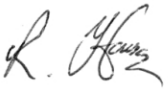
Success in Software

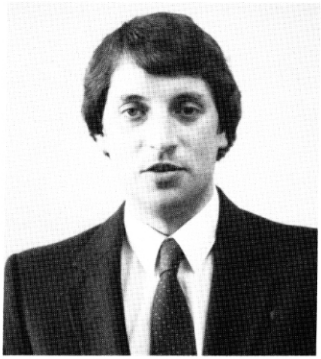
By Richard Hanson

Success in Software

With best wishes,



About the Author



Richard Hanson was born in Hull in 1959. He was educated at Hull Grammar School and The University of Leeds — graduating with a B.Sc. Honours Degree in Computational Science.

Whilst at university, he wrote several Acorn Atom and BBC Micro software titles which were marketed by Program Power (now called Micro Power) of Leeds.

Immediately after graduation in 1982, he set up the software house Superior Software. He wrote five of their early releases for the BBC Micro: *Galaxy Birds*, *Space Fighter*, *Centibug*, *Alien Dropout* and *Road Racer*.

In 1983, when Superior Software changed its status from a partnership into a limited company, Richard Hanson became the Managing Director of Superior Software Ltd.

He has extensive knowledge regarding both the technical and managerial aspects of software production and marketing. His experience ranges from computer programming (in several programming languages) to the administrative and financial control of software promotion and advertising.

Success in Software

Richard Hanson

Published by Superior Software Ltd

First published in 1987 by Superior Software Ltd, Regent House,
Skinner Lane, Leeds, LS7 1AX, England.

Designed and Produced by The Ellis Ives Sprowell Partnership,
Wetherby, West Yorkshire, England.

ISBN 1 870455 00 2.

Contents

Introduction

This book has been written to help you to write high-quality marketable computer software. A major objective has been that the book should be a guide for programmers — particularly games programmers — on all aspects of the now sophisticated home computer software market.

For newcomers, it explains the qualities that you will need if you are to become a successful games programmer. It describes the rewards that can be achieved, but warns of the pitfalls that await the unwary.

It tells you what to look for in choosing a software publisher, how to decide what type of program to write, and gives extensive advice on the software writing itself.

Most importantly, it explains how — once you have created your masterpiece — you can ensure that your program brings you the maximum possible financial rewards that your programming deserves.

We hope that after reading this guide you will know more about the deliberations involved in creating and marketing top-quality home computer software.

Can You Be a Successful Games Programmer?

The rewards are high indeed for top-class games programmers. Stories abound of teenagers who, simply by writing programs in their spare time whilst still at school, are earning more money than their fathers. Many of these tales are exaggerations; not all successful programmers live on caviare, own a new Porsche, and have a villa in southern Spain.

But several of Superior Software's authors certainly have reaped small fortunes from their work:

Tim Tyler, a 17-year-old from Exeter, wrote the acclaimed *Repton* and *Repton 2* games. They quickly rose to the top of the BBC Micro and Electron software charts, and within 18 months Tim has earnt £41,217-15 in royalty payments.

Peter Johnson, a young programmer from Newcastle, has written a number of games for Superior Software including *Deathstar* and *Overdrive*. His total earnings now stand at £36,593-46.

To be a top-class games programmer, what personal attributes are necessary? You do not need to be incredibly intelligent, nor do you need to be particularly artistic or creative. Here's a checklist of the most important requirements:-

1. **Plenty of Available Time**
   Assuming that you can work 30 hours a week, most games take at least two months to write. Some epics have needed over a year of programming time.
2. **Dedication and Hard Work**
   Before you begin to write a major program you must first thoroughly understand the programming language in which you are going to write the program, usually this will be assembly language. Also, it is equally important that you have an in-depth knowledge of the hardware that you are going to use. To amass this information it may be useful to build up a small library of books relating to subjects such as assembly language programming and technical guides to the hardware. A list of recommended books is given at the end of this section.

3. **Tenacity**
   You must be a limpet! It is a common fault among prospective games programmers that many of them tend to flit from one project to the next — never finishing anything off properly. Whilst it's often useful to explore new ideas or programming techniques, don't be distracted from the major project on which you're working. Bear in mind also that almost every game that has been released has needed some final touches. These amendments may literally make the difference between a chart-topping masterpiece and a run-of-the-mill game that collects dust on the computer dealer's shelves.

Now ask yourself the question: can I be a successful games programmer? Really the area of computer games programming is open to anyone. Academically some skill in mathematics is desirable, but you do not need to be able to mentally calculate the value of π to 99 decimal places. Drive and determination are important: if you are determined to succeed as a games programmer you should not be put off by any initial difficulties you may encounter. The key word is perseverance.

In-House or Freelance

Most home computer software authors are freelance and work from home — they are generally paid for their software by way of royalties or outright payments from a software house. (For details regarding financial arrangements, see the section on *Submitting Games*). Being freelance has certain advantages. You can work when you want to work: many programmers like to work late into the night and early morning — when they know that they won't be disturbed by visitors or phone-calls. If you are still at school or are involved in other employment, you are able to work part-time on your programming. But you must have sufficient self-discipline to ensure that you keep to the deadlines set by yourself or by a software house.

