

HARPOON ON VIDEO-REPLAY

Making replay video/animation files with v.3.5.8's auto-screengrab feature

By **Dimitris V. Dranidis**

With the release of version 3.5.8 of Harpoon 3, Jesse Spears has included a new and extremely useful feature: the ability to configure the simulation program to automatically create screen dumps at user-defined intervals (aka "Moviemaker"). Why this is so useful, you might ask? It allows the creation of successive screen-grabs of what the user is seeing. When viewed in reasonably rapid succession afterwards, these form an excellent video-style replay of the scenario.

The availability of such functionality enables a wide range of uses. Some conceivable applications include:

- Creating tutorials for the simulation. Often there arises a user question on some aspect of the simulation that is quite hard to be answered verbally. Visually demonstrating the how's and why's of the program can aid significantly in understanding the mechanics of the program. This is particularly useful for new players who might have trouble getting to grips with the simulation.
- Demonstrating tactics. When discussing tactics and ideas, it is unavoidable that one must first describe the relative positions of forces, both own and enemy (and possibly neutral) and the general situation, or else the discussion has no meaning and purpose. Being able to visually demonstrate this information and, even more important, how it shifts with time, can be of huge assistance. The old adage "a picture speaks a thousand words" comes to mind here. The others can now clearly see what the situation was and what you did; you only have to add *why* you did it.
- Reporting bugs/problems/simulation flaws. As voluntary tech-support staff for Harpoon 3, we at the HarpoonHQ regularly receive reports of various bugs or flaws in the simulation logic. Often these reports are "bogus", i.e. they are not flaws per se but rather perfectly reasonable outcomes of actions performed previously. In order to be able to determine the conditions under which the problem occurred, we need to have a good understanding of what was going on (scenario-wise) at the time. Again, it is impractical to expect from the user reporting the potential problem to be able to describe the virtual environment thoroughly and in detail. Video-replay really comes to the rescue in this case. If the problem is repeating itself under specific circumstances (in which case it is likely a specific flaw rather than a random bug), the user can record the scenario up to the moment of encountering it and then provide it to us along with the usual verbal description, enabling us to have a drastically more complete picture of the circumstances of the problem.
- Describing AARs. The principles here are similar to the endeavor of describing tactics: a large amount of information must be conveyed to others effectively, accurately and in a short amount of time. Visualising this information (which is, after all, visual to begin with) reduces substantially the amount of supplemental verbal/textual information that needs to be provided.

I am quite certain that, in due time, others will think of additional uses for this tool, given the flexibility it provides.

Now that we've seen how this feature can help us, let's take a look at how it works, how we can configure its output and what we have to consider in order to get the most out of it.

Basic operation

The ability to take a screenshot of the game in progress was available already from the Admirals Edition of Harpoon 2, with a menu command. In Harpoon 3, J.Spears added the option of taking the screenshot by pressing Alt+F1. However, until now sequential screenshots had to be taken manually, by repetitive use of the command. This is understandably imprecise with regards to the scenario-time intervals between dumps (particularly when time-compression is active) and distracts the user from concentrating on the scenario at hand. Automating the procedure is paramount if the visual output is desired to be of consistent quality and composed of frames arranged in timely intervals.

Activating the movie-maker function requires putting the *MovieMaker.opt* file in the "Options" folder of the Harpoon 3 installation (the file is on the "disabled" folder by default). Additionally, a text file called *Movietime.txt* must be present on the same folder with the program executable(s). This file must contain a single integer number; this will be the number of game-time seconds between screen shots taken. If you set this to 0, or leave the file blank, it will output EVERY screen update regardless of the time running; for instance if you pause the scenario and simply move around the tactical maps, this motion will also be represented. This can be quite useful in GUI tutorials and fast-paced engagements, but will result in a **huge** amount of screenshots taken (more on the problems of long replays below).

There is also another catch related with this number setting. J.Spears explains it best: "Note: this time figure is a *_minimum_* interval, so that if you have it set at 15, the time between screenshots will be AT LEAST 15 game seconds (it could be longer, depending on how often the screen is updating). If you want the game to output every 15 seconds, you still need to run the game clock at 15 seconds (or lower) to guarantee that you'll get a screen shot every 15 seconds."

