



Chapter 12

Hands-on Arcade Game Project—*Fish Dish*

In this chapter, you'll learn about:

- ♦ Planning an actual game project
- ♦ Designing an online arcade game
- ♦ Designing simple sprites
- ♦ Designing simple backgrounds
- ♦ Designing simple title screens

For much of the last 11 chapters, this book has focused almost exclusively on the theory, tools, and technology behind creating arcade-style game graphics. Now it is time to take what you have learned up to this point and apply it to a real game project. Therefore, in this chapter, we will actually conceive, spec, and design the artwork and animation for a maze/chase game called *Fish Dish*.

Fish Dish is an arcade game in which you control a small fish that must eat other fish to grow while at the same time avoiding becoming the meal of a larger fish.

The simple nature of this game makes it the perfect platform for discussing a variety of important game design topics, including:

- Basic online game design
- Establishing an artwork style
- Optimal color selection and usage
- Understanding graphics orientation
- Game character design
- Title and background screen design

Getting Started

Before proceeding with the *Fish Dish* tutorial, please be aware that this chapter makes extensive use of previously discussed concepts. Therefore it is suggested that you familiarize yourself with topics such as *color theory*, covered in Chapter 7, *color palettes*, covered in Chapter 8, *sprites and animation*, covered in Chapter 9, and *planning*, covered in Chapter 11.

Moreover, this chapter also assumes that you are comfortable with the graphics creation tools discussed in Chapter 5. Specifically, this chapter assumes that you fully understand how and, more importantly, when to use:

- The Pencil tool
- The Brush tool
- The Line tool
- The Smear tool
- The Fill tool
- The Zoom tool
- The Selection tool (including Cut, Copy, Paste, and Flip operations)
- The Lasso tool (including Cut, Copy, Paste, and Flip operations)
- The Color Selection tool
- The Eye Dropper tool



NOTE: This tutorial was designed to be application agnostic. In other words, all of the tips, tools, and techniques described in this chapter should work regardless of the actual graphics software being used.

The Three Phases of *Fish Dish*'s Design

To make it easier to follow, the *Fish Dish* design tutorial is divided into three phases. These are:

- **Phase I: The design analysis**—During this phase, we examine the game's concept, identify important design and technical issues, and discuss how to arrive at a suitable design plan. The purpose of this phase is to develop a solid approach to the game's design and gain a better understanding of the overall issues and problems that might occur over the course of the game's graphic development.
- **Phase II: The design plan**—During this phase, we document and organize all of the game's various components. The purpose of this phase is create a master document that will serve as a reference and road map for the rest of the graphic design process.
- **Phase III: The design execution**—During this phase, we actually implement the game's artwork and animation according to the principles and specifications laid out by the previous two phases. This is the most involved section of the tutorial and where you get to see how the various elements in a game are created and fit together.

All three phases are interdependent and as such, no single phase can really exist or occur without the others. Skipping one or more of these phases may shorten the design time required by a game but does so only at the risk of introducing a number of problems in the process.

Phase I: The *Fish Dish* Design Analysis

The purpose of the *Fish Dish* design analysis is to identify the most important issues that can negatively impact the successful design and implementation of the game. As such, there are several design issues to consider and resolve. These include:

- Differentiating game objects
- Choosing a design style
- Characterization
- Online game issues
- Determining the order of element creation

Once a solution to each of these is determined, we can safely proceed to Phase II.

Differentiating Game Objects

Practically all arcade-style games, including *Fish Dish*, require the player to control an on-screen character, which must avoid a variety of other on-screen objects and obstacles in order to achieve a certain goal.

One of the biggest problems such games face is the player's ability to easily tell the game objects apart. In many cases, the problem is so bad that the player often has trouble recognizing and locating the on-screen character he or she is supposed to control! This situation results in general player confusion and makes the game unnecessarily difficult to play.

Although this is a potentially serious problem, it is surprisingly easy to avoid. The secret to preventing it is through the correct use of what designers call *differentiation devices*. Differentiation devices are simple graphical cues that can help game players distinguish between different game objects. It is the graphic designer's responsibility to add these cues to the game, which is why they are mentioned here. The most common differentiation devices are:

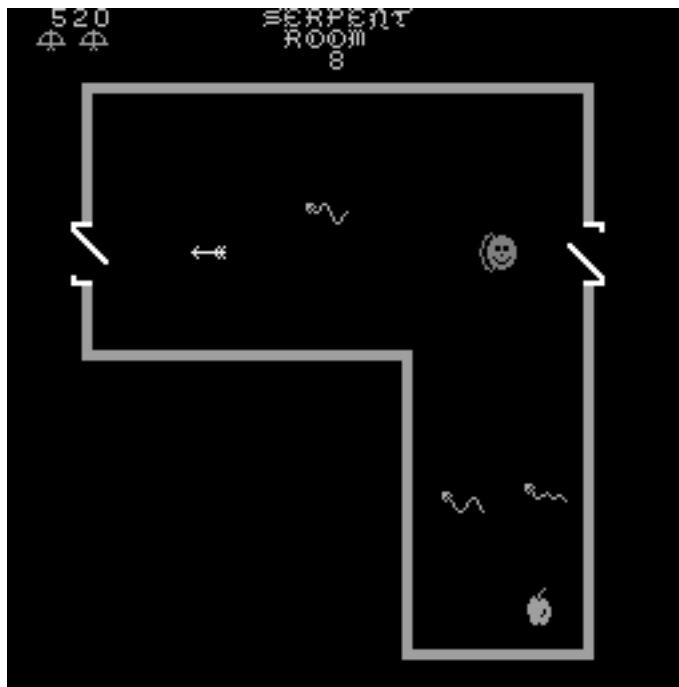
- Size
- Shape
- Color
- Position
- Animation

Size

Size is a popular method of differentiating game elements. Size works well in this role because it has the unique property of emphasizing the importance of one object over another. This is because most people instinctively assume that a larger object is more important in a game than a smaller object or vice versa.

