increasing the slope difficulty to make it meaner in the later levels. The other is that the bullet-firing weapons are basically useless! They're too slow, with too few bullets requiring too much accuracy to hit anything, even the fastest firing guns aren't much good.

The limitations of the program are that eight bullets is tops, fairly small ones with consequently fine collision detection. Thus I've decided to scrap the lot! I've put in a more global systems whereby unseen bullets are fired. A flash of flame is seen coming from the gun that fired and all the collision detection is done behind the scenes. I can make the bullets as big and fast as I need to make the game work. This frees up a little more CPU time too.

Tuesday August 4th

A certain publisher (who shall remain nameless) moaned that the nucleus doesn't do a lot, it just sits there and throbs. It didn't take long to modify that. Now it spits out bullets at irregular intervals up to eight at a time. I've let it have up to 12 bullets on a C64, and 15 on a C128. It now looks menacing and is not a place to stay for a cup of tea. It reinforces the idea that the nucleus is the villain and must be destroyed but not by wading in there with all guns blazing.

I've made selected meanies more aggressive to start with and swapped over the homing mines so that the ones with a bit of random element in come later, as they're much harder to shoot than ones that home directly.

ST has completed the music, and unless I think of any more sound effects, they're all done too, all 53 of them.

Wednesday August 5th

Mostly a day of tuning up, and not playing pianos either. Found a few things that didn't, and never could, work. Again fairly subtle things that had gone un-noticed. It's fairly easy to spot a mistake once you know that something is definitely not working.

Made the meanies more trigger happy and I can't get past level 12 out of 50, so I'll probably back that down a little. I've removed part of the concept of extracting charge and ferrying it to the opposite phase to plaster the negative orbitals. This was making it necessary to transport back a little too often. Now charge extraction and consequent meanie release is not dependent on the shop having some room for charge, except that no points are awarded for extracting charge that cannot be carried so the wily player will till transport back to switch phase, but no-one is forced to do so.

Thursday August 6th

Went up to Hewson's with the new version of Morpheus to show them how to play it. It is virtually impossible to play without instructions, and I haven't written them yet because I may change my mind about anything. I've deliberately made it difficult to understand without instructions because the game has subtleties and complications that need a while to observe and explain. After all, if you'd never seen cricket before and were given a bat, ball and stumps would you get the rules right? I doubt it very much.

John Cumming and Dominic (designers and programmers of Zynaps) sit in on a think tank, and having understood a bit more of what was going on, started getting into the game. We then had a lengthy discussion about what the game is, and what it isn't. Ideas were put forward to improve it and the need for a detailed instructions manual was expressed, with screen shots to back up the text.

We also saw the advertising artwork for the first time in all its glory. It's very pretty with a gorgeous starfield.

Friday August 7th

I've made the larger ships a little cheaper and given all the systems and weapons a two-letter code to give them more indemnity. The control mode is still causing some arguments. At the moment it feels a little like Gribbly to control. The thing is that is has inertia and acceleration to make it feel like space, which requires more skill to control than simple inertia-less system. It's like comparing Asteroids to Space Invaders. Still, I've altered it slightly to give more accurate control. I've completely re-done the systems list to include a new ECM unit and to make the more useful systems available earlier.

Monday August 10th

Began work on a pre-game 'meet the meanies' sequence which means I now have to think of names for them all. I expect people will think of some of their own as well!

I think I need to award more points for later meanies to compensate for having to replace all the systems that they keep blowing up!

Just got the September ZZAP! and came across Andrew Johnson's Rrap. He mentioned the Atari St once too often for him not to own one. I was merely pointing out the technical differences between the Amiga and the ST for the benefit of those who are not sure, and to set the record straight in contradiction to another publication at the time saying that they are very similar. There's always one computer owner ready to rise to the bait though. All this, and Johnson (for we are apparently on surname terms) accuses me of telling porkies. Well, anyone with a colour TV can count the colours being scrolled on Goldrunner, one, two, three, four out of 16 which is what I'd call limited in colour. The other colours are sparsely added later for the ships and bullets. The main playing area on Metrocross has how many colours? Black, white, green and blue, I make that four again. You don't have to ask me, go and ask any honest ST programmers, I know I have.

Tuesday August 11th

Finished off the title sequence to show off the meanies, some with blank names because I haven't come up with many names yet. The limited area on the title screen coupled with the sprite multiplexor running means that the sprite positioning had to be pixel perfect to actually work properly, there's no room for any play in the vertical positioning at all, the sprites are just re-cycled in time.