So essentially, the interval figure counts not the game-time second intervals, but the "clock-tick" intervals between frames saved. If, for example, you have set an interval number of 2 on the text file, and during gameplay you use 5-sec time compression (i.e. each clock-tick passes 5 secs of scenario time), each frame will be 10 seconds of scenario-time after the previous one.

Pre-recording decisions

Before beginning to record a scenario session, you have to make two decisions that will have a large impact on the quality and size of the recording product:

- In what resolution is the recording going to be performed?
- What will be the interval between taken frames?

The resolution question is driven primarily by a related consideration: what is the display resolution of most of the users who are going to view the product? A common consensus is that most desktop users these days employ 17" CRT monitors with typically 1024x768 resolution, or alternatively 15-17" LCD displays with 1024x 768 or 1280x1024 settings.

Let's assume, for the sake of argument, that we are targeting a 1024x768 screen for the replay. Now, the recorded video is *probably* not going to be displayed on full-screen – therefore, a portion of the screen is going to be occupied by GUI elements, so a 1024x768 animation or slide-show is going to have a portion of its image thrown off-screen. This may or may not be acceptable, depending on the replay circumstances and the layout of the H3 session (for example, if the recording user wants to demonstrate something only on one of the tactical screens, and the message box is on the bottom of the screen and useless for the recording purposes, the user may not mind at all if it's left out on the replay). If we do care about having the entire recorded screen visible, it is recommended to go down one step on the resolution ladder, i.e. record at 800x600. (This has the added advantage of making the resultant individual screen dumps smaller in file size – this will help later when we're trying to make a video out of them or package them for distribution).

If you have made a recording on too high a resolution and are thinking of batch-downsizing the dumps through your image-app of choice, a quick word of advice - **DON'T**. Better go back and repeat the recording on the correct resolution. The produced images are in PCX format and are already well compressed; any attempt to downsize them, even on the same format, is likely to produce images of significantly larger size (or at least that's what happened in my case with three different popular apps – ACDSee, Photoshop and Ulead Photo Editor). Additionally, the image quality suffers from the downsizing: A screenshot originally grabbed at 1024x and downsized to 800x is *much* worse than a screen originally taken at 800x.

The interval between captured frames is something that depends on they type of replay you want to record, the file size you are targeting, and your past experience with successful (or failed) attempts to balance between motion smoothness and file size. The general rule of thumb is: For a given resolution and length of movie time, increasing the interval number (essentially reducing the sampling rate) will reduce the file size of the replay but will also make the frame transition less smooth. Conversely, low intervals can make a near-perfect replay of the scene but can increase the file size tremendously.

It is thus important to think of the type of scenario situation that you want to record. If the situation involves relatively simple, linear actions with predictable positions (e.g. a ship transiting the Atlantic), you can get away with a large interval number (low sampling rate) without losing much in the way of visual information. If on the other hand you're dealing with a fast-paced, violently dynamic environment (e.g. a huge air-battle over Central Europe) then it is probably advisable to sample frequently, i.e. set a very low interval number (even "0" if you deem it worthwhile).

Unfortunately, it is often the case with Harpoon that such conditions often co-exist (and frequently interleave) in the same scenario. Both air and naval operations often consist of what men in uniform describe as "hours/days of plain boredom interrupted by moments of sheer terror". This means that at some point in the scenario you may well find yourself recording at less-than-optimum settings (either too many shots on an uneventful duration, or too few shots in an "active" environment).

The obvious solution would be to dynamically adjust the interval number, depending on the action. Currently this can only be done manually: For example, start with a low sampling rate (high interval), save and quit when things heat-up, restart the program with high sampling rate (low interval), go through the action, then save and quit when things calm down, re-enter with high interval number and so on. This is of course an interim and quite crude solution; it would be much preferable to be able to change the interval value on the fly within the simulation. Hopefully this may be implemented in one of the next versions.¹

There is also a special case, that of user-interface tutorials. These, by necessity, will require a reasonably high sampling rate, so that the viewer can visually follow the flow of interaction with the program. You will also need to tick "Enforce Real Time" on the game options at scenario startup. Here is why: As explained before, the interval number of the *Movietime.txt* file refers to intervals between ticks of the game's virtual clock. However, when real-time is not enforced, at certain GUI menus & windows (such as weapon allocation, database browsing etc.) the clock freezes, and thus no new screenshots are taken. It is thus necessary to select this option in order to force the game-clock to continue counting even when these GUI elements are active, and thus take the proper screenshots needed for the demonstration.

