

Chapter 11

Planning Arcade Game Graphics

In this chapter, you'll learn about:

- ♦ **Planning your arcade game graphics**
- ♦ **The game summary**
- ♦ **The game action sequence**
- ♦ **The graphics specification**
- ♦ **Technical restrictions**
- ♦ **The game glossary**

Behind every successful arcade game project stands a well-crafted plan. This being said, before you plot even a single pixel, you should take the time to plan out every aspect of your game's artwork from its appearance to how the individual elements and objects will interact with each other when displayed on-screen. Simply put, creating a detailed plan at the start of each new game project will save you time and effort, reduce mistakes, and improve the overall quality of the artwork that you create.

The Design Plan

Let's take a look at the *design plan*, or the document that details and records all of the creative-related issues that are associated with your game. An effective and well-written design plan will include most, if not all, of these elements:

- Game summary
- Game action sequence
- Graphics specification
- Technical restrictions and stipulations
- Project schedule
- Game glossary

Before continuing, it's important to realize that the extent to which you adhere to these elements in your own game design plans really depends on the size and sophistication of the game project at hand. Larger games should include all of these items, while smaller games can omit one or more of these steps. Use your best judgment here, especially if time is an issue. There's no reason to craft a full-blown design plan for a really simple game. At the same time, however, you're only hurting yourself if you skimp on these details for a big game project.



NOTE: The purpose of this chapter is to point you and/or your development team in the right direction regarding how you want the game to look and work—nothing more. The last thing I want to do is dictate how something must be done. After all, what works for me might not work for you.

The Game Summary (required)

The game summary provides a general overview of the game in question. The purpose of this part of the design plan is to get you thinking about the look of the game as early as possible in the design process. Since the game summary serves as the foundation of your design plan, it should be comprehensive but not overly detailed; use the other sections of the design plan to really get into specifics and fill in any gaps that might develop.

It should include these elements:

- Game back story
- Game description
- Game object inventory
- Game functionality overview

Game Back Story (optional)

This section provides the basic background story for your game.

Generally speaking, the back story doesn't have any direct influence on the game except to make it seem more interesting to the player; therefore, you can leave it out if you want.

In terms of size, it can be as descriptive or as vague as you want. It's really up to you.

Example:

The year is 2299 and you've just arrived to assist in the asteroid sweeping operation along the frontier.

You volunteered for this duty in the hopes of avoiding long-term military service on some alien-infested world. You thought it would be easy. Well buddy, it looks like you were wrong...

Game Description (required)

The purpose of this section is to summarize and explain the overall theme, function, and mechanics of your game. This section is required because without a game description, no one will have any idea what your game is about!

Try to limit the length of this section unless there's a particular need to go into more detail. One or two short paragraphs should be sufficient.

Example:

Disasteroids is an "asteroids"-style arcade game in which the player maneuvers a small spaceship through a dangerous planetary asteroid field. The player scores points by shooting at and destroying these asteroids. At the same time, the player must avoid dense bands of roving asteroids, fast-moving asteroid fragments, and other perils including belligerent alien spaceships and mines.

The asteroids that appear are all randomly generated; however, each game level assigns a specific number of asteroids to be destroyed called a *quota*. After all of the asteroids on a given level are eliminated, the player is automatically transported to the next one.

Game Object Inventory (required)

This section is used to list and describe all of the game's objects from background images to text elements. Each object description should include pertinent information on object properties such as size and color, as well as any associated actions or movements. Doing this not only helps you get a better sense of the game's objects, but should help anyone else involved with the project as well.

The object inventory list should be as thorough as possible, but you should try to keep the individual object descriptions brief. Two or three sentences per object should be more than enough at this stage. Later, once you have a better sense of the various game elements, you can add more.

Examples:

- **Player Ship**—A round, metallic spacecraft that is controlled by the player. The player's ship is equipped with a blaster cannon that shoots various types of energy projectiles and it can move in all eight directions.
- **Arian Magnetic Mine**—A small, metallic, slow-moving object that drifts randomly across the game screen. It will be completely vaporized when it collides with another object (except another mine).



