SpriteWorld 2 Frequently Asked Questions (FAQ)

This file contains answers to many frequently asked questions about SpriteWorld 2. Look here before emailing us if you run into a problem with SpriteWorld 2, have a suggestion, etc. This file contains a lot of useful information, so you may want to read through it even if you don't have a question right now. If you find any errors in this file, or have any suggestions or contributions, email Vern Jensen at:
Jensen@loop.com

For up to date information about SpriteWorld 2, and to download the latest version of SpriteWorld 2, visit the official Web site at:
http://users.aol.com/spritewld2/

==========
Frequently Asked Questions
==========

Q: Is there a SpriteWorld mailing list?

A: Yes, in fact there are two. One is for discussion so you can ask questions, help other people with their problems, share tips, source code, ideas, etc. The other is for announcements, so you can be informed when new versions of SpriteWorld are released. (The discussion list will receive announcements as well, so there is no need to sign up for both lists. The announcement list is provided for those who want to be informed when a new version of SpriteWorld is released, but don't want to be bothered with other messages dealing with SpriteWorld.)

- Discussion List -
To subscribe to the SpriteWorld discussion mailing list, send a message to spriteworld-subscribe@umich.edu with the words "Subscribe Discussion" in the Subject of the message. To remove yourself from the list, send a message to the same address with the words "Remove Discussion" in the Subject of the message. Make sure to send it from the same account that has a subscription to the list. If this is not possible, put the text "Special Instructions" in the Subject of the message, and your instructions for removal in the body of the message. (It will be handled by a human.)

To send mail to the SpriteWorld discussion list, write to spriteworld@umich.edu, and your message will be sent to everyone on the list. Please prefix the subject of your messages with "SWML:"; that will make it easy for those who receive your message to be able to tell that your message was sent from the SpriteWorld mailing list. Remember that you must also send your message to spriteworld@umich.edu when replying. Depending on your emailer configuration, it may automatically reply to the sender, in which case you'll have to replace their name with spriteworld@umich.edu.

Please don't send attachments to the SpriteWorld mailing list unless they are very small (under 10k). Either send the attachment directly to the person you want to receive it, or upload the file somewhere and send the URL to the list, so those who are interested can download it.

- Announcement List -
To subscribe to the SpriteWorld announcement list, send a message to spriteworld-subscribe@umich.edu with the words "Subscribe Announce" in the Subject of the message. To remove yourself from the list, send a message to the same address with the words "Remove Announce" in the Subject of the message.

Those subscribed to the announcement list will get very little mail; only one message every year or so when a new version of SpriteWorld is released, and possibly when a beta version is available for testing.

Note: Currently the list subscriptions and removals are handled manually, not automatically, so you will

not receive a confirmation message when you subscribe to or remove yourself from a list. In addition, the discussion list traffic is pretty light - we may go for weeks without any messages, so don't expect a flood of mail as soon as you sign up. If you want to make sure you have been subscribed, visit the following web site, which shows a list of everyone currently subscribed:
http://www.us.itd.umich.edu/~hinoue/spriteworld.html

Many thanks to Hajime Inoue (hinoue@us.itd.umich.edu) for volunteering to run both of these mailing lists!

==========

Q: Is there a C++ version of SpriteWorld? Are there Pascal interface files for SpriteWorld?

A: Yes. You can download SpriteWorld 2++ at:
http://home.earthlink.net/~trickys/SpriteWorld2Plus/

No Pascal interfaces have been made for SpriteWorld 2.1 or later, since no one volunteered to create them. If they become available, they will be posted at the SpriteWorld web page.

==========

Q: Are you going to do a port of SpriteWorld for Rhapsody, Windows, or BeOS?

A: While I'd love a port for any of those operating systems, I simply don't have the time or know-how to do it myself. However, if anyone else would like to do a port to any of these operating systems, I'm all for it! Just contact me (Vern) first, since someone else may have already started a port to the same OS you want to port it to.

Patrick Edson is currently working on a port to BeOS. To check his progress, visit this web page:
http://paclab.bu.edu/pers/pedson/SW/index.html

==========

Q: I haven't seen an update to SpriteWorld in a long time. Are you still supporting it?

A: Yes. Please don't assume that we aren't supporting SpriteWorld simply because you haven't seen an update for a while. I plan on supporting SpriteWorld for a long time, as I will be using it myself in future games. Generally, however, it may be a year or more between updates, due to the fact that I work on SpriteWorld in my spare time, and like to release new versions only after completing everything on my "To Do" list for the next version.

==========

Q: I'm getting compile errors in Shark Attack or SpriteTest, or I can't open the project files.