For example, consider Figure 12-1. Notice how size is used as a differentiation device. In this scene, the circle object represents the player's on-screen character. Because it is up to twice the size of any other object on the screen, there is little doubt in the player's mind which object he or she controls.

A secondary use of size as a differentiation device is to increase the visibility of certain objects. This is because size can make specific objects easier or more difficult to see on the screen, depending on how it is used and applied.



Venture™

FIGURE 12-1: Example of Size as a Differentiation Device

Shape

Shape is another device we can use to differentiate between objects within a game. Shape helps players to visually establish a relationships between objects. Objects with similar shapes are assumed to have the same function or importance within a game. Meanwhile, objects with different shapes make it much harder for the player to make such associations.

A secondary use of shape is to make objects easier to recognize on a crowded game screen. For example, objects with distinctive shapes tend to stand out more than those with regular ones.

Finally, shape can help players better determine the form and function of a particular object. Consider the game pictured in Figure 12-2. In this game, the paddle's unique shape provides the player with important visual cues as to its purpose within the game.

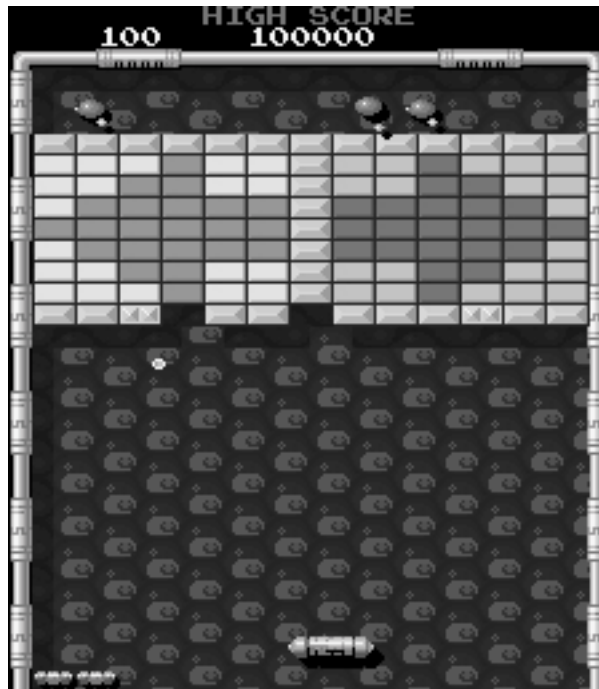
*Arkanoid™*

FIGURE 12-2: Example of Shape as a Differentiation Device

Color

Color is one of the best ways to differentiate game elements. As mentioned in Chapter 7, color has several unique and powerful properties. The properties can be used to great effect in identifying and separating objects on the game screen.

First, we can use color to attract and focus the player's attention, which can be used to direct the player's eyes to specific objects. Second, color can be used to classify specific game objects either by their function or by importance. Finally, it can help to speed up the player's ability to search and recognize different objects.

Figure 12-3 shows how color can be used to distinguish between different game objects. In this example, the bright object at the bottom of the screen is the player-controlled character. Notice how this object stands out when compared to the other game elements. Not only does this maximize the object's visibility to the player but it also serves to reinforce its role and importance to the player within the context of the overall game.

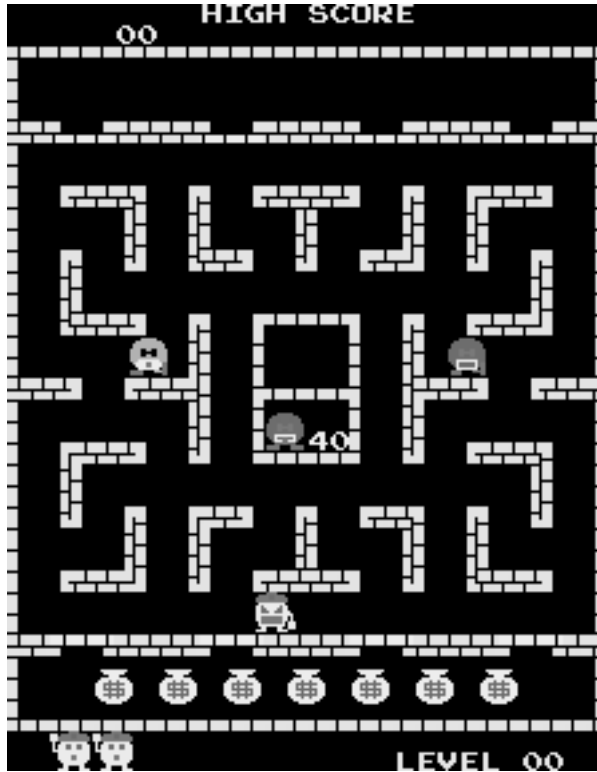


FIGURE 12-3: Example of Color as a Differentiation Device

Position

Position can also act as a differentiation device. In many arcade games, important objects are often positioned at specific sections of the screen to emphasize their importance or function. This is done for two reasons. First, placing objects at certain areas of the screen, especially the sides, corners, or center, ensures that they are inside the player's field of vision and hence are always seen. Second, the position of an object helps to emphasize its relationship with other objects. For example, objects that are close together give the appearance of being related and vice versa.