Prepared a version to take to ZZAP! for a preview and headed for Ludlow at lunchtime.

Wednesday August 12th

The big day, I've still got some names to think of and a high score update routine to write which I think I can cope with along with some more minor tuning. It's hard to think that I started this project before Christmas. Since then we have installed the PCs for doing the editing and assembling on, downloading the code for testing on the C128. We thought it would speed things up, which it did, but it just allowed me to write a much larger and more complex program, about 30K of code compared to Uridium's 18K or Alleykat's 20K. All these games have used all the C64's memory, the rest of the space being taken up by graphics, data, variable areas and buffers. The ratio has just switched to more code meaning that I've had to compress the graphics more. There are still nearly 350 sprite images in Morpheus, more than Alleykat and Uridium put together.

Personally I'm pleased with the result, it does many of the things I had dreamed of at the beginning, some ideas as always fell by the wayside, to be replaced by new ideas along the way.

Although Morpheus has a definite arcade quality look which is a logical progression of everything I've done before, it's not a 'five minute quick-blast' game. It contains a large planning ahead element which is at least as important to master as the control mode. I think you'll find this a game that will be played over many months. This will be the final diary entry and the game can be seen publicly at the PCW show and should be on sale in October. Who will be the first to build and maintain a ship capable of reaching and destroying level 50?

ZZAP! PREVIEW

MORPHEUS

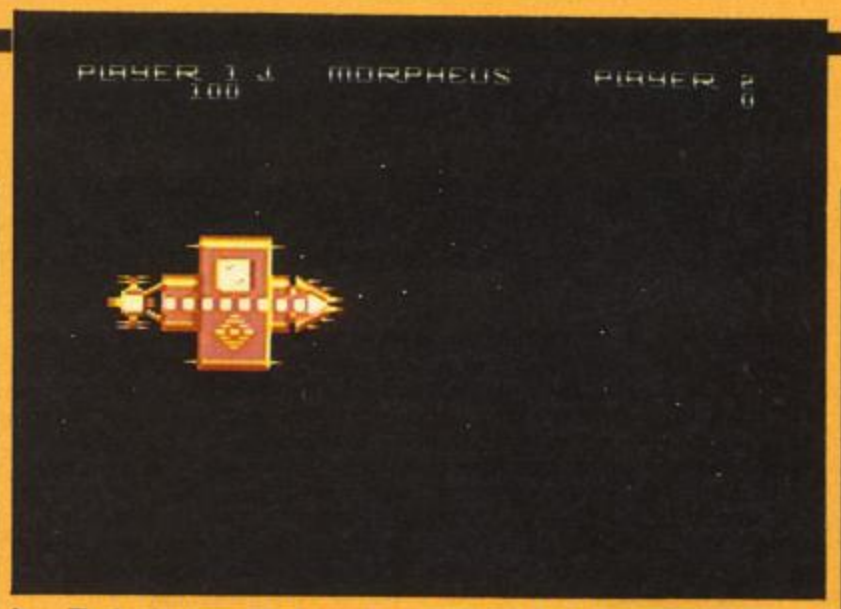
Hewson

At long, long last, after seven months of development, Andrew Braybrook has completed *Morpheus*. If you've been following his trials and tribulations over the past seven months, then you should have some idea of the game's concept. If you haven't, then here's a rough idea of what *Morpheus* is about . . .

The player takes control of an expandable ship, entering 50 different multi-directionally scrolling space sectors to do battle with a wide variety of aliens. The objective is to seek out and destroy a suspended alien power network consisting of 'charge orbitals'. Destroying the required amount of orbitals (this amount corresponds to the level's number) results in the shutdown of the central controlling nucleus, making the area safe again and allowing the player to continue to the next sector.

