Ch2 Display modes

Stuff I already know

CGA/EGA/VGA/MCGA/SVGA screen resolution/color depth..

Resolution adaptation

most graphic artists tend to
learn how to create game graphics at a specific resolution and, over time, adjust their drawing styles to match the particular quirks and eccentricities exhibited by that resolution. As a result, many artists aren't able to easily adapt to a different screen resolution without a significant amount of retraining and practice. For example, if you're used to drawing your artwork at a resolution of 320x200 and suddenly have to work at a resolution of 640x480 for a project, you may be in for a rude awakening. You'll quickly discover that the pixel size and orientation of this screen resolution has totally changed, potentially throwing you off and slowing you down.

Missing material

Aspect ratio is dealt with, but not screen modes that have doublewidth or doubleheight pixels (pixel aspect ratio = .5 or 2.0)

Ch4 file management

file naming

- Consult with others
- be descriptive
- be consistent
- document the scheme.

Sample scheme is ${TYPE}-${DESC}-${SIZE} where TYPE is eg img, spr, mnu, blk and SIZE is eg 64x64px.

IMO the latter part is no longer required since dimension info can easily be extracted via Properties in file manager or via imagemagick etc. It's also reasonably common to use a subdirectory instead of part of the filename for $TYPE, when all resources of $TYPE have a strong relationship with others of the same $TYPE and a weak relationship with others of a different $TYPE. This is more efficient to store as well as quicker to use.

directory organization

- one dir per project
- use subdirs (as I suggest above)

file exchange

Disk section is largely obsolete. Just plan based on ISO9660 (DVD/CD) restrictions with Rockridge extensions for long filenames, this effectively transcends crossplatform concerns. The 'exfat' filesystem used on mobile devices and other SSD based platforms is another possible target. It appears to have slightly less cruft than ISO9660

File format conversion

Largely out of date: the 'base set' of fileformats is more clear now than at the time of writing. Here are my suggestions

- Use PNG for everything static. Avoid application-specific extension blocks. Use compression=9. Apply optipng or similar to minimize distributed size.
- GIF files are an option for <=256c animations. But for distribution, generating a PNG sprite sheet with an associated file that has animation info, will reduce loading times (otherwise, assembling the frames into a sprite sheet may be needed to be done at load time)
- PSD or XCF for complex images that are used as resources for generating other images, but are not themselves resources that the game will load. Depending on frequency of resaving, enabling compression may be wise — use RLE mode for PSD, save .xcf.gz 's instead of .xcfs for XCF.
    - Other compression methods — .xcf.bz/bz2 (bzip2) and .xcf.xz (xz, which is related to the type of compression done by 7zip) — are available. These do usually obtain 5%-30% better compression. However, saving time is substantially inflated (up to 10 times as long). .xcf.xz is also only available on relatively recent versions of GIMP, so it may not be a good choice for exchange.

APNG or possibly even WEBP files may replace GIF in the future, but for now they are not that well supported.

Compressed archives

Various compressed formats are described here, but the different types are mixed up. There are two broad types: Plain compression, and Container. The former (eg .gz, .bz/bz2, .xz) are single file containers — they take a single chunk of data and compress it.

The latter (eg. .tar, .zip, .rar) collect several files and possibly compress them. In .tar's case, it includes options to create compressed archives, but these archives are actually compressed by a second program (which can be anything, but .tar provides easy options for .gz, .bz2 and .xz)

It's probably worth mentioning that .tar.xz is increasing in usage for package distribution on Linux systems, as it provides outstanding compression at the cost of memory usage when decompressing. This memory requirement is perfectly manageable on most remotely modern systems. It also has advantages over the broadly comparable 7zip, on

Linux: xz may be installed by default, if it isn't it's very easy to remedy that. Whereas on 7zip is not installed by default on other platforms, and is not installable in a few clicks.