Figure 12-4 shows how position can be used to differentiate objects. In this example, the player-controlled character always starts a new game level positioned at the screen's center. Doing this places the object directly inside the player's field of vision, which ensures that the object is noticed. It also helps the player recognize the importance of the object since it is visibly separate from the rest.



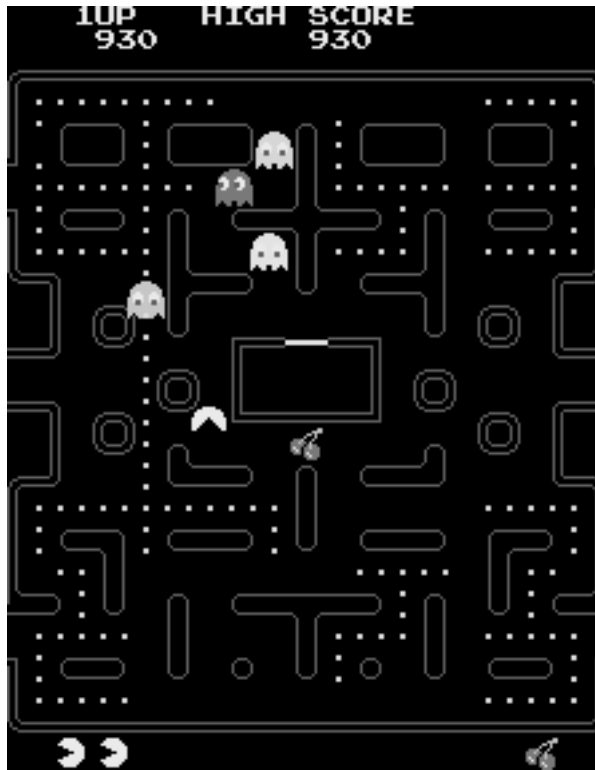
Robotron 2084™

FIGURE 12-4: Example of Position as a Differentiation Device

Animation

Animation is the last and one of the most powerful differentiation devices available to us. People naturally notice any object that is blinking, vibrating, pulsating, or otherwise dynamically changing. In games, these actions direct the player's eyes and provide important graphical cues as to the significance, function, and purpose of a given game element.

For example, the player-controlled character in Figure 12-5 is constantly moving. Visually, this serves as a clue to its importance, draws the player's attention, and clearly distinguishes it from the other on-screen characters.



Pac-man™

FIGURE 12-5: Example of Animation as a Differentiation Device

Table 12-1 summarizes the various differentiation devices that are available to us.

TABLE 12-1: Differentiation Device Summary Matrix

<i>Differentiation Device</i>	<i>Effects on Player</i>
Size	<p>Players are more drawn to larger objects over smaller ones because they are easier to see on the screen.</p> <p>Similarly, players tend to equate the importance of an object with its size, so they logically assume that a larger object is more important than a smaller one.</p>
Shape	<p>When objects have similar shapes, players tend to assume that they are similar in terms of importance and function.</p> <p>Players also tend to notice objects with unique or distinctive shapes over objects with common or simple shapes.</p>

<i>Differentiation Device</i>	<i>Effects on Player</i>
Color	<p>Players tend to associate objects with dissimilar colors as being different from each other.</p> <p>Players also find objects with dissimilar colors easier to identify and locate on-screen.</p> <p>Finally, objects with dissimilar colors tend to attract and hold the player's attention over long periods.</p>
Position	<p>Players tend to focus better on objects that are placed conveniently within their field of vision.</p> <p>In addition to making objects easier to see, the position of an object reinforces its importance and relationship to other objects.</p>
Animation	<p>Players tend to be drawn to objects that move over those that do not. This helps to focus the player's attention, emphasize the function of the object, and provide other visual clues as to the relative importance of the object in question.</p>

Commentary: Differentiation Devices in *Fish Dish*

The best arcade games actually combine one or more of these differentiation devices to ensure that there is little or no chance for player confusion when distinguishing between game elements.

For *Fish Dish*, we used size, color, and position as differentiation devices. Table 12-2 describes how and why these properties were used in the game.

TABLE 12-2: Differentiation Devices to Be Used in *Fish Dish*

<i>Differentiation Device</i>	<i>Comments</i>
Size	<p>The later levels of <i>Fish Dish</i> use size to differentiate between safe and dangerous game characters.</p> <p>Game objects that are small or equal in size to the player-controlled character are safe, whereas game objects that are larger than the player-controlled character are dangerous.</p>
Color	<p>Throughout <i>Fish Dish</i>, we use color to help the player visually distinguish between different game objects. In addition, the player-controlled character will be rendered in a distinctive color scheme to make it easier to locate and follow during the course of play.</p>
Position	<p>The player-controlled character is to be positioned at the center of the game screen in order to increase its visibility.</p>

Choosing a Design Style

Most dictionaries define *style* as “a combination of distinctive features of artistic expression or execution characterizing a particular group, school, or era.”