Post-capture processing: Adding the missing zeros

Okay, so you recorded the scene you wanted, exited the simulation and now you are staring at a folder of the Windows Explorer, said folder having by default the name of the scenario you were recording and containing all the snapshots you have taken. These will be PCX files of the naming convention "[Scen.-Name]XXX.PCX", XXX being the sequence at which the screenshots were taken. Now, if you have used Windows extensively, you have probably noticed an awkward behavior of the Windows Explorer (which is really a shell for the OS's file system): when trying to order files with numerical strings on their names, it takes into consideration only the first digit. Thus, for instance, between two files named "File20" and "File8", it will place the first file before the second, although 20 comes after 8 in true sequence. That is because it only compares the first digits – and naturally places 2 before 8.

Why does that bother us, you might say? Because no matter what we do later on with the screenshots, owing to that logical flaw in Explorer's ordering perception, a number of the screen dumps are always going to be viewed (or batch-imported, or processed, or whatever else you choose to do with them) in the wrong order. Imagine seeing frame-1 of the scene you have recorded...and then having it followed by frame-100, 101, 102, 103 etc. instead of frame-2. Not good. What we need then, is to make sure that the full numbering of the files is somehow taken into account when the files are listed.

One effective method of doing that is to place extra zeros at the front of the numbers. For example, Frame20 becomes Frame0020 and Frame8 becomes Frame0008. In this way, the files are guaranteed to be ordered in their correct sequence.

A problem with this method is that it involves **a lot** of manual work by the user. For example, if there are 1000 frames taken, the user must add zeros to 999 of them by hand (only Frame1000 will not need them). There are better ways to spend an afternoon, so an automatic method to perform this is needed. H3 v3.5.8 automatically takes care of this up to a point: the program assumes that the number of screenshots that are going to be generated is going to be smaller than 10000, so it numbers the dumps in a four-digit scheme that spans from 0001 to 9999.

"What if I have more than ten thousand frames?" I hear you ask. In that case, you can download and use a small utility I have made specifically for this case, aptly called "Zero-Adder". Look for it on the Utilities section on the HarpoonHQ site: www.harpoonhq.com/utilities/. This little app allows for the proper renumbering of an infinite quantity of screenshots.

Viewing the replay: making a slide-show

Once you are done with renumbering the screenshot frames, it is time to actually do something useful with them. How do you actually view a replay out of them?