A: Read the document "About These Demos" in the SpriteWorld Examples folder.

==========

Q: After setting up the animation, when I draw to the window with SWUpdateSpriteWorld, nothing appears, or garbage appears, or the proper image appears, but with the wrong colors. Why?

A: Most likely CopyBits is trying to "colorize" the image that is copied from the offscreen area to the screen. If you are using the SWStdWorldDrawProc (CopyBits) as your screenDrawProc (the default drawProc), you must make sure the foreground color of your window is black, and the background color of your window is white before copying anything to the window; otherwise, CopyBits will try to colorize the image, which can produce some pretty strange results. See one of the SpriteWorld demos, such as Simple or the Scrolling Demo, for an example of how to call ForeColor and BackColor to make sure that CopyBits (if used) won't try to colorize the image. (The demos make the calls right before calling SWUpdateSpriteWorld.)

==========

Q: Why are my masked sprites not being drawn with a mask, or not drawn correctly, when I use one of the BlitPixie masked blitters?

A: Make sure the background (unmasked) portion of your sprite's image is white. The BlitPixieMask blitters require that the background color be white, or they won't work properly. This requirement was never mentioned when SpriteWorld 2.0 was released, but is now documented in the functions SWCreateSpriteFromPictResource, SWCreateSpriteFromSinglePict, and SWCreateSpriteFromSinglePictXY.

If you wish to load a self-masking Sprite that uses a color other than white as its transparent color, you must call SWSetTransparentColor. See the documentation for this function in Inside SpriteWorld for more info.

==========

Q: Would it slow SpriteWorld down if I added lots of layers?

A: Not really. SpriteWorld spends most of its time processing Sprites, so the number of actual Sprites, not layers, is what affects its speed the most. Naturally, the number of layers will affects its speed somewhat, but you'd have to add something like 100 or more layers before you'd be able to tell any difference, and it'd still be a very small difference at that. Even "empty" layers with no sprites in them don't take up much processing time at all. When an empty layer is encountered, SpriteWorld simply moves on to the next layer.

==========

Q: Can I switch between SWAnimateScrollingSpriteWorld and SWAnimateSpriteWorld?

A: Sure. There's really no difference between a scrolling SpriteWorld and a non-scrolling SpriteWorld, except that the scrolling SpriteWorld might have a slightly larger offscreen area. The only thing you need to do before switching to SWAnimateSpriteWorld is to move the visScrollRect to 0,0.

The benefit of switching to SWAnimateSpriteWorld is that you could have a scrolling game, and reuse those offscreen areas for some other part of your game which doesn't scroll, where you need a faster frame rate to perform some animation which would be too slow with SWAnimateScrollingSpriteWorld. Of course, one other option is to use SWFastAnimateScrollingSpriteWorld - see the Scrolling documentation for more information.

==========

Q: I've got a worldRect that's smaller than the window, so I can put stats and other stuff in the rest of the window. I use QuickDraw to draw lines in my worldRect when the ship shoots its laser. The problem is, the lines aren't clipped to my worldRect - they go right into my stats area! How do I fix this?

A: To clip QuickDraw drawing to a certain rect, set the port to your window and call ClipRect(&myRect), where myRect is the rectangle, in coordinates local to the window, of the worldRect that you want the drawing clipped to. However, before doing this, make a call to GetClip to save the clipping region, and then call SetClip to restore it after you're done (so you can draw to the stats area of your screen again.)

==========

Q: I'm seeing "double images" of my Sprites or scrolling tiles that move at a fast rate. How do I fix this?

A: This occurs when the animation runs at a slower speed than the refresh rate of the monitor, such as an animation running at 30fps on a monitor whose refresh rate is 60hz. In this case, your eye builds an image of the sprite halfway between its new and old positions whenever it is moved, causing the blurriness. For instance, if you have a 40x40 sprite that is moving from row 0 to row 40, your eye will see the old image at row 0, a "fake" image at row 20, and the new image at row 40.

The only way to fix this problem is to either run the animation as fast as the monitor is refreshing the display, or to slow down the animation so much that your eye can't blur the images. The latter generally isn't appropriate for games, since you'd have to slow it down to less than 10fps, and the former has its own set of problems. For one, you'd have to base the speed of your animation on the refresh rate of the monitor, meaning that your game would run at different speeds on different monitors, since not all monitors refresh at 60hz. (Larger monitors in particular need a faster refresh rate such as 75hz, since otherwise the pixels will start to fade before the scanline can make a complete pass.)