Although Chapter 7 briefly touched on the concept of design style as it pertains to color use, this chapter will examine design style as it relates to the actual appearance of your artwork.

A design style is one of the most powerful tools at your disposal. It helps you to define the appearance, theme, and audience for your game. In addition, it gives your artwork life and conveys a range of emotions from fear to nostalgia in the player, which ultimately reinforces their interest.

This being said, the process of choosing a design style for your game artwork is not something to be taken lightly. The design style you choose can have a dramatic effect on the visual aspects of your artwork from the size and shape you give your objects to the colors you use. Furthermore, once selected, a design style must be consistently carried through and applied to all of your game’s graphic elements; otherwise you risk undermining its power.

To help you determine which design style to use for your game project, this section of the chapter examines three of the most important stylistic factors:

- Audience and emotional appeal
- Visual characteristics
- Difficulty of implementation

Audience and Emotional Appeal

It is very important to realize that a game is just an idea until you define its style. As you may recall from Chapter 8, all arcade games generally fall into one of three design styles: cartoon, retro, and realistic. There are others but these tend to be variations of the three types mentioned. Each design style can give your game a different look and feel depending on the audience segment you are trying to reach and the general theme of the game. For example, are you trying to rekindle childhood memories with your game? If so, then consider designing your artwork using a retro style. Do you want to make your game funny? If so, then consider using a cartoon design style.

Tables 12-3 and 12-4 classify the three main design styles according to these facts. Such information is useful when trying to match up a game’s design with a particular demographic group and game theme.

TABLE 12-3: Design Style Demographic Appeal Matrix

Style Type	Adult Men (Ages > 21)	Adult Women (Ages > 21)	Young Men (Ages 12-21)	Young Women (Ages 12-21)	Primary Emotion(s) Associated with Design Style
Cartoon		✓	✓	✓	Cute, friendliness, warmth, familiarity.
Retro	✓				Nostalgia, familiarity, simplicity.
Realistic	✓		✓		Realism, elegance, and seriousness.

Although it may not be your primary consideration, you should never ignore or underestimate the emotional connection the player has with a specific game style when deciding how your game will ultimately look.



NOTE: Please use this matrix as a general guide only. It is based on observations made of players of various types of arcade-style games. For example, games with cartoon-like design elements and styles tend to appeal to women who are often put off by computers and technology. Meanwhile, both adult men and young men seem to prefer games with realistic design styles because they mirror the world around them.

TABLE 12-4: Design Style Theme Matrix

Style Type	Humor	Adventure and Fantasy	Science Fiction	Action	Horror and Mystery
Cartoon	✓				✓
Retro	✓	✓	✓	✓	✓
Realistic		✓	✓	✓	✓

Visual Characteristics

In addition to their emotional and audience appeal, each of the three design styles is characterized by a unique appearance. For example, games that employ a realistic design style look elegant and tend to mirror reality. Such games feature artwork that is designed to support the player's belief system through a careful interplay of color, size, patterns, and general design quality.

Table 12-5 summarizes the visual characteristics of each arcade game style.

TABLE 12-5: Design Style Visual Characteristic Matrix

Style Type	Visual Characteristics	Comments
Cartoon	<p>Tend to feature objects with large areas of solid color contained by sharp, defining border elements.</p> <p>Objects often seem “cute” or simplistic, and mirror elements from comics or television cartoons.</p>	<p>A properly executed cartoon-style arcade game can score big points with players due to its inviting visual appeal.</p> <p>Games with cartoon design styles tend to translate very well to the online world since the Java/Netscape palette actually provides the appropriate color schemes these games need.</p>
Retro	<p>Tend to feature objects with large areas of solid color and no border elements.</p> <p>Objects appear coarse, crude, and blocky, and mirror elements that appeared in video games during an earlier age.</p>	<p>A relatively risky design style to use for games for mass audiences since many users will perceive games rendered in this style to be of inferior quality after being exposed to the large numbers of realistic games.</p> <p>Games with retro design styles translate well online since they use relatively simple color schemes.</p>
Realistic	<p>Tend to feature objects with many small areas of color contained by light, defining border elements.</p> <p>Objects appear detailed and realistic, and mirror the elements that one sees in everyday life.</p>	<p>Many players will identify with realistic design style arcade games. However, at the same time, players may be overly critical of a game that fails to properly maintain an illusion of reality.</p> <p>Games with realistic design styles generally do not translate well online since the Java/Netscape palette color palette is too limited in terms of the colors it provides.</p>

For obvious reasons, the visual elements of a game’s design style are paramount to its success. Not only does it influence how the player sees and perceives your game, but it is often associated with quality as well. In fact, many games are considered to be good simply because they look good. However, this being said, it is crucial that you successfully execute whatever design style you choose. Otherwise, you risk disappointing players and alienating them from your game.

Difficulty of Implementation

One consideration that is seldom mentioned but just as important as emotional appeal or aesthetics is the difficulty associated with implementing a given arcade game design style.