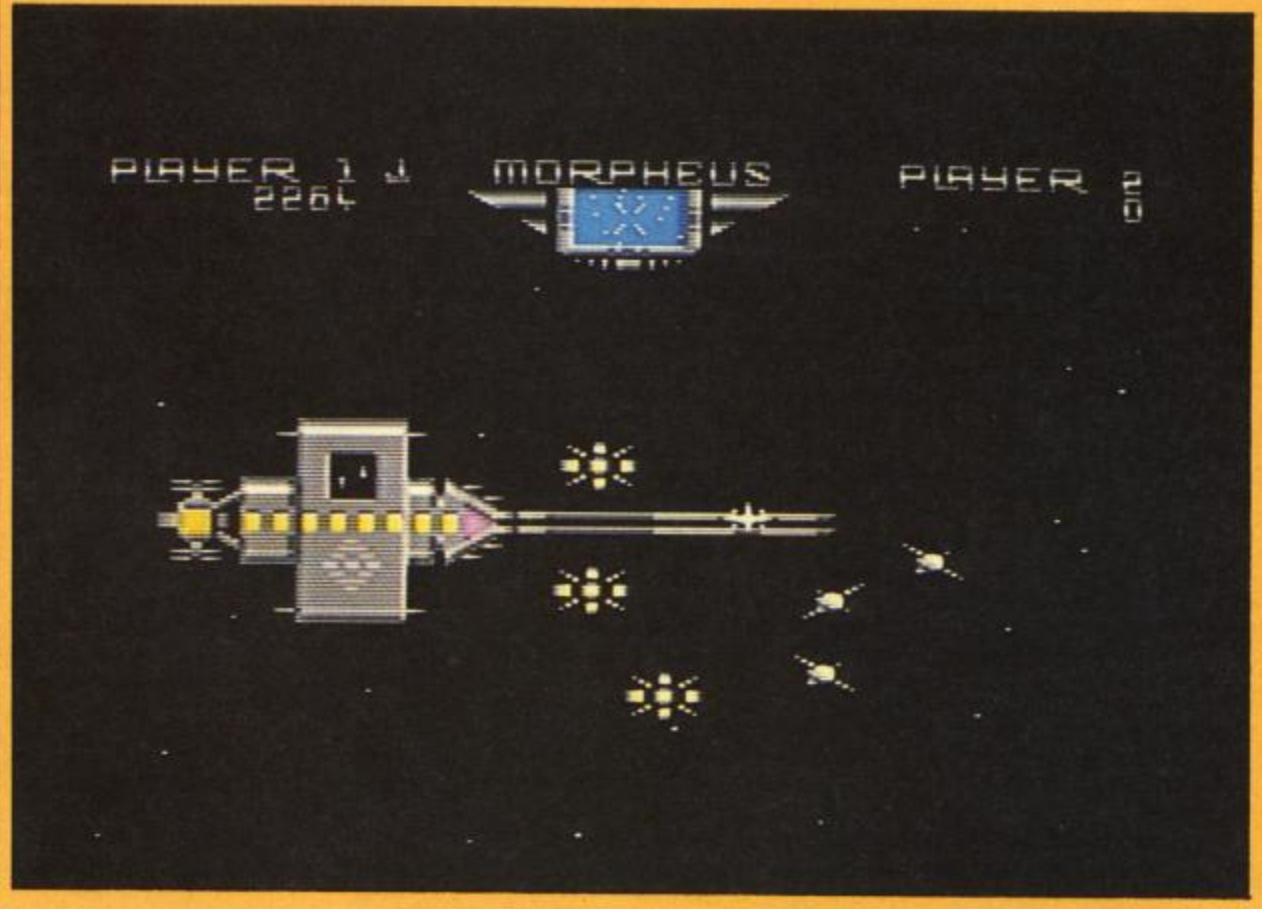
Naturally, there are alien ships guarding the network, and though these don't appear very aggressive at first – try wounding one of them and see how he reacts. As the game progresses, the aliens evolve and become more and more violent, and consequently capable of inflicting more damage on the ship.

Money is awarded for everything that is shot, and is saved and used to buy new, more modern ships or extra features for the current model. As higher levels are reached, it is essential that new machinery is bought, including shields, battery power-packs, inertia converters and other devices (both offensive and defensive), to maximize chances of survival.