This chapter does not cover ZIP64 and similar extensions for very large archives.. Probably they weren't established yet. Personally I'd suggest that if you need to stick more than 4gb in a single archive, you have an asset management problem.

Overall I agree with the suggestion that ZIP is the most useful overall: it provides reasonable compression, is readable on all platforms, and compression is per-file (meaning that any corruption in a particular area will usually only effect that file, not the entire archive)

Backups

Hazards outlined include

- accidental file deletion
- hard drive crashes
- system crashes
- Natural disasters

Much of this section is reviewing standard information about backup systems.

Backups are (now) cheap and can be done over the internet (Dropbox etc) People, including me, still routinely neglect to do them.

Version control

This section is lacking. Aside from the option of 'manual' versioning, versioning with Git or similar is a better option than the ancient RCS. Finding a Git GUI you like is also important.

The problem of diffing between graphics files is still largely unresolved, and you may need to create simple tools to implement this.

AFAICS:

- Comparing two versions of a file that have the same dimensions and number of frames, should generate an image with transparency in areas that haven't changed between the versions, and solid pixels taken from the 'target' version in areas where they have. For animations, you need to pick a suitable disposal method (replace) so that the transparent areas do not show through pixels from previous frames. You can then view the set of 3 images: (a, diff, b) in your favorite image viewer.
    - Animated diffs are currently limited to GIF (since it needs to be viewable in your image viewer). In some cases this may require color reduction. Generating a movie clip using a lossless codec can circumvent this, but movie viewers generally don't have panning/zooming facilities as accessible and easy as image viewers.

- Versioning is useful at any scale, but the same restrictions applied to archiving apply here, as Ari vaguely alludes to:
- Versioning introduces disk overheads. Git is most efficient in this regard, particularly if you set your git gc threshold so that GCs get performed often, But as a rule of thumb, versioning data occupies at least SIZE*log(nchanges, 2) bytes on disk.
- Example:
- Supposing you have an image of 10000 bytes in size, and you have committed 20 different variations of it. log(20, 2) is 4.3. Then it probably takes at least 10000 * 4.3 → 43000 bytes to store the versioning data for that file. After you have accounted for the versioning data, you must also include the size of your working data (eg. 43000 + 10000 → 53000 bytes for that file and its versioning data) to account for the total disk space you will need.
    - This is mainly a concern for larger files like CG or 3d data.
- You should also consider what not to version.
    - eg. generated files, like 3d renders
    - Git and related systems make this fairly easy via '.gitignore', which allows you to specify file patterns and paths to ignore.

asset management

Use when projects are large (dozens or hundreds of files)

Gives these criteria:

- Database driven
- Thumbnail support
- Asset keywords
- Large catalog capacity
- Multiple image catalogs
- Low cost

I believe the combination of TMSU + sxiv, with appropriate image-info / key-handler scripts, covers that. I already use it for references, it supports multiple databases in a similar manner to git, it supports hundreds of thousands of files. The only lacking feature is of annotations with spaces in them — these are not legal in TMSU. My annotation extension for TMSU addresses this.

Perhaps it is also lacking for non-image formats. My solution to this was to automatically create 'meta-thumbnails' — thumbnails showing an image form of non-image data (eg. an image containing multiple thumbnails of different frames, for a movie)

10 font

Covers various font characteristics.. introduced me to the concept of font color, which has nothing to do with hue, saturation, or value. One noticable lack is the mention of SVG

fonts, which combine the virtues of both TrueType (scalable, kernable. .) and bitmap-based (color, complex patterning)

General principles:

- emphasize legibility over aesthetics
- avoid custom bitmap font
- Avoid ALLCAPS words
- Be conservative with use of bold, italic, underline
- Avoid AA on very small text (my note: doesn't apply so much for bitmap fonts)
- Use kerning and tracking adjustments as needed
- .. and line spacing.
- Make moderate use of centred text
- use sans serif for game or casual text
- use serif for elegant / serious text