A Guide For The Shareware Developer : 8/24/91
update: 4/17/93

If you've written what you thought was a great piece of shareware, had hundreds of downloads, but never made a nickel, this article is for you. If you've written shareware that nobody was interested in, this article's for you, too. In short, this article is for anybody who has tried, or would like to try, the shareware market.

In order to market shareware (or any product) you must follow some basic rules: understand the market; target a segment of the market; develop product that suits the targeted segment and stick to it; realistically appraise your product's life-cycle; and support your product. Additionally, there are a several rules that pertain specifically to shareware: don't expect anyone to pay for something they already have in their possession; provide real inducements to pay for your product; and give your product polish.

Understand the Market:

Lots of good shareware fails to be profitable because it isn't targeted for the right market segment. The fact is, lots of shareware developers don't target a market segment at all. They assume their product will have mass appeal, then wonder why they aren't making any money from it. The only product I'm aware of that has had universal acceptance in the shareware industry is PKZIP. Software that is written to have mass appeal is usually a turn-off to the market segment you need to appeal to the most. Conversely, shareware that's written to appeal to a specific market segment is doomed to fail if that targeted market segment doesn't typically utilize shareware.

Target the Market Segment:

There are two types of PC users: home and business. The home PC market consists of low-end, intermediate low-end, mid-range and high-end classes. The business market consists of small and large business users. You can develop shareware products that will appeal to small business users, but it won't be easy. The small business is likely to put his trust (and business) in the hands of a local consultant. They're going to buy the products their consultant pushes. Targeting shareware for large business is a waste of time. It's extremely unlikely that the person who downloads and uses the shareware is the one who pays the bills. Many large companies even have policies against downloading any shareware.

The low-end home market segment consists of AT, 286 and smaller 386 class systems. There are typically a lot of teenagers in this group, so games can do pretty well. Unfortunately, this market segment has the least money to spend, and the least inclination to pay for shareware. This is the market segment you're likely to appeal to most if you don't specifically target your software elsewhere.

The intermediate low-end consists of higher end 386 machines; the 386/33 or 386/40 or smaller 386s that have a lot of add-ons (graphics card, memory, high-end printer, etc.). This is a good market to target, especially for products that will extend the life or usefulness of these systems. This market segment tries a lot of software out to see what will work best on their system. They'll pay for shareware that fits.

The mid-range market segment consists of high-end 386 (33 or 40 megahertz with lots of add-ons) and most 486 systems. This is probably the best market segment to target. It means VGA (or better) graphics, sound cards, adequate memory, decent size hard-drives and maybe multi-media. This market segment typically looks for products that showcase their system features. Conversely, products that support CGA or EGA graphics (for example) are a turn-off to this segment. If your smart, you'll develop your product specifically to appeal to this level.

The high-end market segment consists of the 486/50 and 66 systems with all the "bells and whistles". We're talking the 5000.00 - 15,000.00 PC systems here. A bad market to target. Nothing but the best, most expensive software goes on these babies; no "stinkin" shareware!

There is one other market segment that should be mentioned, the consultants and developers. This is a good market segment for diagnostic tools and utilities that make their jobs easier. Consultants and system developers are consistently conscientious about paying for shareware and often will carry shareware to their customer sites.

Develop Product That Suits the Targeted Segment:

Once you've picked your target market segment, design products that specifically fit the target. You can't sell a game that supports EGA graphics to a 486/33 user with local bus video, and vice-versa. Likewise, if

compromise your product to make it fit more than one market segment , you'll loose all market segments to products that are better fits. We used to have a saying: "don't piss in the soup"; which is very appropriate.

Realistically Appraise Your Product's Life Cycle:

There are damn few shareware products on the market that have a life cycle that can't be measured in months, some even in days; unless you've created the definitive product (such as PKZIP or NeoPaint). What is a 'life cycle" anyway? Simply stated , it's how long can you expect to receive registration fees before the market segment loses interest or your product is superseded by a similar but superior product from another vendor. A short "life-cycle" isn't necessarily bad; it just means you have to adjust your marketing philosophy. For instance, you can plan frequent upgrades to keep your product "fresh". One thing you don't do is offer 30 day free trial period for a product that's got a "life-cycle" of 25 days.

When determining "life-cycle", measure only the registration fees you get from your "home" BBS. All other fees are the results of "trickle-down" and are meaningless in determining your product is due for a "freshening up". The nature of your product will also determine "life-cycle". Games typically have a peak when first introduced , which drops off fairly quickly. Small shareware products tend to have a somewhat more gradual curve. Large shareware products tend to be the most gradual sellers, and last the longest. Plan your enhancements accordingly.