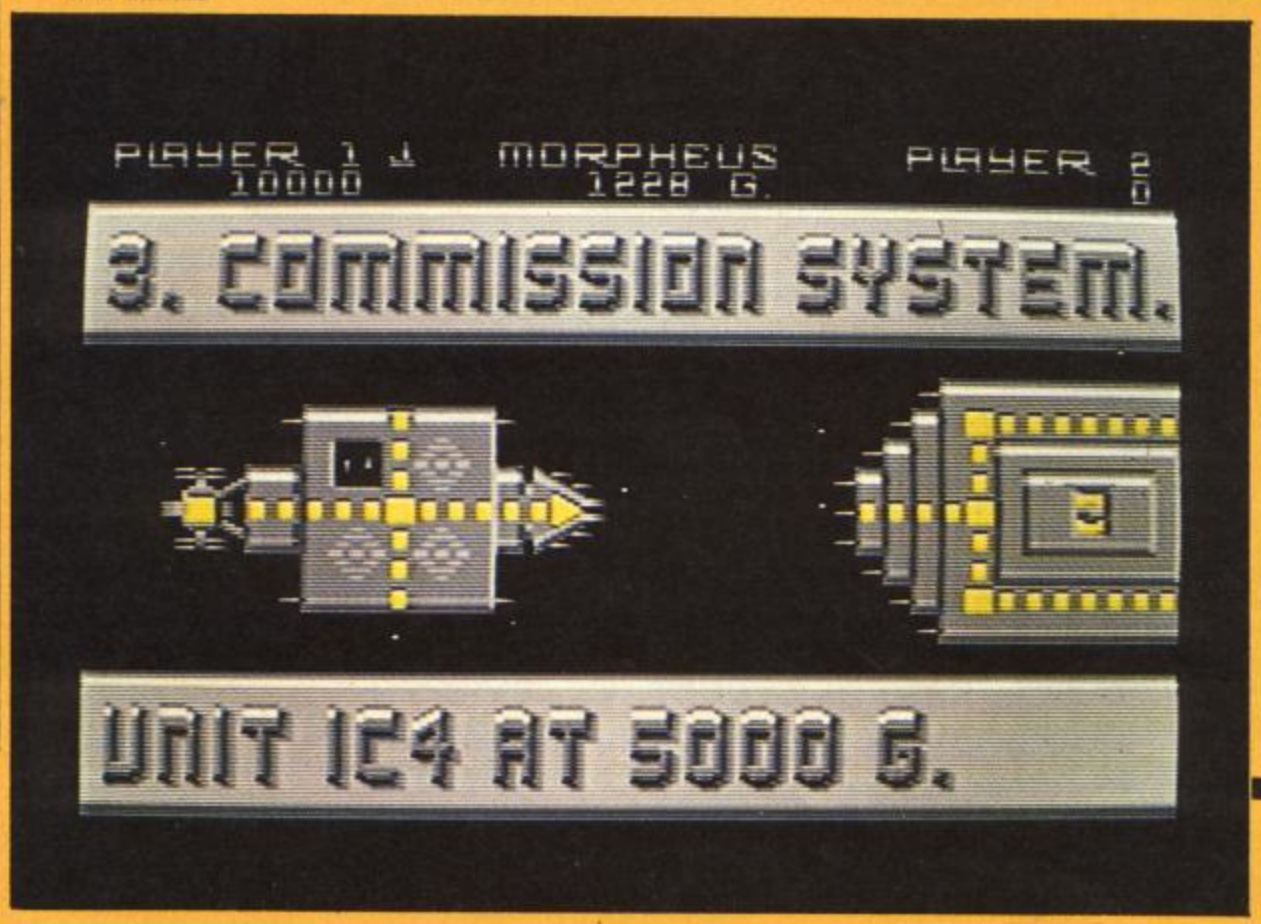


► **The basic, unmodified vessel just waiting to be customised with go-faster stripes and fluffy dice**

► **Using the giant tooth-paste weapon in Morpheus**



► **Commissioning the latest in weapons systems – a snip at 5000 Guineas**



Unlike other 'progressive' shoot 'em ups (such as Nemesis and Zynaps) extra weaponry and features are not just added to the ship. What sets Morpheus apart is the fact that extra equipment has to be commissioned – not simply added when you pick up a credit or icon. Also, the extra features are 'bolted on' to the ship, which means you actually need room on the side of you vessel for extra weapons and peripherals. It's not just a matter of killing the aliens, collecting the money and buying weapons and systems at random.

Morpheus has its own timescale – timeslices – and as time passes, the aliens become immune to older weapons. Therefore it is likely that a sensible player will progress further than one with quick reactions or an awesome fire rate.

The mission starts with the ship capable of carrying two devices, but if money is used wisely – and plenty of aliens are destroyed – an extremely large ship capable of carrying a wide variety of armament and features can gradually be purchased.

One nice feature is that the game automatically detects whether the machine it's being loaded into is a C128. If it is, an extra set of sprites are included into the gameplay.