NOTE: You may also find it helpful to include rough sketches of the different game objects mentioned here. You can place them on a separate page or simply draw them alongside each object's entry. Doing this can help you and other development team members to get a better handle on what you're thinking regarding the objects you're designing. It might also help eliminate any *dead-end* artwork, or art that doesn't quite fit the feel of the game, before things go too far.

Game Functionality Overview (required)

The purpose of this section is to define the game's desired functionality. It should consist of a simple, bulleted list.

Example:

- Runs on any Pentium-based PC running Windows 95, 98, or NT 4.0.
- Offers 10 increasingly difficult levels of play.
- Includes a special arcade "Classic" mode with eight levels of play.
- Battle four types of asteroids, alien spacecraft, minelayers, and mines.
- Obtain weapon power ups and extra lives.

The Game Action Sequence (required)

The game action sequence is essentially a storyboard that details the various screens needed to play the game, the order in which they appear, and the various attributes and actions associated with them. All arcade games include two types of game screens: *functional screens* and *play screens*.

Functional screens are screens that contain game-related information, control, and configuration options. This includes title/introductory screens, menu/option screens, transitional/cut-scene screens, help screens, credits screens, and game over screens.


Play screens are any screens where game activity occurs; in other words, they are the actual game levels.

The game flow sequence consists of three parts:

- The game action flowchart
- The screen summary
- Screen mock-ups

The Game Action Flowchart (required)

This is a diagram that illustrates the basic flow of action within the game. It can range from something as crude as a piece of loose-leaf paper with hand-drawn flow diagrams to something very sophisticated that was produced with a professional-quality flowcharting package. Either is fine as long as they include information about the various paths of action that can occur in the game.



NOTE: Whenever possible, use flowcharting software to create these diagrams. Such programs allow you to make changes easily and will give you flexibility when it comes to printing and sharing the information with others. To help you with this, I've included a trial copy of *WizFlow* flowcharting software on the book's accompanying CD-ROM. See Appendix B for more information on this program.

Example:

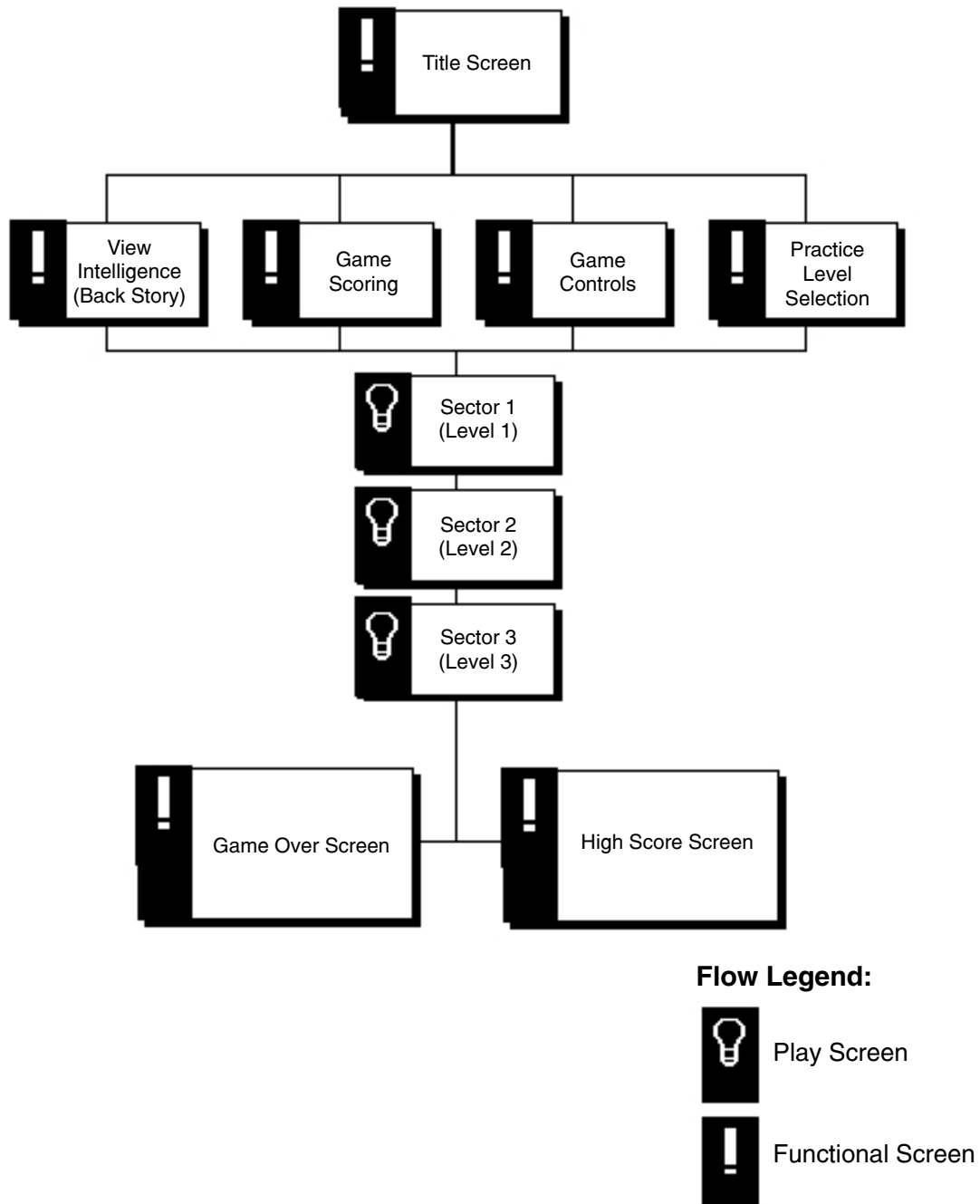


FIGURE 11-1: Example of Game Flow Sequence

The Screen Summary (required)

The screen summary should include a brief description of what each screen does along with a rough, general description of the various objects that may appear on it and any actions it supports.

In terms of length, for now, one or two short paragraphs per game screen should be enough.

Examples:

Title Screen

This is the screen that the user sees upon starting the game. It contains the program's title, credits, copyright information, and menu elements from which the player can access different game options.

Available Game Actions:

- Play a game (go to level one)
- View game controls (go to the Game Controls screen)
- View game scoring (go to the Game Scoring screen)
- View game help (go to the View Intelligence screen)

View Intelligence Screen

This screen is the game's main information screen. Among other things, it provides the player with some of the game's back story as well as a summary of the different game elements and tips on how to deal with them.

Available Game Actions:

- Play a game (go to level one)
- View game controls (go to the Game Controls screen)
- View game scoring (go to the Game Scoring screen)

Screen Mock-ups (required)

This section should include illustrations, sketches, or diagrams of each game screen in order to help you to better visualize how each game screen will look.

Whenever possible, it is highly recommended to graphically mock up each screen since better visuals result in less confusion and give everyone concerned a much clearer idea of what each screen will look like. However, if you're lazy and/or just pressed for time, rough sketches can usually work just as well.

In completing this section, it's also good practice to label your objects and map them back to your object inventory list.

Examples:

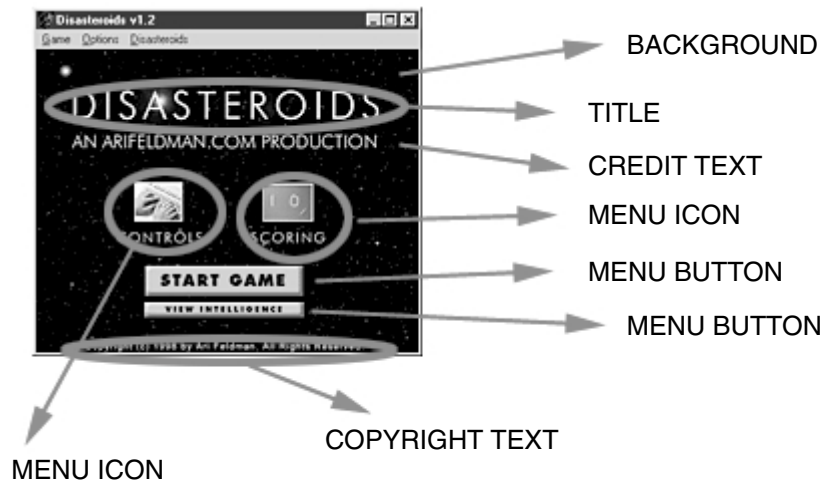


FIGURE 11-2: The Title Screen

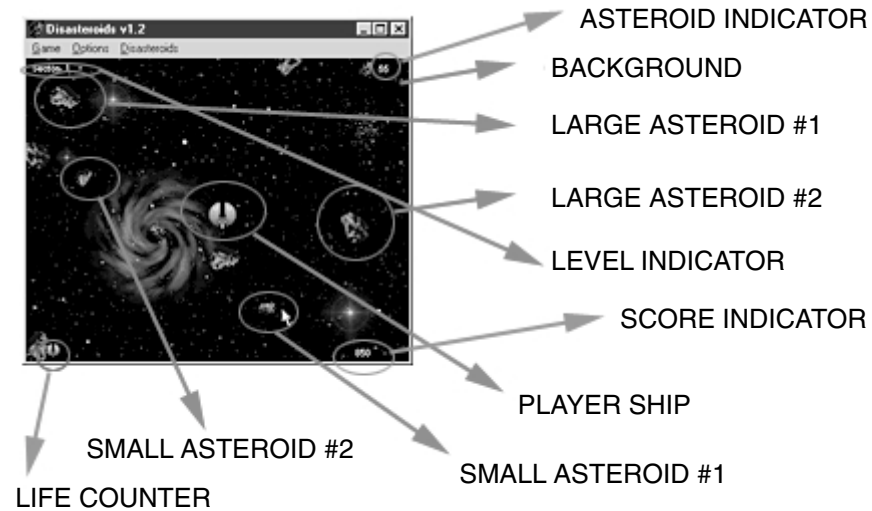


FIGURE 11-3: The Main Game Screen (Levels 1-3)

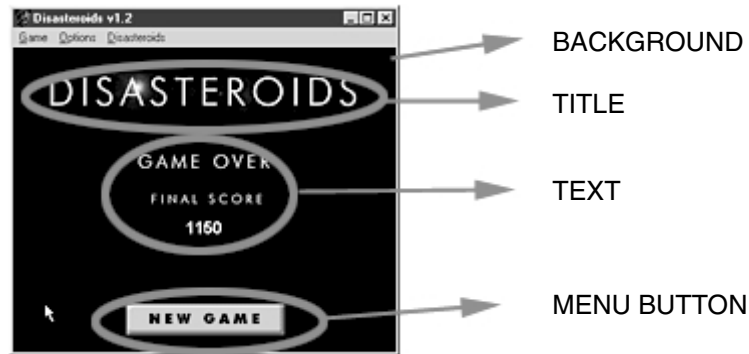


FIGURE 11-4: The Game Over Screen

While this seems like a lot of work (and it often is) to include such screens in your design plan, the contents of this section are crucial to the design process. Without them, you won't have a visual record of how each game screen will look, work, or tie together.

NOTE: Most of the programs mentioned back in Chapter 6 are excellent for this purpose.

The Graphics Specification (required)

The graphics specification details the graphic elements needed for the game in question. Specifically, it should contain such information as:

- Game creative statement
- Artwork orientation
- Target platform
- Estimated object count
- Artwork screen resolution and playfield size
- Artwork color depth
- Artwork file format(s)
- Artwork filenames scheme
- Artwork color palette
- Artwork gamma level
- Artwork object dimensions
- Frames per object

- Object actions and facings
- Game text font(s)
- Miscellaneous art direction

For obvious reasons, the graphics specification section is the largest and most important part of your design plan. Not only will you rely on it, but so too will the other members of the development team. Therefore, it's crucial that you make this area as detailed as possible in order to avoid any misunderstandings between team members and to ensure that you get the results you want.

Game Creative Statement (required)

This is a creative statement that describes the overall style, feeling, and personality expected of the artwork (i.e., realistic, cartoon-like, happy, gloomy, dark, serious, etc.). The statement should be concise yet sufficiently descriptive to get the point across. Two to three sentences should be more than adequate for this purpose.

Example:

The artwork in *Disasteroids* will be hyper-realistic in style. All objects will appear as detailed as possible and will be rendered using colors that support this effect to the fullest.