Some programmers work in-house at the premises of a software house. The main advantage is that you will receive an agreed salary from the software house, possibly with bonuses payable dependent upon the software you write.

Alternatively, a software house may agree to pay you a small salary (as a downpayment on future royalties) even if you are working from home.

(Note: The term "software house" is a commonly-used synonym for "software publishing company".)

Other Home Computer Software Occupations

1. **Games Designers**
   There is always a demand for new ideas for games. These ideas must be well thought-out and their manner of presentation is important: you should provide clear descriptions and documentation, with illustrations where necessary. Try to think of totally original concepts for the games. For further information on the subject of games design, read the next section *What Type of Programs Should You Write?*
2. **Composers of Computer Music**
   Most games have background music which plays whilst the game is running. It is becoming increasingly common for games to have loading soundtracks; that is, appropriate music which plays as the game program is being loaded into the computer. There is currently a dearth of composers of computer music. Good composers must have a musical bent (of course), the necessary creative skills, and an extensive knowledge of how to manipulate sound effects on specific computers.
3. **Graphics Designers**
   Graphics are an increasingly important factor in games. The graphics used to illustrate the main game characters and their surroundings must be designed with creative and artistic flair. Loading screens must be impressively attractive to give players something to enjoy as the game program is being loaded. There are now many Artist-type programs available which effectively provide designers with a blank sheet and a pallette of paints. Picasso would have had a field-day!

Recommended Books

1. **Assembly Language Programming**
   Z80 Assembly Language Programming by L. A. Leventhal (Osborne/McGraw-Hill, 1979)
   6502 Assembly Language Programming by L. A. Leventhal (Osborne/McGraw-Hill, 1979)
   6800 Assembly Language Programming by L. A. Leventhal (Osborne/McGraw-Hill, 1978)
   68000 Assembly Language Programming by L. A. Leventhal & G. Kane (Osborne/McGraw-Hill, 1986)
   Programming the 8086/8088 by J. Coffron (Sybex, 1983)
2. **Technical Guides to Specific Computers**
   The Complete Spectrum ROM Disassembly by I. Logan & F. O'Hara (Melbourne House, 1983)
   Advanced Spectrum Machine Language by D. Webb (Melbourne House, 1984)
   Commodore 64 Programmer's Reference Guide (SAMS, 83)
   Programming the Commodore 64 by R. West (Level Computer, 1985)
   The Anatomy of the Commodore 128 by K. Gerits, J. Schieb & F. Thrun (Data Becker, 1985)
   Commodore C16 User's Manual S. Finkel (SAMS, 1985)
   Complete C16 ROM Disassembly by P. Gerrard & K. Bergin (Duckworth, 1986)
   The Amiga Handbook by D. Lawrence & M. England (Sunshine, 1986)
   Amstrad Advanced Users Guide by D. Martin (Glentop, 1986)
   Amstrad 1512 PC Technical Reference Guide (Amstrad, 1986)
   Tricks and Tips of the Atari ST by R. Brückman, L. Englisch, J. Walkowiak & K. Gerits (Data Becker, 1986)
   The Atari ST Machine Language by B. Grohmann, P. Seidler and H. Slibar (Data Becker, 1986)
   The Advanced User Guide for the BBC Micro by A. Bray, A. Dickens & M. Holmes (Cambridge Microcomputer Centre, 1983)
   The Advanced Disk User Guide for the BBC Micro by C. Pharo (Cambridge Microcomputer Centre, 1985)
   A Hardware Guide for the BBC Micro by A. D. Derrick, D. S. Harding, S. D. Middleton & M. P. Smith (Wise Owl, 1983)
   Acorn Electron Advanced User Guide by M. Holmes & A. Dickens (Acornsoft Adder, 1984)
3. **Computer Graphics**
   Principles of Interactive Computer Graphics (Second Edition) by W. M. Newman & R. F. Sproull (McGraw-Hill, 1979)
   Computer Graphics and Applications by D. Harris (Chapman and Hall Computing, 1984)
4. **Data Manipulation**
   Fundamentals of Data Structures by E. Horowitz and S. Sahni (Pitman, 1977)
5. **Structured Programming**
   Data Structures and Program Design by R. Kruse (Prentice-Hall International, 1983)

All of the above-mentioned books are available by mail order from Computer Bookshops Ltd, 30 Lincoln Road, Oulton, Birmingham, B27 6PA. (Telephone: 021-707 5511).

What Type of Programs Should You Write?

So you think you can write a top-selling computer program? OK, you now have to decide:-

(a) which computer to use, and
(b) what program to write.