Morpheus will be available in October, priced £8.95 on cassette, and £12.95 for the disk version. If you want more details, there'll be a full review next issue . . .

ZZAP! TEST

SIZZLER

MORPHEUS

Hewson, £9.95 cass, £12.95 disk, joystick only

● **After nine months in gestation, Andrew Braybrook's baby has finally arrived**

Morpheus features 50 sub-universes, or Aithers, each consisting of a central Nucleus surrounded by 32 Orbitals. The objective is to enter each Aither in a large spacecraft and shoot enough Orbitals to force the Nucleus to shut down. Just before the Nucleus dies, it spits out triangular Morphi which are destroyed for bonus points and money. The ultimate goal is to destroy the Morpheus on level 50.

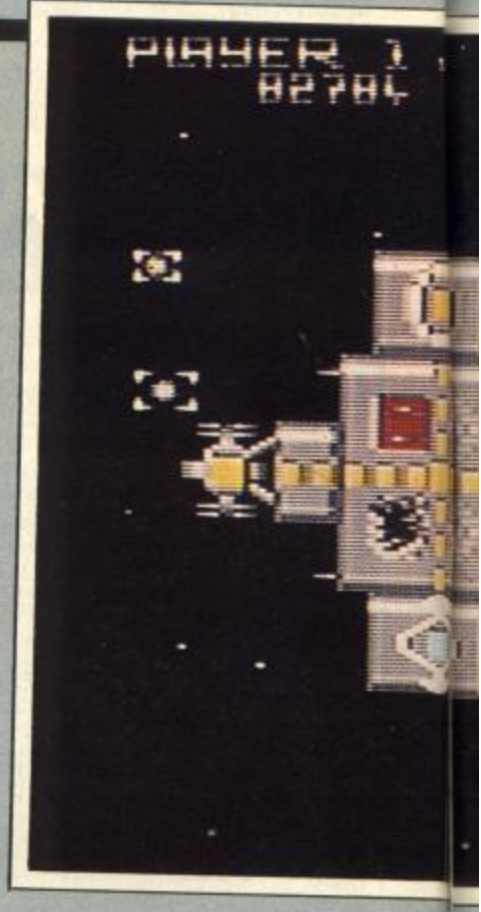
The mission begins with the ship docked at base, where a new hull may be purchased, or systems and weapons commissioned, scrapped or installed. Initially, the ship has no room for extra weapons and can accommodate only one extra system.