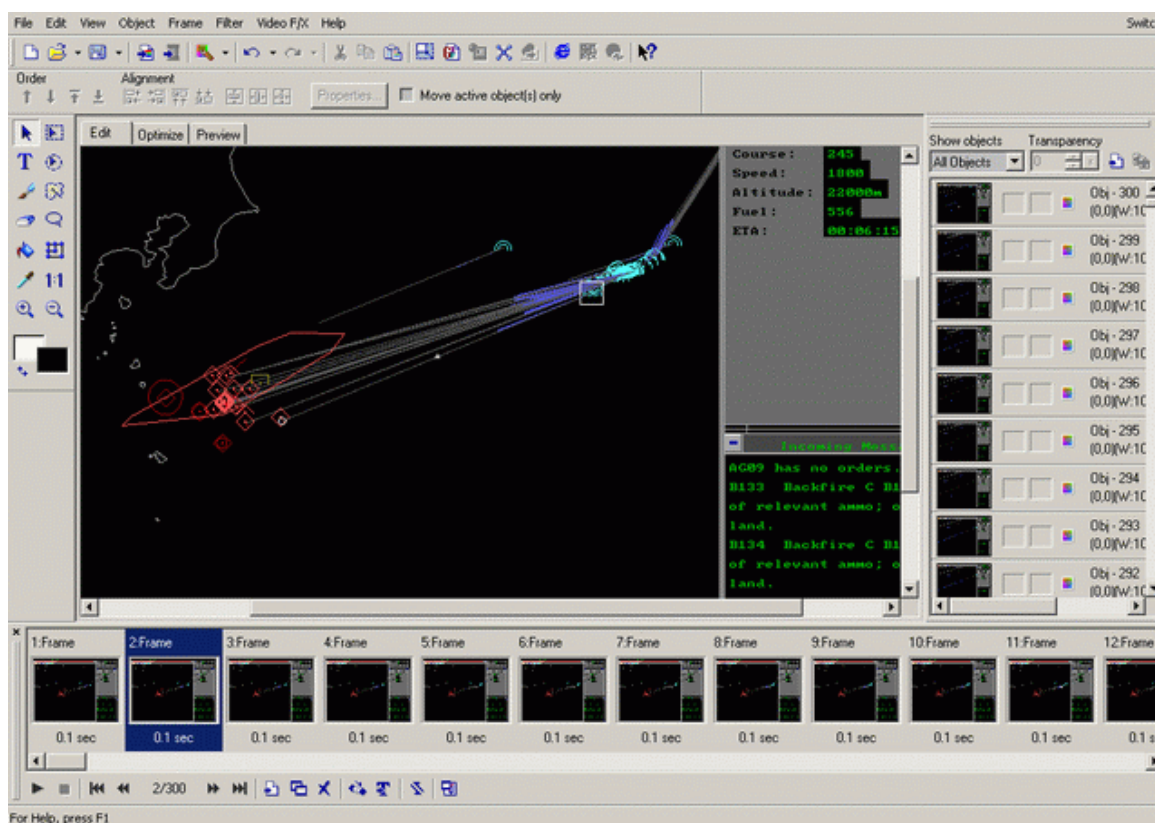
The first option, and the simplest/easiest one, is to view them as a slide-show. Most image-viewing/editing programs feature this function and most of them provide a host of detailed settings such as the time delay between frames, the order of showing them (forwards, backwards etc.) and other options. The benefit here is that you don't need to worry about further reprocessing of the images – you can simply view them as they are. The drawback of course is that this is not a real video animation per se; it is more a visual trick that gives the impression of motion. Having the full frames

¹ *There is yet another way of manual optimization: Record the entire sequence at a low interval (high sampling rate), regardless of the action. Then go through the files with your favorite image browser and selectively chop-off the frames of the "boring" scenes. While workable, this is definitely an exercise in patience and runs the risk of disrupting the game-time intervals between the frames if you're not keeping a close eye on the game-clock.*

also increases the disk space occupied by the replay sequence. That is not very important when you keep your replays to yourself, but becomes a problem when you want to distribute them to others. The good news here is that the raw PCX files can be quite efficiently batch-compressed. By using a popular compression program such as WinRAR or WinACE, compression ratios of up to 1:100 may be achieved using the best-compression (slow speed) settings; I have found WinZip to be less efficient. This makes it possible to pack-up replays of hundreds of megabytes into manageable-sized files suitable for download.

Viewing the replay: making a movie-like animation

The second option, and potentially the most interesting, is to create a “true” movie file out of the PCX screenshots you have. The benefits of this are obvious; the generated product is a true movie/animation-file that can be shared and distributed without any further packaging (although compressing files that are going to be transmitted through the web is rarely a bad idea). Furthermore, forming a movie out of the static frames usually reduces the file size of the same information drastically, thanks to the various image-compression techniques developed over the years for video-editing tools (there is a single exception to this, see below).



Making a GIF replay with GIF-Animator 5.0. Not for the feint of heart and most certainly not for the impatient.

There are many tools out here suitable for making an animation or movie sequence out of still frames: everything from freeware utilities up to heavyweight, feature-laden packages. These days I'm using the trial version of Ulead's GIF Animator 5.0 for the purposes of making single replay files. This application allows the generation of both AVI and GIF sequences, with a wide range of effects (most of them unnecessary for our purposes for the time being, but one can never tell about the future). It would be interesting to hear about recommendations from other users on the most suitable application for this specific task.