Games programmers generally develop preferences for particular computers. Some are attracted to the Commodore 64 because of its graphics and sound capabilities, others prefer to program the Amstrad 6128, whilst others enjoy the sheer challenge of producing good-quality games on the Spectrum. Your decision may be influenced by the software charts which are dominated by Spectrum and Commodore 64 software, but Amstrad, BBC Micro, Commodore 16 and Electron software also sells well. Do not forget the newer larger-memory computers such as the Atari ST, Commodore Amiga and Amstrad 1512 PC; these computers may be dramatically reduced in price in the near future, and their extra memory enables you to program extremely sophisticated games. The Amstrad 1512 PC is compatible with the IBM PC (and its clones); this means that there is potentially a vast market for PC-compatible software.

Whichever computer you decide upon, the use of discs for program storage is recommended. Many computers now have built-in disc drives, but if your's hasn't you may quickly come to detest the soporific nature of continual loading and saving with cassettes. If you feel that your hardware purchases may send your bank balance into the red and your bank-manager into a violent rage, have a word with a software house. If you convince them of your capabilities, they may be prepared to loan you some computer equipment for developing your software.

Let's assume that you have made up your mind with regard to the hardware which you are going to use. Your next course of action is deciding what program to write. There are two possibilities:-

(a) ask a software house to give you a specification for a program that they would like you to write, or
(b) design a game yourself — perhaps with the help of a friend or two.

If you contact a software house they will need to be sure that you're capable of writing a game for them, so you'll have to let them see some programs you've already written. They may then show you a game that is already being

programmed on one particular make of computer, and ask you to program the same game on another computer. Computers, like politicians, are notoriously incompatible with one another; but in some cases you may be able to transfer the program or graphics data from one computer to another and use these as a basis for the new version.

Alternatively, the software house may give you a specification to write a new game, perhaps one for which they have bought a publishing licence.

Licensing

Many of the recent successful games have been published under a licensing agreement. In these cases, the software house buys the rights to use either:-
(a) a particular game theme (for example: *Gauntlet*), or
(b) A particular well-known name (for example: Scooby Doo or Frank Bruno).
The software house feels that the use of the theme or name will give the computer game a strong marketing advantage.

When a game theme has been bought, the programmer will attempt to write a game that is as close to the original as possible. The hardware used by modern arcade game machines is considerably more sophisticated than the hardware of small home computers, but skilful programmers can generally produce surprisingly good copies of arcade games.

However, when a software house buys the rights to a well-known name, a games designer then has to sit down and create a game around the name. For example, J. R. R. Tolkien's book *The Hobbit* was used as the basis for a very successful adventure game for home computers. The game involves the characters and places mentioned in the book, and the player of the game has to skilfully plan his actions whilst collecting and using the objects which he finds around him.

Designing Computer Games

If you decide you would like to design a game yourself, here are some suggestions that you may find useful:-

1. Identify your target market. Making the game too strategic and less graphic will take it out of the mainstream games market. It may appeal to your aunt who can complete the crossword-puzzle in *The Times*, but teenagers generally prefer fast-action games.

2. Look at the software charts and consider the top-selling games. Ask yourself why they are doing so well. Watch out for new trends such as animated-fighting games, or the use of WIMPs (windows, icons, mice and pull-down menus).

3. Take a look at as many of the recently-released computer games as you can. Write down notes on any particular features that you think are impressive or innovative, and compile these snippets into a notebook for future consideration. Don't blatantly rip off other games, but try to use your notes as seeds which you must then nurture into a completely new game. Consider including humourous touches in your game — these games are supposed to be played for fun after all. Also try to think of some clever novelty features that will make your game stand out from all the others.

4. Game ideas can be based upon your work experience, your hobbies, interesting events that have occurred in your life, or an action sequence in a film or TV programme.

5. Once you have decided upon the basis for a game, you will then need to think about all the elements to go into it: the characters, the screens, the action, the puzzles. A very useful technique, which is known as "brainstorming", is to sit down with a large note pad and write down everything that comes into your mind as you think about the concepts, but without making any attempt to evaluate their usefulness. A thesaurus is a good source of further inspiration — let your mind drift from idea to idea, noting down any words or phrases which you think may be worthy of consideration later.

6. Sift through all your notes and assemble them into a meaningful whole by linking various ideas and thus gradually developing the game design. Always note down new ideas that suddenly spring to mind (even if you're in the bath at the time) otherwise you might forget the idea and the memory of even having had the idea!

7. If you have any doubts about whether your game idea is going to be a successful one, it is always best to ask a software house for their opinion before progressing further. Suppose you spent several months writing a program which, due perhaps to copyright considerations, the software houses refuse to release. That would be absolutely soul-destroying.

Full-Priced Software or Budget-Priced Software

It is worth considering whether you should aim your software at the full-priced software market or the budget-priced market. Budget-priced sotware usually retails at £1.99 or £2.99. Of course you will make less money on each sale of a budget-priced title, but you will be able to write the games more quickly because budget-priced games are generally not as good in terms of quality as the full-priced titles. This also means that if you feel you will never make the grade as a programmer of full-priced software, you may nevertheless be proficient enough to write a series of budget-priced software releases. However, bear in mind that some budget-priced releases are now as good a quality as their full-priced rivals. Despite the extra sales generated by the lower price, the programmer will usually lose out in terms of the total royalty payments he receives.