Artwork Orientation (required)

This section describes the physical orientation and perspective of the artwork (i.e., above-view, profile-view, etc.). All that is necessary is a brief, one- or two-sentence statement.

Example:


Disasteroids will employ an above-view perspective. In other words, all game objects will be rendered as if the light source is shining directly above them.



NOTE: It might help to include a few simple sketches of objects rendered in the described orientation in order to illustrate the concept.

Target Platform (required)

This section simply describes the intended target platform of the game. It's important to include this information since it can influence the RGB values selected for the palette as well as the file formats used to store the associated game artwork.




NOTE: It's also necessary to mention the target platform just in case this information wasn't originally included in the design plan's game summary section.

Example:

Disasteroids is intended to run on all Windows 95, 98, and NT 4.0 systems.

Estimated Object Count (required)

This section provides an estimated count of the overall number of graphical objects (sprites and backgrounds) that will need to be designed and created for the game. The object must match the game object inventory described earlier in the design plan.




NOTE: Since the game is still in the planning stage, the object count is not definitive. You may find yourself adding or removing game objects as your design plan evolves.

Example:


It's estimated that *Disasteroids* will require a total of 35 different graphic objects to be designed and produced.

Artwork Screen Resolution and Playfield Size (required)

This section details the recommended or desired screen resolution in which to create the game's artwork (i.e., 320x240, 640x480, etc.) and the size of the game's play area.



NOTE: This information should only be included here after an initial consultation with a programmer. This is due to potential technical issues that might crop up when using a particular screen resolution.



NOTE: The playfield size usually only applies to systems that using a windowing operating system. Therefore, it wouldn't be defined for a game that's intended to run on DOS. It can also be left blank if you indicate that the game will run fullscreen, i.e., DirectX 640x480 fullscreen, etc.

Example:

All of the artwork in *Disasteroids* will be created in a screen resolution of 640x480.

Disasteroids will use a screen window that is 400x380 in size. This will remain constant for ALL game screens, regardless of their type.

Artwork Color Depth (required)

This section describes the color depth that the game's artwork will be created in (i.e., 16 colors, 256 colors, thousands of colors, etc.). Among other things, the information provided here will help the programmer determine whether or not certain special color effects are possible.



NOTE: This information should only be included here after an initial consultation with a programmer. This is due to potential technical issues that might crop up when using a particular color depth.

Example:

All of the artwork in *Disasteroids* will be designed in 256 colors (8-bit color).

Artwork File Format(s) (required)

This section details the preferred or acceptable file format(s) to store all of the artwork in (i.e., BMP, PCX, etc.).

This information should only be included here after an initial consultation with a programmer. This is done to avoid any potential technical issues that might crop up when using a particular file format.

Try to provide as much information here as possible, including listing any subformats of the selected graphics format. Including this information helps to flag any potential compatibility problems with your software.



NOTE: Be sure to be explicit in mentioning the selected graphics file format for each type of artwork screen you use. For example, you might choose to store your sprite screens in the BMP format but use TGA for your title screens.

Example:

All of the artwork in *Disasteroids* will be stored using 8-bit, RLE format Windows BMP files.

Artwork File Naming Scheme (required)

This section details the file naming convention that you intend to use for your artwork. You should break out the naming scheme for all game object types separately. For more information on this issue, refer to Chapter 4.

Example:

Sprites:

- ani_player.bmp (main player)
- ani_alien.bmp (alien objects)
- ani_asteroids.bmp (asteroid objects)
- ani_effects.bmp (explosions, shots, and missile objects)
- ani_assorted.bmp (miscellaneous game objects such as title screen animations, etc.)

Backgrounds:


- bkg_level1.bmp (level one background)
- bkg_level2.bmp (level two background)
- bkg_level3.bmp (level three background)

Other:

- mnu_buttons.bmp (buttons for menu and help screens)
- mnu_assorted.bmp (miscellaneous menu screen objects)

Artwork Color Palette (required)

This section describes the preferred color palette(s) to be used in the game. This information should include a complete list of RGB values, the transparent color value, and any reserved or off-limit palette entries. You may also opt to include palette format information if the programmer requires you to supply the color palette in a separate file.



NOTE: Be sure to provide all palette definitions using this format: palette entry: RGB value. For the transparent color and the off-limit/reserved colors, you only need to provide the palette entries affected.