What's the solution? Just find a frame rate that's good for your game and try to ignore the ghosting effect. You're probably much more aware of the effect since you're working on the game, than someone who plays it will be. For instance, play Maelstrom, and move the ship around and shoot. The ship gets blurry and it looks like there is more than one bullet. But I'll bet you never noticed this before!

==========

Q: It would be cool to be able to make a multi-parallax scrolling game. Do you plan on adding parallax scrolling support to SpriteWorld?

A: Parallax scrolling will be added sometime, but I can't say when. I may not even be the one who adds it. But if I do add it, I probably won't do so for another years or so, since I've got a ton of other stuff to do first.

==========

Q: Are you going to add a pixel-doubling option to the scrolling routines?

A: I'm not planning on it. Generally games that use pixel-doubling are either PC ports, or Mac games where the author wanted the game to be able to run on slow Macs (in a tiny window), but take advantage of the speed of faster Macs by allowing the game to also be played in a double-size window. The only problem with this is that the tiny window is hard to see, and when the graphics are doubled in size, they usually look pretty bad. Hopefully the scrolling routines provided by SpriteWorld are fast enough that you can design your game to run in one size window, and still have it run acceptably on most Macs. Or you could allow the window to be resized, allowing the user to see more or less of the scrolling world at once, instead of scaling the image to fit the window.

==========

Q: Is there a Level Editor for SpriteWorld?

A: Ian Baxter has made an excellent, free editor which is currently in beta testing. You can download it here:
http://www.buglineltd.com/ian/bluefinkle/levelcreator/

==========

Q: Why don't you optimize SWAnimateSpriteWorld so that whenever the oldRect and destRect of a Sprite don't overlap, the screen is updated in two pieces instead of one? That way, the two pieces would be smaller than the large deltaRect that would normally be used to both erase the old position and draw the new at the same time.

A: Karl and I have both considered this scenario, because we also thought that it could be improved. However, when we took it into consideration, we found that:

A) Most sprites don't move faster than their size.
B) SpriteWorld would probably waste more time checking to see if it should draw a sprite in 2 rects instead of 1 than it would save when actually drawing the sprites that can be optimized using this method.
C) Often drawing a sprite in two pieces would actually be slower than drawing it with one (larger) rect, even if more pixels would end up being copied, simply because of the overhead of calling BlitPixie two times instead of one. The more you break it up, the slower it goes.

Another consideration is that SpriteWorld automatically aligns the left and right side of each sprite's deltaRect to the closest long-word boundary. In 8-bits, this would be every 4th pixel. So consider the following rects, where "A" shows the old rect, "0" shows background, and "B" shows the new rect:

AAAAAA000BBBBBB
AAAAAA000BBBBBB
AAAAAA000BBBBBB

If the right side of A does not end on an even long word boundary, its rect will be pushed right to the next long word boundary. This would mean that even if two rects were used to copy the old and new position of the sprite, just as many pixels would be copied as there would be otherwise. However, I guess this example is a little biased - let's give a more "realistic" example, where is sprite is moving diagonally:

AAA000000
AAA000000
000000000
000000000
000000BBB
000000BBB

Here it would seem that things could be sped up by copying the old and new rects in two separate pieces. However, there are two more considerations, besides the ones mentioned above (such as BlitPixie's overhead):

1) Blitting rects from offscreen to the screen goes very quickly, at least when you're blitting small rects (i.e. not the whole screen, as in a scrolling game). The time it takes to copy a rect from the offscreen area to the screen pales in comparison with the time it takes to draw the sprite offscreen, which is much slower, even with compiled sprites.

2) If you were to copy two rects instead of one, both rects would have to be aligned to long word boundaries, which naturally results in more code needing to be processed than when one rect needs to be aligned. Here's what the aligning code looks like in SpriteWorld.c:

```
{
    short temp;

        // align the left edge to long word boundary
    curSpriteP->deltaFrameRect.left &=
            (spriteWorldP->workFrameP->leftAlignFactor);

        // align the right edge to long word boundary
    temp = curSpriteP->deltaFrameRect.right &
        spriteWorldP->workFrameP->rightAlignFactor;
    if (temp != 0)
    {
        curSpriteP->deltaFrameRect.right +=
                (spriteWorldP->workFrameP->rightAlignFactor + 1) - temp;
    }
}
```

Between the BlitPixie overhead, the extra alignment calls needed, and the fact that SpriteWorld would have to process extra code to test every single Sprite to see if it should be drawn with two rects instead of one lead me to believe that this "feature" would actually make SpriteWorld slower rather than faster. About the only time something like this might help would be if you have a Sprite whose delta is *much* bigger than its size, such as an 8x8 Sprite that's moving 100 pixels diagonally each frame. However, I've never seen a game where a Sprite would move this fast, so I don't think this is something to even consider adding to SpriteWorld.