Recommended Books on Idea Generation

Lateral Thinking: A Textbook of Creativity by E. de Bono (Penguin, 1977)
De Bono's Thinking Course by E. de Bono (BBC, 1985)
Use Your Head by T. Buzan (BBC, 1982)

Writing Games

Getting Started

Before starting to write a game, it is generally worth your while to decide which software house you would like to market your game. (For information on choosing a software house, see the section on *Submitting Games*). You can then take the view that you are working in partnership with the software house — and this has several advantages. To save you re-inventing the wheel, you can establish what the software house can do for you. For example, the software house may be able to assist with software protection, turbo-loading, loading screens, music, etc. This will save some of your time, and it will put experts onto these specific jobs.

Additionally or alternatively, you can try to build up a software team yourself from amongst your friends. You might have an artistic friend, for example, who can help you with the graphics designs.

Attempt to estimate how long it is going to take for you to write the program. The programming may take a surprisingly long time, and it is important to try to plan ahead and schedule your work. If you are working to a deadline for a software house, it is vital that you tell them as soon as possible if your programming starts to fall behind schedule. The software house may be committing several thousands of pounds to advertising and promotional campaigns for your new program, and it is impossible to cancel advertising at short notice.

When you start to write the program, adhere to the following important rule: Before the start of each day's session of work, make a back-up copy of your program, and every week make a further back-up copy which you should keep separately in a safe place. Through no fault of your own — for example, a spike on the electricity mains supply — the computer might erase your entire program and corrupt the disc or cassette you are using. If this results in you losing a day's work, then at least you can start again from your back-up copy. If you do not keep back-up copies it is possible that you might lose a program which has taken you literally months to write. That would be devastating, and you may have to be restrained from throwing your computer through the window!

Structured Programming

An inexperienced programmer often starts to write major games programs in a very haphazard way. He may first decide to develop a routine which produces a stunning on-screen explosion effect: perhaps it's an effect that he was looking forward to programming and he may say, "This will be ideal for my new game's finale when the enemy's HQ is destroyed." Although he may have created a spectacular son-et-lumière display which may indeed be useful for his new program, he is not writing his program in a systematic manner.

Structured programming is one of the most useful skills that a programmer can learn. Basically, it's just common sense: Write your program by firstly writing the main control routine which sends almost all the actual work out to various subroutines. As you write the main routine decide exactly how the work will be assigned to the subroutines. When you come to write each subroutine you will know precisely what it is expected to do — but once again delegate the detailed work from each subroutine to further subroutines. Only when you reach the level of a subroutine that you have specified to perform a simple task, should you fill in the details by coding the particular task that you have designated. This approach is known as top-down refinement. It's not the only systematic method you can use, but it is a highly recommended one.

Game Coding

Use a good assembler or software development kit. Document your source code well so that changes can easily be made. This is very important, because if you return to some source code that you haven't seen for a few months you may suddenly discover that it is almost unintelligible to you. Try to be precise in your documentation — and never use cryptic comments.

Don't despair if you encounter a seemingly insurmountable programming problem. Try using a different approach to the particular programming task that you are trying to implement. If necessary, make compromises rather than scrapping the idea completely.

Software houses will usually be able to give technical help to their programmers — over the phone in most cases. So if you hit a major stumbling-block, assistance may be readily at hand at the other end of a phone line. But the technical experts at the software house will probably have a 1001 other things to do, so don't ring them excessively with minor technical queries.

Game Graphics

It is generally best if you write your own character designers enabling you to easily modify your characters if required. Commercially produced graphics packages could be used, but you may be able to devise more efficient storage methods for your graphics. For example, if you only need to use 6 bits of each byte for colour definitions instead of 8, you will be able to store about 33% more graphics by correct rotation of the bit patterns.

Loading Screen Graphics

Do not skimp on these graphics. How many times have you switched off the TV during the opening credits of a film or serial? In the same way you must stimulate and excite players, perhaps by using simple animation or colour changes whilst the game is being loaded.

Here graphics packages really come into their own. Recently, loading screen designs have improved tremendously and they are now often a work of art in themselves. This is a specialised area, and if you are not particularly artistic you are advised to ask a friend to help, or leave it to the software house to produce the loading screen for you.

Animation

Use as many frames as possible to animate each character. Four frames is really a minimum for walking in one direction. Make a dying man crumple elegantly to the floor; make him bend over and pick something up rather than have the object suddenly materialise in his hand.

Sound

As well as adding appropriate sound effects, considering including musical soundtracks which play:-

(a) when the game is being loaded into memory, and

(b) when the game is being played.

Some people, however, find sound effects distracting (particularly other members of the household); so always include sound on/off and music on/off options or, better still, a simple volume control: press a certain key to decrease the volume, and press an adjacent key to increase the volume. These considerations are very important if the computer you are programming produces its sound effects directly; they are not as important if the computer emits its sound effects via the television or monitor.