Each design style requires a different competency level to implement. Far too often, designers over-reach their abilities and attempt to design games using a design style they are simply not skilled enough to handle. For example, many beginners attempt to create realistic-style artwork before they fully master basic artistic techniques. This results in subpar work and taints what might be an otherwise good game.

Table 12-6 rates each design style according to the relative difficulty and artistic skills required to implement it.

TABLE 12-6: Comparison of Design Style Implementation Difficulty

Style Type	Relative Difficulty	Designer Skill Level	Comments
Cartoon	Generally easy to implement but somewhat difficult to perfect.	Beginner to intermediate	Requires a firm knowledge of color, shape, and basic animation techniques.
Retro	Very easy to implement and perfect.	Beginner	Requires rudimentary artistic skills and animation techniques.
Realistic	Difficult to implement and very difficult to perfect.	Intermediate to advanced	Requires a complete mastery of color, shape, light, and shadow, and advanced knowledge of animation techniques.

With regard to implementing a design style, the rule is this: never attempt to use a particular design style for your game unless you are completely comfortable with the skills and effort needed to properly use it.

Commentary: Design Style Selection in *Fish Dish*

The artwork featured in *Fish Dish* uses a cartoon design style throughout.

There are several reasons for this. First, as a close clone of an existing game that already incorporates many cartoon-like elements, it makes sense to carry over this particular design style. Second, using a cartoon design style guarantees that the game will have a larger appeal than if it used one of the other design styles. This is due to the fact that cartoons appeal to both children and adults alike. Third, aesthetically speaking, games with cartoon-based design styles tend to work well with the Java/Netscape color palette. This means that we can create relatively attractive and colorful artwork without having to worry about accurate shading and rendering of objects. Finally, games with cartoon design styles tend to be relatively easy to design and therefore are suitable for both beginning and advanced users.

Characterization

After a suitable design style is selected, you must decide on the *characterization* of your game elements. Characterization is what gives your game characters life. In effect, it is analogous to personality. However, for our purposes, characterization refers to the stylistic and emotional interpretation of one or more game characters. It can include anything from the color of the object to the size of the object's eyes. These elements work together to instill a sense of life into a given character. Instinctively, players pick up on these elements and identify with them. For example, bad characters look mean, good characters look cute and friendly.

Unfortunately, instilling personality in your artwork is not a skill that is learned; rather it is developed through time, experience, and practice. Over time, many designers develop a specific drawing style that they are able to incorporate into the various objects they create.

Table 12-7 provides a list of common mechanisms you can use to introduce personality into game characters.

TABLE 12-7: Common Methods for Representing Characterization in Arcade Games

<i>To make the character appear...</i>	<i>Try this...</i>
Cute	Add large, bulbous eyes. Large eyes seem to convey a sense of wackiness in many characters. This makes characters more appealing and less threatening to many players when compared to characters that do not have large eyes.
Friendly	Add a broad smile and large eyes. People (and game players) are naturally attracted to smiling. They find it comfortable and familiar. Often, you will find what works in the real world has application for games.
Happy	Add a broad smile with wide-open eyes and raised eyebrows (if applicable to the object in question).
Mean	Add a grin and/or down-turned eyebrows that hang low over the eyes.
Old	Add white hair or wrinkles. Hair color and wrinkles imply aging in almost any context.
Sad	Add a down-turned grin.
Smart	Add glasses with large eyes. Traditionally, glasses are associated with being learned.
Stupid	Add large eyes with small pupils placed close together. Traditionally, characters with such eyes have been considered unintelligent.

Other personality traits can be added by simply studying those things around you. You would be surprised at how much video game characters tend to use real-life expressions.

It is important to realize that characterization is not just limited to expressions. The color you give certain game objects plays an important part in the player's interpretation and understanding of a game object characterization. Color could be used to enhance an object's characterization, for example, a red devil or a black knight. Chapter 7 covers this topic at length. Please refer to it when deciding on the characterization for your game.

Commentary: Characterization in *Fish Dish*

Fish Dish uses characterization extensively. All of the game's objects exhibit it through a variety of visual means. For example, to make a particular game object look stupid, it was given large eyes and a blank expression. This cues the player into the fact that that character is not considered very bright.

Similarly, color is used extensively to reinforce the characterization of *Fish Dish*'s characters. For example, the main character is a salmon and is pink. Another character is green and has a mean temperament.

Online Game Issues

Online games, compared to the average offline arcade game, require a bit more work and planning to design. Specifically, you must examine and address such issues as:

- The available display area
- Artwork file size issues
- Performance
- The color palette

The Available Display Area

Virtually all online games are played from inside a Web browser. Due to significant differences in how most browsers handle the display of content, most online games cannot count on being able to take advantage of the entire screen area that is available at a given screen resolution. For example, a browser running at a screen resolution of 800x600 does not translate into the full 800x600 resolution being available to the online game running inside of it. Often, there will be as much as 50% less screen area available to an online game at any screen resolution.

Having less usable display area available means that your game's objects, title screens, and background screens have to be smaller than if they were appearing in

a comparable offline game when running at the same screen resolution. Although this places some limits on your creativity and makes designing the game artwork more of a challenge, this issue can be dealt with as long as you know how much physical area you actually have to work with.