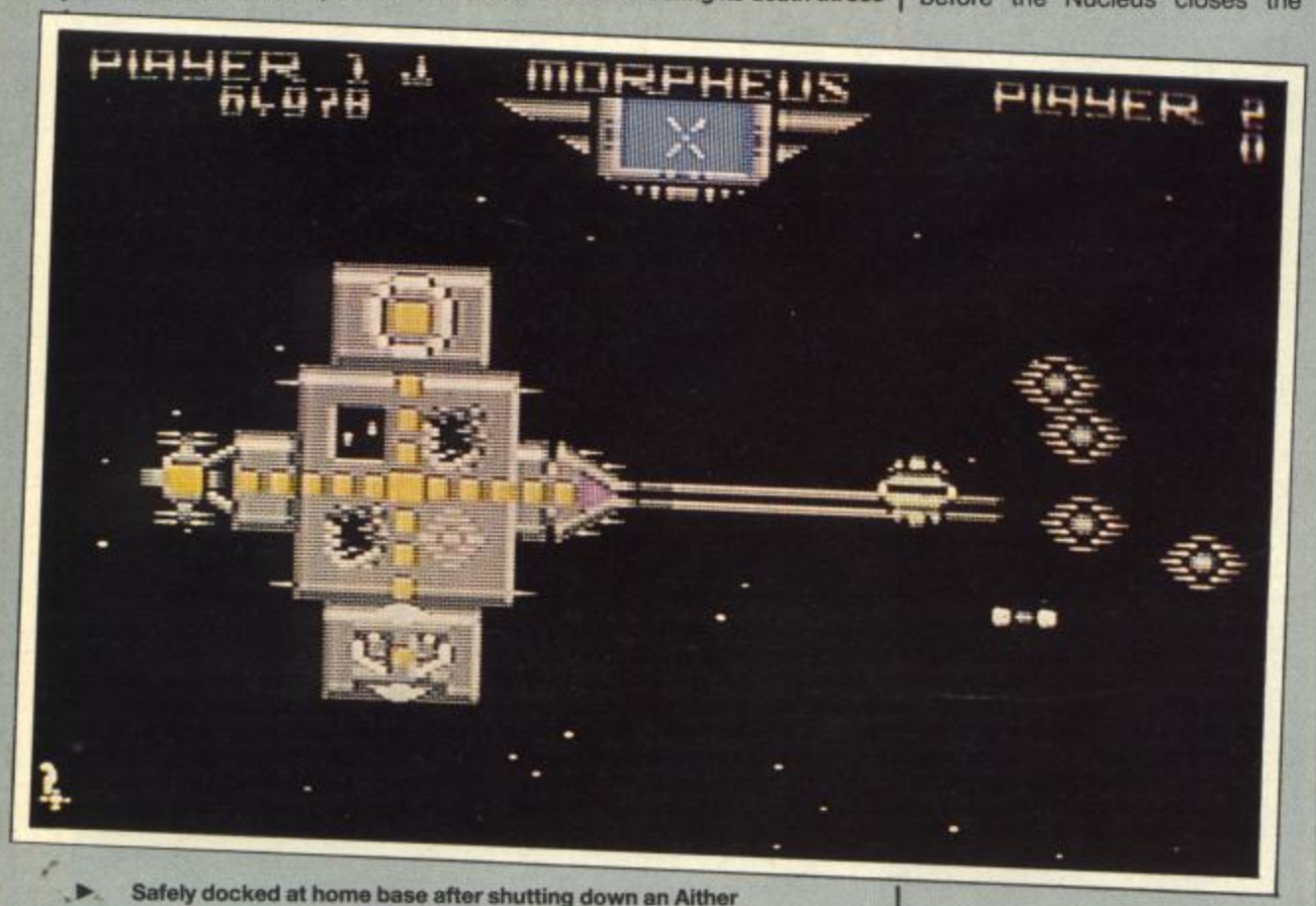
New weapons, hulls and systems are bought using the money amassed from killing Aithers, Morphii and aliens. Hulls are instantly accessible, but weapons and systems have to be ordered – forcing a craft into the fray while it's being built! Throughout the game, new systems and weapons are included into the inventory 'as they are developed'. The purchase of certain weapons is vital if the mission is to succeed. Others, however, are superficial – it's a case of working out what's best by trial and error.

Selecting the 'Deploy Ship' option and pressing the fire button launches the basic ship into the first Aither. The ship flies in any direction, with a parallax starfield showing movement. Above the main scrolling window is a scanner showing the location of Orbitals and the Nucleus in relation to the craft. A decaying Orbital is killed by lining the craft up and letting rip with a series of well-placed laser blasts. The number of Orbitals that must decay or be destroyed before the Nucleus closes the entire Aither down starts at one and increases to a maximum of ten as progress is made.

Morpheus also has its own timescale – measured in Timeslices, which are about two minutes long – and while hanging around in an Aither waiting for Orbitals to die of their own accord is one way of progressing, it isn't the route to success as no money is made.

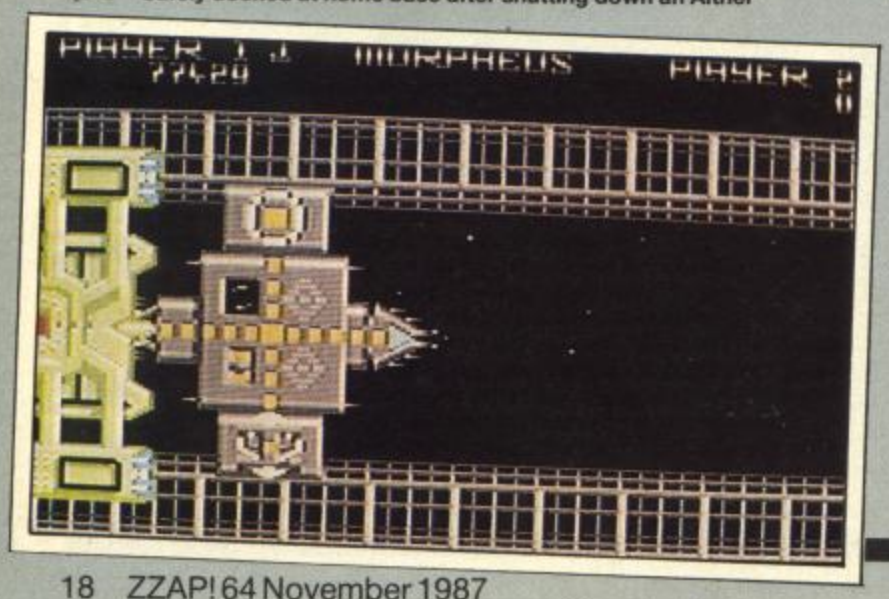
As time progresses, aliens become more aggressive and shoot faster and more accurately. Adversaries also become more intelligent and immune to weapons, so you have to keep buying the latest equipment – back at base the manufacturing plant produces increasingly more sophisticated items as game-time expires. Timeslice 50 marks the stage after which no further refine-