If you do not feel that music is your forte, this may be another job that the software house can deal with for you. Make sure you leave enough space in your program for the music routines and data.

Game Controls

Make the game as easy to operate as possible and avoid complex key operations. With some games, it seems as if you need to have at least three hands (or a pet octopus) in order to be able to use all the control keys. The best games require very few instructions for playing the game. Don't drop the player right into the thick of the action. Ease him gently into the game so that he can get used to the controls. Not every player is an arcade ace — many are young children.

It is best for the player to have the option to select joystick control or keyboard control — with user-selectable keys if possible. Some players do not have joysticks, others may prefer to use the keyboard.

Include a game pause control, causing your main character to crouch, sit or tap his foot, or do something completely out of the ordinary.

Special Features

Most computer games involve playing through several scenes and then moving on to the next — with each scene becoming progressively more difficult than the last. The first scene may be relatively easy, but this is not always the case. However, once the player has completed the first scene, the challenge then switches to "can I complete the second scene?" and so on until the entire game is completed or a new high-score is established. In some games it can be extremely frustrating if the player has to play through many scenes that he has already completed before he reaches the "challenge scene" — the one that he is currently desperately trying to finish.

So how can the frustrating stages be avoided? You can program into your game one of the following **"frustration-alleviation" techniques**:-

(1) (A poor method). Allow the players to select any scene by identifying each scene with a letter of the alphabet. At the start of the game, the player may select any scene by typing the appropriate letter . . . and the game then starts at the chosen scene. This method is the simplest of the four described here, but it is definitely not recommended because the game may quickly lose its appeal; the player can go straight to the last scene and he then knows the ultimate aim of the game. It is usually much better if the player doesn't know quite what's going to happen next.

(2) Incorporate a simple password feature into your game. When the player completes a scene, a password is shown briefly on the screen. When he starts the game again, the player can enter the password and start immediately on the scene after the one which he has already completed. You should, however, include a reward (perhaps an extra bonus scene) that can only be achieved if the player completes all the scenes from start to finish without using the passwords.

(3) Include a "jump" feature in your program. Once the player has completed the first scene, subsequently he may start directly on the next scene, and similarly for later screens. The scenes can be selected by letters as in method (1) but the player can only select a scene if he has completed all of the previous scenes. This method differs significantly from method (2) because each time the player starts a new session of playing the game, he has to begin again on the first screen and complete it before he is allowed to subsequently select the second scene.

(4) Save/Load Current Position. In certain types of game — usually adventure-type games — a "save current position" option is beneficial. Organise the variables so that only one or two blocks of data are saved out. If the game is sold on disc, allow the position to be saved to disc rather than cassette. This may seem obvious, but it is often overlooked: some disc owners simply do not have a cassette recorder.

Another important feature for most games is a high-score table. Players want to see that they are improving. Keep the number of entries to a maximum of 20 with sample scores included which increase in such a way that the player can see his skill developing as he slowly but surely improves his ranking in the table. The top position should be attainable but quite difficult to achieve. Allow at least 8 characters — more if possible — for name input. You can also include sample names, perhaps psuedo-military ranks.

During game play, you could display a special message for a score of 10000, another at 20000 and so on. Or consider using graphical effects: a simple change of colour at very high levels can be immensely stimulating and rewarding, especially as the player realises that he is seeing something that very few other people will ever see.

Intermediate bonus scenes in between the main scenes of the game make an interesting diversion in some games. It is better if you make the intermediate scene as different as possible — probably requiring different skills and controls — to the main scenes.

Some of the most important special features have been mentioned previously: a game pause control (so that players can leave the game to go and answer the phone), joystick/keyboard options, sound on/off controls, etc. These are also some of the easiest and most economic features to implement so no game should be without them.

Checklist of Important Game Features

Here's a checklist of important features that should be included in most games:-

(1) "Frustration-alleviation" techniques.
(2) High-score table — with sample scores and names.
(3) Special messages or effects when high scores are achieved.
(4) Game pause control.
(5) Game abort control.
(6) Joystick/keyboard options.
(7) User-definable keys.
(8) Sound on/off, music on/off or volume control.
(9) Demonstration mode — this is a very important selling point when games are displayed playing automatically in shops.
(10) On-screen information — such as: score, lives remaining, scene number, fuel/energy level, and warnings.
(11) Humourous touches — keep the player entertained.

Finishing Touches

Include a suitable pause between each life lost and game-over messages so that the player knows what has hit him, so to speak. But allow the player to move on to the next screen immediately if, for example, he presses the space bar. Allow the player to abort a game easily if he is doing badly, and don't make him wait 30 seconds before he can restart because you have insisted on inserting a nifty animation routine at the start of every game.