==========

Q: What should I use for playing asynchronous sounds?

A: There are several options. For one, you could use the sound kit included with SpriteWorld; see the add-on Docs folder for more information. However, if you need more features, I'd recommend using "Hollywood"; a nicely implemented library for playing asynchronous sounds. It's included on the CD-ROM that comes with Tricks of the Mac Game Programming Gurus (Hayden Books, (800) 428-5331, hayden@hayden.com), and you're free to use it in your projects. As far as I know, the only (legal) way to obtain Hollywood is by purchasing Tricks of the Mac Game Programming Gurus. I believe the book is currently out of print, but you can usually find used copies available for auction at www.ebay.com. (You can also get some great deals on older PowerMacs at eBay.)

An alternative is the "Caveman Sound System", a freeware sound library by David Hay <hay@alumni.cs.colorado.edu>. I haven't worked with Caveman myself, but it looks very full-featured and well done. Caveman Sound System should be available from the following Info-Mac mirrors:
http://hyperarchive.lcs.mit.edu/HyperArchive.html
ftp://tornado.rfx.com/pub/info-mac/_Development/_Library/
ftp://ftp.dataplex.net/pub/info-mac/_Development/_Library/
ftp://ftp.pht.com/mirrors/info-mac/_Development/_Library/

==========

Q: Will you add gamma-fading routines to SpriteWorld?

A: SpriteWorld is an animation library. If I were to try to turn it into a "Game Construction Kit", it would be too difficult to maintain. (It takes up a lot of my time as it is!) There are already many options for doing gamma fading; for one, you could use DrawSprocket. For another, you could download one of the gamma fading libraries available freely on the internet. There's no reason this needs to be incorporated into SpriteWorld.

==========

Q: Is DrawSprocket compatible with SpriteWorld?

A: You can use DrawSprocket's depth and resolution switching features, as well as its gamma fading routines, but unfortunately you can't use its capabilities for page-flipping, etc., while also using SpriteWorld. A new version of SpriteWorld would have to be made to take advantage of this, and I currently don't have the time to do this. :-(

==========

Q: Just wondering if you were going to release a PDF, DocViewer, or DOCMaker format of the SpriteWorld 2 documentation?

A: Some alternate versions of the docs have been made and are available from the SpriteWorld web page.

==========

Q: What programming books would you recommend for a beginning programmer?

A: For questions such as these, you should post a message to comp.sys.mac.programmer.help. Please do not write to Karl or Vern with questions like this. That being said, you can get some good books from Developer Depot. Call 1-800-MAC-DEV1 to request a free copy, or visit them at www.devdepot.com. They have books on C, such as Learn C on the Macintosh by Dave Mark, books on learning to program the Mac once you know a programming language, such as the Macintosh C Programming Primer, Volume I by Dave Mark and Cartwright Reed, as well as books on other topics, such as Tricks of the Mac Game Programming Gurus. They also have the complete Inside Macintosh Series. The books IM: Toolbox Essentials, More Toolbox Essentials, Imaging with QuickDraw, and Memory are all essential references once you've completed a book that teaches basic Macintosh programming.

==========

Q: I’m using Think C, and I’m getting errors like “prototype required for function ‘LMGetMBarHeight’” ;, and “Error: syntax error: ‘SInt8 mmuMode;’”.

A: The problems you’re describing all have to do with the fact that you have Think C’s “old Headers” installed on your system. The “Headers” refers to the “.h” files Think C uses to interface with the Mac toolbox. These header files have changed since Think C 7 was released, and SpriteWorld 2 was written to use the newer Headers (usually called “universal headers”). It’s possible to get SW to compile with the old Headers, but that’s not the best solution. It would be a much better idea to install the universal headers on your system, since a lot of other programming libraries you may want to use in the future will require the universal headers. Also it will make it easier to switch to a newer programming environment if you start using the universal headers now.

Installing the universal headers in Think C requires two steps: First you have to acquire the header files

themselves and copy them to your hard disk. The headers are freely available from Apple, and can be found at various sources on the Internet <http://devworld.apple.com/dev/sdk.html>, as well as commercial online services such as AOL. The headers arrive contained in a folder named “CIncludes”. You need to copy this into a folder called “Mac #includes”, which is located in a folder called either “Think C” or “Symantec C++ for Macintosh”, depending on which version of Think C you’re using. Thus, the path should be:
Symantec C++ for Macintosh:Mac #includes:CIncludes:
You then need to remove the folder called “Apple #includes” from the “Mac #includes” folder, as this contains the old Headers.