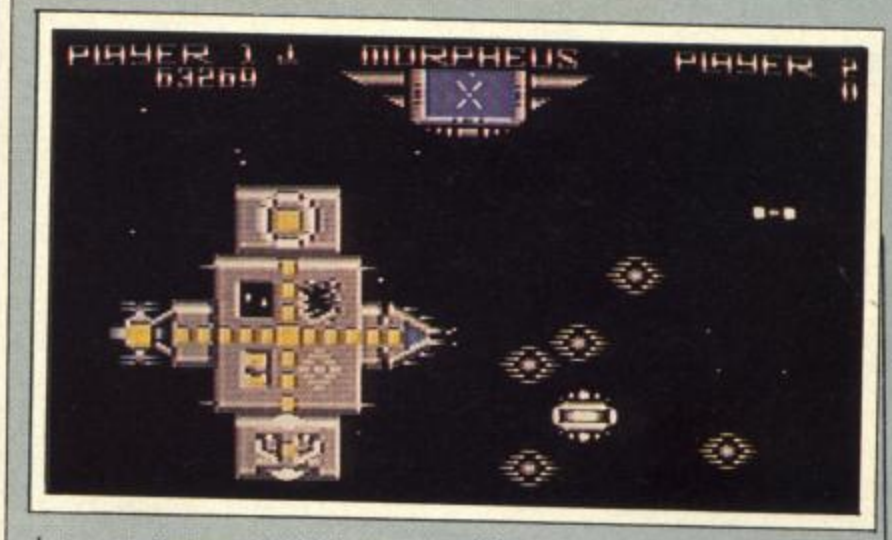
► **A blasted orbital spews out all manner of nasties during its death throes**

► **Safely docked at home base after shutting down an Aither**



No-one could ever accuse Andrew Braybrook of complacency when putting a game together as Morpheus **is without doubt one of the most finely constructed games ever written for the 64. Its graphic design and implementation are flawless, (the use of colour is phenomenal) and the work lavished on the piece is obvious merely from sight. The gameplay, however, is another story completely.** Morpheus **is no simple blasting game, (though admittedly the destruction of alien species does play a major part of the scenario), and it would be wrong to infer that brilliant design is a precursor to brilliant gameplay simply by default.** Morpheus **is fun to play; it's involved and should keep most people happy for a long time. But be warned – it's not like any game you've played before. Do your best to try it out before committing your tenner.**