Bug Checking and Game Testing

The importance of bug checking cannot be over-emphasised. The computer magazines' game reviewers are generally extremely skilful players who will track down any remaining bugs in a program. If they do discover a major bug in your program it may signal the death knell for your game because the bad publicity caused by their reviews will put off many a prospective customer.

Continuously as you are writing your program you should try to ensure that all your routines are bug-free. Always check the extreme or unusual values that may be fed into each routine. Try to check all possible routes through the game. When you have to make amendments to a routine, make sure that the changes do not disturb the rest of the program.

Periodically, as you are writing the program, get someone else to play the game without your supervision, and allow him to do daft things with the game. Ask people who do not know you well, as they will be more blunt and honest about your program. You may be surprised how many new bugs are uncovered. Watch out for easier ways to solve your difficult puzzles. It may be possible to complete the game by finding trick ways through the game, or by solving puzzles in a different way.

Are the testers having fun? Do other people stop what they are doing and crowd round the display, or do they take a quick look and then wander off? A good game should draw the crowds like the Pied Piper of Hamlyn. Note down any comments — don't rely on your memory — then retire to your computer and make improvements.

Compatibility

Check the game on different versions of your computer, and with all the popular peripherals. Avoid using "illegal" techniques which, although clever, may mean that your program will not work on new versions of your computer. Check your code carefully — due to poor editing the game *Repton* contained the following 6502 machine-code: JSR:NOP:NOP. This caused a JUMP to &EAEA which RTSed without any problems on the standard BBC Micro. When the same program was run on a BBC Master Series computer the game crashed.

The Use of Game-Creating Software

Game Creating Software is basically software which takes away some of the drudgery of writing either entire games or certain parts of games.

Two well-known commercially available programs are *The Quill* and *The Graphics Adventure Creator*. These programs can be used to construct adventure games complete with screen graphics. (You will need to also use *The Illustrator* program to achieve screen graphics when using *The Quill*). If you are considering selling a game that you have created by using one of these types of program, you must carefully read the documentation supplied with the programs in order to establish the copyright position. In any case, be fair and include a credit for the game-creating software you have used.

You can, of course, always write your own game-creating software if you feel that using such software will save your time in the long run. This may be particularly appropriate for creating part of a game rather than the entire game.

Are You Under-equipped?

One problem that programmers quickly experience is "how to fit a quart into a pint pot?" Let's suppose that you have drawn up a memory map for a prospective game: the computer's operating system will take up or use a sizeable amount of the computer's memory, the graphics screen area uses another chunk of the memory, and the rest of the memory is available for your program and data. Let's say that there is 16K of memory available for your program. You will probably find that you want to fill virtually the entire 16K — because you will want your game to have as many features as possible.

Now, the problem is: where can you store your assembler source code whilst it is being assembled into the computer's machine-code? The first solution is to store it in the memory spare allocated to the graphics screen area whilst you are using the more economic text screen area. This is usually acceptable, except that in order to assemble 16K of machine-code you will need to load a sequence of several files of source code into the graphics screen memory one at a time. So, a better solution, where possible, is to buy extra add-on memory for the computer — you can then use this memory space for the assembler source code. For example, for the BBC Micro you can buy a 2nd-processor which includes 64K of additional memory. In the long term, the time saved will be considerable.

If you are serious about making home computer programming your profession, consider using the earnings you make from your published software to buy extra computer equipment.

Submitting Games

If you have written a game which you have designed yourself, you will need to find a suitable software publisher. You could publish the game yourself; but the software market is now rather sophisticated and unless you have an astute business backer with the money to pay for expensive colour advertising, packaging and public relations, you would be well advised to leave the marketing to the professionals. Typically, a single page advertisement in the leading computer magazines costs more than £1000!

So which software house should you choose to publish your game? There are several major factors to consider:-

1. **Financial Arrangements**
   There are three basic types of payments that software house, may make to authors of games:-
   (a) An outright payment. An agreed sum (usually over £1000) is paid to the author for the rights to sell his program.
   (b) Royalty payments. The software house agrees to pay the author a fixed amount (normally between 10p and 80p) for every copy of the program that they sell.
   (c) A downpayment. The software house agrees to pay royalty fees (as described above) but also makes a downpayment (typically £200 to £1000) on the royalties. This sum is deducted from the royalties before any further payments are made.

The financial returns you can expect from a game obviously vary widely from one game to another. But it is usually the best policy to take royalty payments if they are offered, or ask for royalty payments if they are not. This way you will benefit if your game sells really well. An outright payment will often be a compromise based on the company's other games. Don't accept any percentage based on "profits" — this is too vague a term. You should really be getting about 80p per copy sold for an original title retailing at £9.95. Be prepared to accept a lower royalty if licensing is involved but expect much higher sales.

2. **Helpful and Efficient Service**
   When you send your program to a software house for evaluation, you must expect to wait a few days before they get in touch with you — it takes time to properly evaluate each program they receive. But programmers have been kept waiting for many months in some cases; any software house that treats you in this way should be rejected — they'll probably keep you waiting for your royalty payments as well.