Example:

Color Palette Definitions:

Index:	R,	G,	B
0:	0,	0,	0
1:	128,	0,	0
2:	0,	128,	0
3:	128,	128,	0

```

4:  0,  0, 128
5: 128,  0, 128
6:  0, 128, 128
7: 192, 192, 192
8: 192, 220, 192
9: 166, 202, 240
10: 255, 219,  0
11: 219, 167,  0
12: 183, 127,  7
13: 147,  87,  7
14: 111,  59,  7
15:  75,  35,  7
16: 255,  11,  15
.  .  .  .
.  .  .  .

```

Transparent Color: Index 254 (RGB: 0,0,0)

Off-Limit or Restricted Colors:

None aside from the normal 20 Windows reserved system colors.

Other:

- There are no palette entries reserved for color cycling.
- Palettes will also be supplied as Microsoft .PAL files.

Artwork Gamma Level (required)

This section lists the gamma level at which the graphics will be created.



NOTE: This information can be important, especially if the artwork is created on a different platform than the game's target system, i.e., you design the graphics on a Macintosh while the game is slated to run on Windows.


Chapter 2 provides more information on estimating the gamma level for your particular system or platform.

Example:

The graphics for *Disasteroids* will be designed on a Windows PC using a gamma level setting of 2.2.

Artwork Object Dimensions (required)

This section lists the maximum and/or recommended pixel dimensions (grid square size) for all sprites and background images.



NOTE: This information should only be included here after an initial consultation with a programmer. This is due to potential technical issues that might crop up when dealing with certain development tools that might require game objects to be created in certain sizes.

Examples:

- Sprite Grid Square Dimensions: 64x64
- Background Image Dimensions: 400x380

Frames per Object (required)

This section details the number of frames of animation required for each game object. It's important to record this information in order to estimate how long it might take you to create the artwork for a given game. This information also helps you determine how smooth or coarse the individual animations for each object will be.


Refer to Chapter 9 for more information on animation frames.

Examples:

- **Player Sprite:** moving (2) - Total Frames: 2
- **Asteroid Sprite:** moving (8) - Total Frames: 8
- **Explosion Sprite:** exploding (13) - Total Frames: 13
- **Warp Signature Sprite:** warping (8) - Total Frames: 8

Object Actions and Facings (required)

This section describes all of the potential actions that each game object (sprite) will perform in the game (i.e., walking, running, shooting). As part of this information, you should include the number of facings that each game object requires (i.e., left, right, top, bottom). It's important to record this information in order to estimate how long it might take you to create the artwork for a given game.



NOTE: Try to be as detailed as possible with regard to this section and be sure to include the total number of actions for your records.

Example:

- **Player Sprite:** moving (8) - left, right, upper-left, lower-left, up, down, upper-right, lower-right



NOTE: You may find it helpful to indicate how each object facing maps to a specific game control. For example, you can include a sketch of a mouse movement, keyboard press, or joystick movement that produces each object facing.

Game Text Font(s) (optional)

This section describes the preferred font and text style to be used in the game (if applicable). Most games default to the currently active system font (i.e., 8x8, 8x16 if DOS or TrueType if Windows or the Macintosh, etc.). This information applies to both the fonts used for housekeeping tasks such as displaying the game's score and the font style that appears on the game's title and menu screens.

Always make sure you provide the exact font names for each font you use. Doing this will help minimize confusion during implementation.

Also, make distinctions between what should be rendered in graphic text (i.e., bitmapped font) and what should be rendered in system text. There's a huge difference in quality between the two and system text is usually not something that you can change since it's under program control.

Finally, be sure to include the X and Y pixel dimensions for each custom bitmapped font you intend to use as well as a list of all of the characters this font includes.

Examples:

- **Housekeeping Font:** *Disasteroids* will use 10-point, aliased Arial for all game screen indicators such as level and score.
- **Menu Screen Font:** *Disasteroids* will use 18- and 24-point, anti-aliased Futura for all graphic text.
- **Title Screen Font:** *Disasteroids* will use 12-point, anti-aliased Futura for all graphic text.
- **Custom Bitmapped Font:** 16x16 - includes A-Z, a-z, 0-9, and !\$+-. ,?()=:

Miscellaneous Art Direction (optional)

This section includes any additional art direction that was left out in the previous design plan sections.