► A considerably customised ship about to approach another orbital

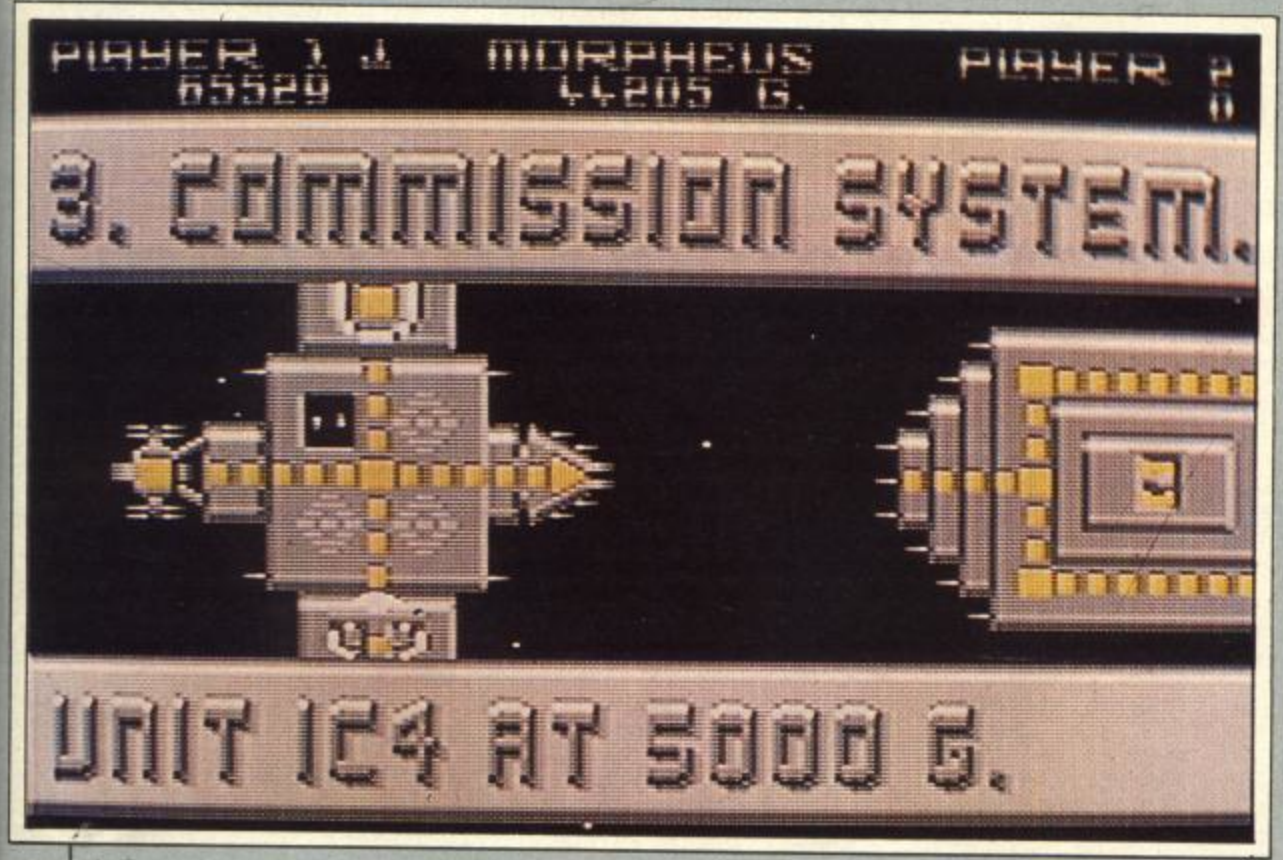


► For the less cosmically aware, positions of all orbitals are depicted using the blue radar screen

ments in technology takes place, and after Timeslice 60 you can no longer buy new weapons or systems.

The disk version of *Morpheus* features an 'All-Time Top Ten Greats' table which saves itself to disk for posterity and general gloating. Cassette owners just have to make do with 'Today's Greatest'.

Mr Braybrook has put an awful lot of work into his latest release – and it really shows. The care and attention which has been lavished on the superb graphics and atmospheric sound is matched only by the depth which has been instilled in the gameplay. In essence this is the game's main attraction – a gameplay which involves more strategy than shooting, and should consequently be providing pleasure long after an average game has been left to gather dust on the shelf. Morpheus is a thinking man's shoot 'em up – do yourself a favour and buy it as soon as you can.



► Time to commission a new unit. A grade four inertia converter for 5000 Guineas – what a bargain!

Fans of previous Braybrook games will be surprised by his latest program – it's a complete departure from the fast blast arcade games he's previously been associated with. All the usual Braybrook hallmarks are there though, including the superb presentation, graphics and sound, but Morpheus **is far more involved than any of his other offerings, and requires an awful lot of forethought, planning and trial and error to progress. The gameplay is deceptively simple – fly into space, blast the required orbitals, head for the nucleus and collect the morphi before it shuts down. This starts off easy, but starts getting really tough around level ten.** Morpheus **is a truly outstanding program, but unless you're really prepared to sit down and spend an awful lot of time working out which systems to buy at the right time you won't get a lot of satisfaction from it. I'm sure that many will get a great amount of pleasure from** Morpheus**, but I do stress that you try it out before buying.**

PRESENTATION 95%
Faultless in-game presentation.

GRAPHICS 93%
Superlative throughout, with gorgeous use of colour and wildly imaginative sprites.

SOUND 82%
Reasonable title tune and atmospheric effects.

HOOKABILITY 82%
The concept is tricky to grasp, but with practice it all becomes clear.

LASTABILITY 89%
Those willing to persevere have many months of exploration and experimentation ahead.

OVERALL 90%
A superlative program which could inspire a whole new style of game.