   You can also assess a software house by the response you get if you phone them with a query of some sort. Are they friendly and helpful, or do you get the impression that they just don't care?

3. **The Size of the Software House is Important**
   A small software house may not be able to market your program to its maximum potential. The advertising and promotional budget for a major new software release is often well over £20000.

   The large software houses will certainly be able to afford to promote your game properly. But some of the larger companies are now releasing more than 50 games a year; therefore, your program may easily get lost among a mass of other new releases.

   To get the best of both worlds, it is probably best to select a successful medium-sized software house.

4. **Marketing Abilities**
   Check through the various home computer magazines and note down those software houses whose advertising impresses you. Look for companies who:-
   (a) produce quality advertising with clever promotional text,
   (b) advertise regularly and take prominent advertising space (that is, front and back cover positions),
   (c) get good editorial coverage on the news pages, in the reviews sections, and on television programmes.

5. **What Do Other Programmers Say?**
   If you know any programmers who have had games published by software houses, ask their opinions regarding the companies they have dealt with. They may be able to steer you clear of some shady companies and recommend other firms that have given them a fair deal.

Presenting the Game and Presenting Yourself

Firstly, do check that your game is completely bug-free, and also remember to check carefully for spelling mistakes in the screen displays. For some inexplicable reason, it is a common occurrence that some of the best computer programmers are notoriously bad at spelling.

Record your finished program onto a new disc or cassette and mark it clearly with the name of the program, the computer used, and your name, address and telephone number. Software evaluators are very busy people, so it is always advisable to supply your program on disc if possible. Also, don't send a disc or cassette that looks as if it has been retrieved from the dog after he'd been chewing it. This will put off the software house straightaway.

Most software evaluators are completely trustworthy, but regrettably there are knaves in all professions; so you might consider recording a dated copy of your software and lodging this with your bank at the same time that you send a copy to the software house for evaluation. Another possibility is to make a video recording of your game in action and send this to the software house, rather than sending the actual game itself.

Supply detailed documentation with your game. It will make the software evaluator's task much easier if you can provide him with concise maps and solutions. Also, consider including a "cheat" option in your game to enable the evaluator to immediately move on to different sections of the game. And don't forget to mention the loading instructions for the program.

Try to give your package a finished presentation. You could even quickly produce a mock-up cover for the disc or cassette. Remember that you are attempting to sell the game to the software house.

Before sending your masterpiece to the software house, make a final check that the disc or cassette loads correctly. When you are satisfied that all is well, send the game and documentation to the software house in a padded "jiffy bag".

If the software house decides to publish your game, they will try to generate as much editorial about the game and the author as possible. To assist them, provide photos and information about yourself and the games you have written. You can also attempt to raise your own publicity by sending photos and personal information direct to the editors of the computer magazines. This publicity will help to boost the sales of your software by making more people take notice of you.

An important point of professionalism: Don't be tempted to give copies — particularly unprotected copies — of your new game to anyone. It has been known for unreleased programs to become circulated around the country when dishonest people copy the programs and pass these copies on to other people.

Enhancements to the Game

If the software house accepts your game, be prepared to make some changes to the program. The software evaluators are usually experienced professionals and they may provide you with a large list of amendments that they would like you to make to the game. Most of these amendments — apart from any bugs discovered — should be quite straightforward to incorporate, perhaps just involving a change of name, or some new character designs.

You should actually welcome the suggestions since they could ultimately greatly increase the sales of your game. It is quite feasible that 10% of extra time spent on your program could result in you earning 50% more money from the game.

When you submit your game to the software house, it may be useful to include a list of possible amendments that you have yourself considered making to the game. Also mention enhancements that could be made to the game for use on larger memory computers. For example, a 48K Spectrum game can be significantly improved if it is to be also released in the form of a 128K Spectrum version. Similarly, standard BBC Micro games can be greatly enhanced by making use of the extra memory available on the BBC Master Series computers.

Many good games have never been released simply because the programmer was not prepared to make simple improvements to the game. If you are unsure of any major changes being requested, contact the software house and discuss your reservations. Perhaps an alternative solution may be possible.

It is worthwhile reiterating that when you make amendments to a part of your program, try not to disturb the rest of the program at all if possible.

Legal Agreements

Find out when the software house intends to release your game. Launching new programs takes time and is strategic: promotional artwork and advertisements have to be prepared, review copies must be sent out, editorial features need to be obtained in magazines, the software distributors have to be informed well in advance. In order to obtain the best exposure, your program may need to be slotted in at the optimum time among a series of other new titles. So rely on the software house's judgement as to the timing of the release of your game.

But insist on having a contract straightaway and check that the written contract agrees with the verbal assurances you have been given. For the written contract to be legally binding, if you are under 18 you must ask one of your parents (or guardian) to sign the contract as well as signing it yourself.

Check carefully the extent of the territory of the contract: are you signing away the World rights or just the U.K. rights? Check also whether you would be giving the software house the full commercial exploitation rights for your program to be produced in the form of versions for different computers. If you have devised an original name for your game, check whether you would be signing over to the software house the right to use this name (perhaps for a sequel to your game).