The next step in installing the universal headers is to create a new “MacHeaders” file; this file also resides in the “Mac #includes” folder. To create the universal headers version of MacHeaders, you must “precompile” the source file “Mac #includes.c”. If you have Think C 7.0 or later, this file is in the “Mac #includes” folder. If you have Think C 6.x, contact Symantec for a copy of Mac #includes.c. To precompile this file, first open any project with Think C. Then open the source file “Mac #includes.c”. Read the comments at the beginning of this file for instructions on setting its flag for the universal headers. With this file as the front window in Think C, select “Precompile...” under the “Source” menu, and save the result as “MacHeaders”, replacing the old “MacHeaders” file. You will then have the universal headers installed.

==========

Q: How do I update CodeWarrior to Universal Headers 3.0.1 or later?

A: Naturally, every version of CodeWarrior is different, so I can't guarantee that this method will work for all versions, but the instructions below should work with CW 9 and 10. You shouldn't need to install the Universal Headers in CodeWarrior Pro, since it already comes with a version that is compatible with SpriteWorld.

Here's the directory tree for the "MacOS Support" folder for CW9, so you can compare it to your version of CW. If they're the same, then you can follow the instructions below word-for-word:

```
MacOS Support: Headers: 68K Specific
                        ANSI Headers
                        PPC Specific
                        Res Headers
                        Universal Headers
               MacHeaders
               Libraries: ANSI 68K
                          ANSI PPC
                          MacOS 68K
                          MacOS Common
                          MacOS PPC
                          Pascal 68K
                          Pascal PPC
                          Runtime
                          SIOUX
               Interfaces: 68K Specific Interfaces
                           ANSI Interfaces
                           Projects & Libraries
                           Universal Interfaces
```

Of course, you may have extra items depending on what install options you chose. There is no Interfaces folder in CW 10, but that doesn't matter, as you don't need to do anything with this folder.

Note: These instructions are for the C Universal Headers only - no pascal, no assembly. If you've got the

full Universal Headers, you'll need to do some other stuff.

First, you need a copy of the Universal Headers. You can download them from <http://devworld.apple.com/dev/sdk.html>. Then follow the steps below (ignoring the OpenTransport and MPW only folders). At all stages, when copying, choose to replace any existing items when prompted by the Finder.

1. Put the contents of "Universal:Interfaces:CIncludes" in "MacOS Support:Headers:Universal Headers"

2. Put the contents of "Universal:Interfaces:RIncludes" in "MacOS Support:Headers:Rez Headers"

3. The document "NewRoutineNames.dict" in "Universal:Interfaces" is a text file for reference purposes, and can be put anywhere. (This may or may not be included with your headers, depending on what version you're updating to.)

4. Put the contents of "Universal:Libraries:Classic68kLibraries" in "MacOS Support:Libraries:MacOS 68K"

5. Put the contents of "Universal:Libraries:StubLibraries" in "MacOS Support:Libraries:MacOS Common". The libraries in that folder should be the only copy of each library that you have; i.e. make sure no other libraries, such as the ObjectSupportLib, are in any other folders.

6. Take "MathLib" out of "MacOS Support:Libraries:MacOS Common" and put it in "MacOS Support:Libraries:MacOS PPC". (Replace the existing file.)

7. Open each of the three projects in "MacOS Support:MacHeaders". (Of all the files in the folder, the only ones which are project files in CW 10 are MacHeaders68K.[1], MacHeadersCFM68K.[1], and MacHeadersPPC.[1].) Choose "Bring Up To Date" from the "Project" menu. Don't make the project, or you will get funny libs with no name appearing. You may also need to bring up to date the headers for other CW items that rely on the standard headers (e.g. PowerPlant).

Congratulations. You're done!

The constant OLDROUTINENAMES was on in CW9, but was off after the update to U.H 3.0.1, so you will have to change a lot of procedure calls in your projects (e.g. GetDItem -> GetDialogItem). Use the "NewRoutineNames.dict" document to help.

Some of the toolbox functions now have stricter type checking. (For example, the Delay() function now requires an unsigned long instead of a long.)

Some functions have moved to new header files - in particular, SysBeep is now in "Sound.h", and now has to be included specifically.

Also, the new headers and constants may cause conflicts with your project (e.g. can't use menus.h for one of your own project headers anymore!). If you get bizarre errors when trying to compile your project, make sure none of your project headers have the same name as any of the files in "MacOS Support: Headers:Universal Headers".