Table 12-8 provides some estimates of the available browser display area at common screen resolutions.

TABLE 12-8: Estimated Available Browser* Display Area at Common Screen Resolutions

<i>Screen Resolution</i>	<i>Estimated Available Display Area (Minimum)</i>	<i>Estimated Available Display Area (Maximized)</i>
640x480	470x300	635x380
800x600	470x430	795x500
1024x768	470x600	1019x668

* Applies to 3.0 and 4.0 generation browsers only.

NOTE: For the purposes of this table, minimum represents the available display area when browsers are first started while maximized represents the available display area when the browser window is maximized.

Figure 12-6 illustrates the difference in available display area in online games compared to offline games.

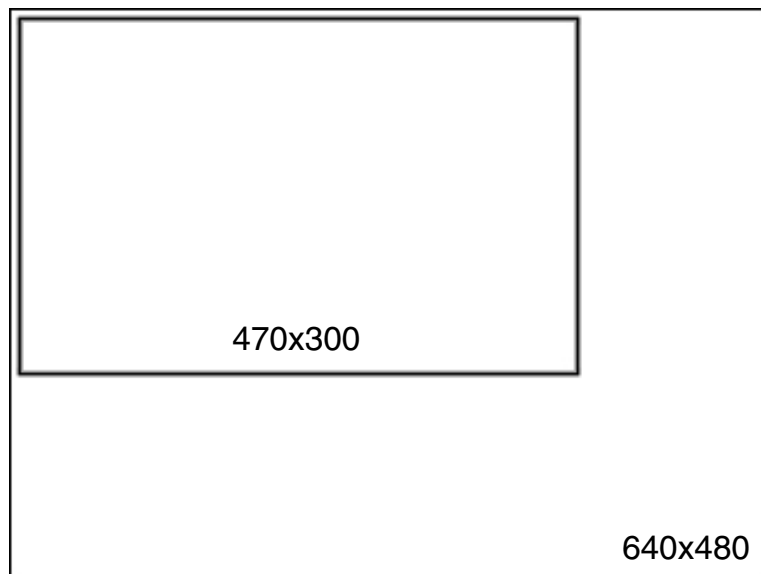


FIGURE 12-6: Comparison of Available Display Area at 640x480

In order to ensure that your online game is viewable by as many people as possible, it is strongly recommended that you always design your game's play area around the smallest available display area. According to Table 12-8, this means restricting your game screens to a maximum size of 470x300.

Commentary: Display Area Issues in *Fish Dish*

Because of the previously mentioned issue, all game screens in *Fish Dish* are restricted to a maximum size of 470x300 to ensure that they are viewable by the largest possible audience on all platforms.

Artwork File Size Issues

Despite rapid advances in communication technologies, most people do not have very fast Internet access, especially outside the United States. In fact, the vast majority of users are still connecting to the Internet with 33.6 Kbps and 56 Kbps modems. Therefore, online game developers face a nasty dilemma: they have to create games that are compelling enough to attract players, yet small enough to access and play through relatively slow Internet connections.

Unfortunately, this is not always easy to do. For one thing, arcade-style games tend to be very graphic intensive. This makes for larger, more complex programs since next to digital sound, graphic images are the most expensive component of an arcade game in terms of program size and disk space used. Second, there is the issue of the user/player's experience. With a typical online game weighing in at 200-300 KB in size, it can take most players over a minute to download and play it. The fact is that we live in an impatient world. Studies have shown that the longer a person has to wait for the computer to perform an action the more likely he is to become distracted. Therefore, it is in our best interest to do what we can to reduce the size of the game and shorten the player's wait before he can start playing.

Fortunately, we have some tricks up our sleeve. As it turns out, one of the best ways to reduce the size of online games is to use image compression since creating smaller graphic files ultimately means creating smaller games. Although there are several compressed graphic file formats available, only three are of any use for online games. These are:

- GIF
- JPEG
- PNG

Of these, GIF is the best all-around choice due to its solid compression ratios and almost universal application compatibility.

Once you have selected a compressed file format, you should try to keep every piece of game-related artwork down to reasonable file sizes. Table 12-9 provides

some loose guidelines for specific limits to set for common game graphic elements.

TABLE 12-9: Online Arcade Game Artwork File Size Guidelines

<i>Artwork Type</i>	<i>Suggested Maximum File Size (in KB)</i>
Background Screen	< 25*
Menu Button/Element	< 1*
Single Animated Sprite	< 2.5*
Single Static Sprite	< 1.5
Status Indicator	< 8*
Title/Menu Screen	< 18*

* Actual results may vary depending on the compression scheme used and the artwork involved.

As noted, the numbers in this table will vary depending on the artwork and the image compression scheme used. This is because different compression schemes only work well on certain types of images. Table 12-10 summarizes these issues.

TABLE 12-10: Image Compression Issues and Usage Recommendations

<i>Compression Scheme</i>	<i>Issues</i>	<i>Recommendations</i>
LZW (GIF)	Works best on images that contain simple shapes and large areas of solid color. Images that contain large numbers of colors and complex patterns do not compress nearly as well. Especially optimized for smaller objects that cannot afford to lose any detail.	Best when used on cartoon- or retro-styled artwork.
LZ77 (PNG)	Generally similar to LZW.	Best when used on cartoon- or retro-styled artwork.
JPEG (JPEG)	Works best on images that contain complex shapes and patterns. Especially optimized for relatively large objects that can afford to lose detail.	Only use on photographic or hyper-realistically styled artwork.