Be wary of signing any contract that commits you to writing a series of programs in the future for the software house. The software house might decide to pay you a lower royalty on a subsequent program than you could obtain elsewhere, or they may accept the program but never actually release it — this has actually happened on a few occasions.

If in any doubt about the legalities of a contract, it is best to consult a solicitor before you sign anything.

After Your Game Has Been Released

Check the advertising and software packages in the shops to ensure that the selling price is as expected. Look out for reviews of your game in the home computer magazines; reviews are extremely important in influencing the sales of programs, but don't take critical comments personally. Reviews are intrinsically subjective, and even the best of games may receive a bad review amongst a series of otherwise excellent ones.

Watch for your game entering the software charts — the most reliable charts are compiled by Gallup Chart Services and published in several magazines as well as appearing on Channel 4's Oracle service. A high chart position is a strong influence on future sales of your game.

If you are being paid by way of royalties, check when the royalty payments are due to be made. If you have not received your royalties within a month of the due date, contact the software house for an explanation. Just in case of discrepancies, don't spend the money you expect to receive until the royalty cheque actually arrives.

The life of a software title varies greatly. It is dependent upon the type of program and the popularity it achieves. Most programs will make the vast majority of their sales within the first two months of release; the sales will then gradually dwindle down over the next few months. Utility software usually has a longer life-span than games. However, games that come to be regarded as classics may continue to sell well for a year or more.

Taxation Considerations

If you have been unemployed for several weeks before you started to earn money from your software writing, you may be eligible for a government grant. You might need to put forward some capital yourself but, unless you already have other employment, it's worth obtaining details about grant entitlement.

Let's suppose that, after considerable dedication and hard work, your game is a great success; after just a few months your program has earnt you several thousands of pounds. Remember, though, that you must legally declare this income to the Inland Revenue on your annual income tax return. This income will be taxed over and above any other income that you may have earnt. The software house is legally obliged to inform the Inland Revenue of the name and address of anyone to whom they have paid royalties.

If you find that your software writing is bringing in a considerable amount of money for you, it may be beneficial for you to enlist the services of a good professional accountant. Your bank-manager will be able to recommend some accountants to you.

The accountant will advise you on the following subjects:-

(a) How much money you should keep aside for paying income tax.

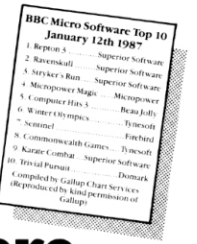
(b) Whether you are able to offset some of the tax against the money you have spent on computer equipment, software, magazines, travelling and telephone expenses. (It is essential that you obtain and keep receipts for any such purchases you make).

(c) Whether it would benefit you financially to be registered for V.A.T. — this may be a legal necessity in certain circumstances.

(d) Whether it would be financially advantageous for you to form a limited company.

Although accountants' fees may seem to be fairly high, accountants claim that they can generally save you more money than you are paying them because they know how to effectively minimise your tax liability.

You might, of course, find that your software writing is making you an absolute fortune. In this case, the services of a good professional accountant are essential. The rest is up to you . . .



A Brief History of Superior Software

Superior Software was established in 1982 by Richard Hanson, and the company's first release was *Galaxy Birds* for the BBC Micro in October 1982.

Superior Software achieved major success in 1983 with the BBC Micro version of the arcade game *Hunchback* — which sold over 30,000 copies.

The company has now published over 100 software titles, and is unquestionably the leading software house for the BBC Micro and Acorn Electron.

Recent chart-topping programs have included *Repton, Thrust, Karate Combat, Stryker's Run, Citadel, Ravenskull* and *SPEECH!* Additionally, *Repton 3* and *SPEECH!* have been successful releases for the Commodore amd Amstrad computers.

Increasingly, Superior Software is looking to consolidate its position as top BBC Micro and Acorn Electron software publisher by releasing new software across the range of home computers: Spectrum, Commodore 64/128, Amstrad 464/664/6128, Commodore 16/+4, Atari ST, Amstrad 1512 PC and Commodore Amiga.

Future programs under development across this range include the exciting new games *By Fair Means or Foul, Nautilus, Haunted House* and *Elixir.*

Index

SUCCESS IN SOFTWARE is the definitive reference guide for all aspiring home computer programmers.

The author, as a games enthusiast, programmer and Managing Director of Superior Software, is able to give a bird's-eye view of the paths to success and failure in the software industry.

His advice is based upon personal experience; and subjects ranging from games ideas to V.A.T. considerations are covered in great detail. Stage by stage you will discover how to produce top-quality games, choose a software publisher, and then maximise your earnings.

An invaluable reference book for software authors.

SUPERIOR SOFTWARE Limited
Regent House, Skinner Lane, Leeds LS7 1AX.
Tel: (0532) 459453

ISBN 1-870-455-00-2

9 781870 455008

Price: £1.95