There are also various options for making a good animation out of the grabbed frames, each with its own benefits and disadvantages. Some of these include:

- **Making an uncompressed AVI sequence.** This provides a picture-perfect sequence and guarantees the widest compatibility with other PCs, since pretty much every Windows PC is able to playback uncompressed avi files. You pay dearly for this compatibility however: uncompressed AVI files are humongously large in size, since no effort whatsoever is made to optimize the frame-to-frame transition. This not only makes it very difficult to transmit these files outside a LAN, but also creates hiccups during playback unless you have a very strong I/O subsystem – the bit rate is simply too high.
- **Making a compressed AVI sequence.** This is one of the most popular methods and also the one with the fewer limitations on file size. Those in the habit of reproducing their (legally owned, of course) DVDs or VCDs

in a multitude of formats will be most familiar with this option. A wide range of codecs are available (from the ancient Radius Cinepak up to the very modern and efficient DivX 5.0) so there is plenty of choice and enough room for experimentation in finding the right balance between quality and size. One disadvantage of most of these codecs is the fact that they are optimised for real-film compression i.e. natural colors, smooth color transitions etc. In contrast, H3 uses a limited number of colors and sharp color transitions for which most of these codecs are not optimised. As a result, you may find the picture quality during playback to be less than perfect – particularly in high-compression rates, the lines of range-rings etc. may give the impression of “blurring” at the edges. Nevertheless, this option enables very satisfactory compression rates and allows full customization (codec type, bit rate, dimensions etc.) so that users can really tailor the final replay product to their own needs.

- **Making an MPEG1 or MPEG2 sequence.** Before the establishment of DivX as the de-facto standard for high-quality non-DVD movie sequences, MPEG1 was the choice of pros – and for good reason. MPEG1 combines satisfactory low bit rates (and therefore file sizes) with good audio and video quality (far more so than Apple’s ludicrous QuickTime at equivalent bit rates). MPEG2 improved on this standard with much-improved audio and video quality – good enough to become the standard DVD-Video format. MPEG2 however is a bit of an overkill for our purposes and is quite a bit rate hog, so it may be considered only for “internal use” replays. MPEG2 encoders are also quite hard to find at reasonable prices (it would be useful for readers to inform us of any new contenders in this area). MPEG1 encoding on the other hand, while offering nothing stellar in terms of small file size, has respectable video quality and can work satisfactory for H3 replays.
- **Making a GIF animation sequence.** The GIF image & animation format is probably the most perfectly suitable to the task of converting the PCX images to an animated sequence. The reasons for this become obvious when one considers the main technical characteristics of GIF: it handles only 256 colors (exactly as many as H3 employs) and favors sharp color transitions and large screen areas of the same color (of which both H3 uses a plenty). It also achieves **very** impressive compression ratios, often managing to squeeze tens of megabytes of PCX screenshots into a full-size GIF file of less than a megabyte.

The end-all be-all ideal method for a replay movie, then? It could be, but for two showstoppers: one, the process is **slow**. Users of less-than-bleeding-edge PCs should probably avoid this paragraph altogether if they wish to work on a really long replay file. Furthermore, it seems impossible to produce a GIF file much larger than 1.1-1.2 megabytes in size, at least using GIF Animator 5.0 and AVI-Edit 3.37. Both these programs report running out of memory when attempting to compile a GIF file larger than that (doubling my swap file to 1.5GB did not result in any relaxation of the limit). Therefore it is necessary to either split the intended replay into multiple 1-meg-sized GIFs or alternatively reduce the number of intermediate frames. If anyone out there has managed to overcome this limitation somehow, I would be **very** interested to hear about it.

This early article is the result of just two weeks of experimentation with the new auto-grab feature. As such, it will probably be updated in the future to reflect new discoveries made with regards to the various resolution, frame-rate and conversion options, new tricks for improving the reply quality under given sets of restrictions plus potentially improved implementations of the feature in future versions of the Harpoon 3 simulation. Until then, we encourage aspiring replay-makers to experiment on their own, try every trick in the book (and write a few new ones on their own) and contact us. Feeling ready to make your own masterpiece yet?