Ultimately, such issues can affect the stylistic appearance of your artwork. For example, you may have to choose to design your artwork using a cartoon style instead of a realistic style in order to maximize the file compression savings offered by a particular compression scheme.

As a rule, you should limit your online game's artwork to a maximum of 25% of the game's total file size and often you will want to limit them to less. Therefore, if the game's final size is 300 KB, no more than 75 KB of it should be artwork. If it

is, you will need to remove some elements or find ways to crunch down the artwork even further. Table 12-11 provides some examples of this limit applied to games at various sizes.

TABLE 12-11: Maximum Artwork Size Matrix

<i>Final Game Size</i>	<i>Maximum Artwork Size</i>
64 KB	16 KB
100 KB	25 KB
150 KB	37.5 KB
200 KB	50 KB
250 KB	62.5 KB
300 KB	75 KB

To determine whether your game is meeting or exceeding the maximum suggested sizes for its artwork, you need to consider several variables, including:

- MT = The maximum target size for the online game
- GC = The size of the game's code without graphics or sound
- SS = The amount of space required by the game's sound effects and/or music
- TG = The size of all the game's graphical elements

Consider this example:

MT = 300 KB
 GC = 100 KB
 SS = 100 KB
 TG = 75 KB

$$GC (100) + SS (100) + TG (75) = 275$$

$$MT (300) - 275 = 25$$

This means that you are 25 KB under the target game size, which is great. This means that the final game will actually be smaller than originally thought and will download faster. However, in most cases, the opposite will be true and you will find that your game is actually larger than originally anticipated.

If this happens, look at your artwork file sizes and see which items can be pared down since for every 2 KB you eliminate from your game, you can save the player up to one second of download time. For larger online games, this can really add up, especially for those with really slow Internet connections.

Commentary: File Format Issues in *Fish Dish*

Fish Dish uses the GIF format to store its artwork because it provides good image compression and widespread compatibility with different programming languages, graphics tools, and computer platforms.

GIF is also a good choice for other reasons. As it happens, GIF compression works particularly well on images that contain large areas of solid color. Cartoon-style artwork usually falls into this category. Consequently, we enjoyed significant compression savings by using this file format.

Performance

Performance, or how fast and smooth a game plays, affects an online game much more than it would an offline game. The primary reason for this is due to how online games are written.

Practically all online arcade games are written using either Java or *Shockwave*. We briefly discussed Java in Chapter 1 but not *Shockwave*. For those of you not familiar with it, *Shockwave* is a compression engine for games created with *Macromedia Director*, a powerful, multimedia authoring tool. The *Shockwave* engine compresses *Director* applications so that they can be efficiently used on the Web.

Java and *Shockwave* are popular with online game developers for three reasons. First, games created with them can reach a very large audience because Java comes built into many Web browsers while the *Shockwave* player comes bundled with many computer systems. Second, they are portable across multiple platforms. With the appropriate support (a Virtual Machine in the case of Java or a special player in case of *Shockwave*) installed Java and *Director/Shockwave* content will run consistently regardless of the system it is running on. Finally, they provide programmers with many features that greatly speed up the development process. This means games can usually be developed much faster using these tools than with more traditional programming languages.

However, all of this flexibility comes with a price—speed. You see, both Java and *Shockwave* are highly *abstracted* development environments. This means that they execute commands through multiple layers instead of accessing the computer's hardware and operating system directly, which in turn limits performance. For example, in order for a Java game to move an on-screen object, it must send instructions to a corresponding Java function that performs this action. In turn, the Java function must then be translated into the appropriate Windows, Macintosh, or Linux operating system call and so on. In contrast, using a traditional programming language like C or C++ only requires one such translation rather than several to perform the same task. Therefore, games written using either of these technologies do not offer anywhere near the same level of performance as their offline counterparts.

To make matters even worse, games written in Java usually store their artwork as separate files. For games with many different objects, this additional overhead can further degrade performance. However, due to the way they are compressed, *Shockwave* games do not have this problem.

Table 12-12 compares the performance of Java and *Shockwave*.

TABLE 12-12: Comparison of Online Game Development Tool Performance

Online Programming Environment	Screen Performance Rating	Other Issues
Java	Mediocre to good. Depends on the game type, the complexity of the animation, and the number of objects.	Overall, performance depends on the version of Java and the implementation of the Java Virtual Machine. Game-related artwork is usually external to the game and is loaded when the game first runs, which can cause a noticeable delay.
Director/Shockwave	Mediocre to good. Depends on the game type, the complexity of the animation, and the number of objects.	Performance is tied to the speed of the player's machine and the version of the <i>Shockwave</i> player being used. Game-related artwork is usually internal to the game program and is loaded when the game first runs. The resulting performance penalty tends to be less than for Java because most, if not all, game artwork comes embedded in the game itself.