NOTE: Due to the importance of the graphics specification section of the design plan, don't proceed any further if any required information is missing. Even missing a detail as small as the number of frames in animation required per sprite can throw a big wrench into the works of a game project. Also, never guess or make assumptions about any item mentioned here


since you might guess wrong. If a certain piece of information is missing, track it down before proceeding. It may lengthen the design process but it helps reduce the likelihood of problems during the project. It is better to be safe than sorry!

Technical Restrictions and Stipulations (optional)

This section of your design plan discusses any technical restrictions imposed by the game and/or the development environment.

For example, this section might mention fixed sprite size limits or color use restrictions that have the potential to adversely impact how you create and implement your artwork. It's very important that this information be obtained and detailed as early in the project as possible before you start drawing your graphics! This is necessary because you don't want to create extra work for yourself in case some technical issue crops up that renders your artwork useless for the project.

Typically, the programmer would convey such information during the start of the project; therefore be sure you discuss any potential caveats with him or her if you're not actually coding the game yourself.



NOTE: This section is optional because there may not be any technical restrictions that apply to your particular game project.

Example:

In order to maximize screen performance on slower systems, no object in *Disasteroids* should be larger than 128x128 pixels in size.

The Project Schedule (optional)

Most game projects, including those that are done in one's spare time, have some sort of schedule associated with them. A schedule helps to introduce the element of order into what is often a chaotic process. Without a schedule that establishes firm *milestones*, or special project-related deadlines, your game project could potentially drag on forever. To prevent this, use a schedule to help you impose deadlines for yourself as well as for other members of your development team. A game graphics schedule might include such items as:

- The estimated start date for the project.
- An estimated date for presenting the initial game artwork to the game's programmer(s).
- An estimate for how long it will take to revise the artwork as per programmer/other team member feedback.

- An estimated date for delivering revised artwork to the game's programmer(s).
- An estimate for how long it will take to complete the remaining game artwork.
- The estimated end date for delivering the final artwork to the game's programmer(s).

As a game graphics designer, it's important to define milestones for the project early on in the process in order to establish good work habits and to facilitate the delivery of a timely game. Also, be sure to establish realistic milestones by giving yourself sufficient time to complete the designated task, as there's no sense having a deadline if you can't meet it.

You should always base your schedule milestones on realistic estimates. This means if you think it will take you at least two weeks to create the artwork and animation for a particular game, don't say you can do it in one week. Often, your estimates will be wrong, especially when working on your first few projects. Therefore, the best advice I can give you here is to overestimate. That means, if you think you can complete a project in two weeks, give yourself three weeks. This will give you some slack time in case you run into some unforeseen snags (and you will) while working on a project.

Schedules are an invaluable resource for medium to large scale game projects and are best created using a tool that allows easy revisions and modifications. For our purposes, any spreadsheet program or text editor should be enough to create and manage your schedule. However, if you're interested in a more scaleable and robust solution, then I recommend using professional project management/schedule software such as *Microsoft Project* or *IMSI's Turbo Project* series.

In closing, remember that schedules are just estimates. This means that they can be adjusted. Don't be afraid to do so as long as you inform the other members of your development team about changes in the schedule so that they can make adjustments on their end. Hopefully, you won't make many adjustments as the original schedule you created was done with achievable deadlines. Still, at least you have the option to do so.



NOTE: This section is optional, especially if you develop games strictly for your own pleasure. However, even then you might want to use one anyway since it might give you the incentive to actually finish a project for a change!

The Game Glossary (optional)

The purpose of the game glossary is to tie up any remaining loose ends by defining the terms and vocabulary introduced in the design plan. Essentially, it serves as the development team's "rosetta stone" as it allows all team members to get on

the same page regarding the terminology used for all of the game's characters, objects, and events.

It's generally considered good practice to include a glossary at the end of each section if you introduce a lot of unique terms. This will help to minimize any confusion over terminology that carries over into other sections of the design plan. However, as design plans vary, feel free to place the glossary section where you best think it should go. The end of the design plan will suffice for most game projects.