Support Your Product:

There's nothing worse for a sale than a poorly supported piece of software or shareware. If a potential customer has a problem, and can't contact you expediently, you've lost that sale and any other sales that may have been created by "trickle-down". The ideal level of support is a fully staffed "800" support number. Obviously, this would be cost prohibitive even for the most prominent shareware vendors. You could provide a Compuserve number, or a mailbox on your "home" BBS. My choice is to schedule specific hours each day on my own PC to act as my own BBS. This affords me the opportunity to plug other products I offer as well as offering convenient downloading of same.

Only you can decide how much support your product requires or is worth. Only you can determine what is an acceptable level of lost sales due to lack of adequate support.

Don't Expect People to Pay For What They Have In Their Possession:

Here's the trickiest part about shareware development; how do you get payment for your hard work? Consider this: the shareware concept gives you free marketing and distribution that probably reaches at least 25,000 people, and provides a ready pool of people willing to try out your product. A good shareware product will probably reach hundreds of thousands of people, with the "trickle-down" effect. It's a marketer's dream! But, you've got to make people want to pay for your product.

Many shareware developers create a fully functional product, distribute it as shareware and accompany it with a letter asking for payment. Others program gentle reminders in their fully functional product. More often than not, neither of these methods works (except possibly for Neopaint).

Most people don't download shareware initially with the intent of stealing it. It just happens this way. The fault is that everyone has a limited amount of money to spend on computer products and "goodies"; why should they pay for your's now if they don't have to? They can use that money to buy something else and keep on using your product until they get some extra money. Well, that day never comes! You have got to make them want to pay for your product now, and postpone buying that other product.

Polish Your Product:

A good part of getting paid for product lies in perceived value. It doesn't matter how well your product performs, if it looks cheap and has a bad tactile feel to it, people will perceive it as not worth much. If it's difficult to use, it will be perceived as not worth much. Worse yet, someone else will steal your idea, polish it up, add some "bells and whistles" and reap the benefits. So, give it pretty graphics, give it windows and boxes, give it mouse support and give it menu selection. The only "ugly" product I've ever seen succeed is PKZIP, and that used an entirely unique marketing philosophy (give it to everyone except businesses for free). If people perceive your product as being expensive and professional, they're more likely to pay for it and more likely to consider it worth their while to keep you in business to see your subsequent product offerings. Damn few of us don't want to support the vendor that consistently turns out good quality software.

Provide Payment Inducements:

For game developers, providing payment inducements is pretty easy, providing you've got a good game. Just follow the example set by the developers of Wolfenstein, the Abyss, or Redhook's Revenge, and give the customer a fully functional first round, with the promise of lots more fun in the registered version. Most any game can do this, if it's part of the initial development and planning phase.

For software product developers, it's a lot tougher. Making a product fully functional except for some critical feature (like saving files) works, but you have to be careful about your product being dubbed as "crippleware". Lot's of people don't look twice at "crippleware". On the other hand, maybe these are the people you wish to avoid appealing to, anyway. Some products use very obtrusive payment reminders. This doesn't guard against unauthorized use, but does make it damn inconvenient. There are a few products that "self-destruct" after a specific number of uses. I like this approach, it has real nuisance value.

Many products (not just computer related) are purchased spontaneously, based on emotional desire, not on actual need. Generally, with this type of purchase, the individual wants it right now and is unwilling to wait. If they have to wait, they realize they don't really need it. The shareware marketplace is not conducive to this type of spontaneity, but there are things you can do to accelerate things. If at all possible, use your PC to provide a contact number for credit card purchases. In some areas, it may even be possible to automate the credit authorization procedure. If you can, use your home PC to download to the customer a temporary full version of your product until you have time to ship the finished version. You could even provide the paying customer with a code that turns the shareware version into the full function version (it would have to self-destruct in a few days).

Development Teams:

If you follow the guidelines presented here, the chances are good your shareware project will significantly increase in effort. So, if you want to succeed, your chances are a lot better as part of a development team, than as an individual. This is particularly true if you're going to try to develop a good game. High quality game development requires that you have access to a variety of sound and video cards (unless you use low resolution graphics or rely on VESA drivers). Using a development team approach spreads development and testing over a variety of configurations, to help ensure your product will perform as expected. It also shortens your development cycle, which is more significant to a software product, than to

a game. It wouldn't be much use spending eight months developing the perfect system to capture graphic images from a CD if somebody else beats you by two months. Don't miss your "window of opportunity". There's more PKZIPs and Wolfensteins out there, waiting for somebody to program them. Go to it!

About the Article:

This article was edited and updated by John Poulakos, from a presentation on shareware development, given by Dave Poulos at the Software and Shareware Developer's Conference in Houston (8/22 - 8/24/91), with the approval of Mr. Poulos.