From a design perspective, there are several ways to enhance an online game's performance. These are:

- **Reducing the number of on-screen objects**—Fewer game objects means fewer items for the game to track, move, and manipulate. Limiting the number of game objects can help to speed up a game's performance.
- **Using simpler animation**—Simple animations (only a few frames) are far less resource intensive than complex animations (many frames). Using less sophisticated animations can improve game performance.
- **Reducing the size of on-screen objects**—Smaller sprites and game objects require less computing time to display, track, and move than larger sprites and game objects. Therefore, restricting the size of your game's objects can translate directly into faster overall performance.

- **Eliminating or minimizing background activity**—Background activity such as scrolling is a huge drain on a computer's resources. Removing such effects from your online game can go a long way in improving its performance.

Commentary: Performance Issues in *Fish Dish*

Fish Dish can be implemented using Java, *Shockwave*, or any other programming language for that matter. However, to ensure optimal performance regardless of the technology used, *Fish Dish* was designed with these considerations in mind:

- **Simple animation**—The need to keep game activity moving at a reasonable pace means that *Fish Dish* will use very simple animation. Specifically, *Fish Dish* uses the open/close and swing animation primitives, and several of the game's objects do not use animation at all. In addition, all game objects are limited to only a few animation frames each.
- **Static background screens**—The backgrounds used in *Fish Dish* are static in order to minimize non-essential animation and improve the game's overall performance.
- **Object size**—Most of *Fish Dish*'s game objects are small to medium in size. This was done to take advantage of the fact that smaller objects animate faster than larger ones.

These design modifications were made after a careful evaluation of *Fish Dish*'s various components so that we could maximize performance without dramatically diminishing the overall quality of the game.

Good performance is important to any arcade game, but you should never sacrifice the integrity of the game just to see a performance benefit. Rather, you should carefully weigh what is to be gained by removing or reducing the graphical elements of your game and what is to be lost. For example, although all of the design modifications to be implemented will contribute to *Fish Dish*'s performance, none of them actually detracts from the game as a whole. In other words, these changes can be made and *Fish Dish* will still be a good game. Therefore, only make design changes if they benefit the game without ruining the player's experience.

The Color Palette

All online arcade games use a common color palette known as the Java/Netscape palette. First described in Chapter 8, this palette has the ability to display artwork created with it on multiple platforms without color loss or dithering when in an 8-bit display mode. Although this property makes it very useful for online games, it has its share of issues to contend with, including:

- **Color availability**—Most color palettes give us 256 different colors to choose from and use. However, for various technical reasons, the Java/Netscape palette only gives us 216 usable colors. This leaves us with fewer available color choices and can (at times) place restrictions on our creativity.

- **Poor color variation**—As explained in Chapter 8, this palette has two major shortcomings with the colors it provides. First, there are too many saturated shades of primary colors and very few intermediate or tertiary colors. Second, there are only six shades of any one color available. This makes it useless for games that require artwork with complex shading.
- **Poor color harmony**—The colors contained in the Java/Netscape palette exhibit poor color harmony. This means that they do not work well together and give us few useful color combinations to use in our artwork.

Indeed, these issues definitely make using this palette in a game a challenge. However, one of the signs of a good designer is being able to determine how to best utilize the colors available. The key to doing this is by identifying which color combinations work well together and the most suitable styles of artwork to use with the palette in question.

It takes time and experience to really determine which colors work well with each other for a given palette. This is especially true when creating artwork for online games using the Java/Netscape palette. As it happens, there are only about 16 useful color combinations supported by this palette.

Shown in Table 12-13, these color combinations have been identified through trial by working on numerous game projects. They all provide sufficient diversity to allow you to render many different types of objects without being overly repetitious or boring. While other color combinations are indeed possible using this palette, they usually produce color combinations that are too bright, too dark, or do not provide sufficient contrast to make them effective in a game.

TABLE 12-13: Effective Color Combinations for Use in Online Games

<i>Color Combination</i>	<i>Colors (RGB Values)</i>
Combo 1	Dark Red: 153 0 0
	Maroon: 204 51 51
	Light Red: 255 102 102
	Pink: 255 153 153
Combo 2	Light Brown: 204 102 0
	Dark Tan: 204 153 0
	Sand: 255 204 51
	Pure White: 255 255 255
Combo 3	Dark Brown: 102 51 0
	Medium Brown: 204 102 0
	Light Tan: 255 153 0

<i>Color Combination</i>	<i>Colors (RGB Values)</i>
Combo 4	Magenta: 204 51 153 Medium Pink: 255 102 204 Hot Pink: 255 153 255 Pure White: 255 255 255
Combo 5	Dark Gray: 51 51 51 Medium Gray: 102 102 102 Light Gray: 153 153 153
Combo 6	Pure White: 255 255 255 Light Pink: 255 153 255 Dark Purple: 102 0 153 Light Purple: 153 102 255
Combo 7	Dark Green: 0 102 0 Medium Green: 0 153 0 Light Green: 102 255 0
Combo 8	Bright Green: 204 255 102 Medium Green: 51 153 51 Light Green: 153 255 102
Combo 9	Light Orange: 255 153 102 Dark Red: 153 0 0 Dark Pink: 255 102 153
Combo 10	Gold: 255 204 0 Light Brown: 153 102 0 Pale Brown: 204 153 102 Flesh: 255 153 102
Combo 11	Dark Blue: 0 51 153 Medium Blue: 0 102 255 Light Blue: 0 153 255 Sky Blue: 51 204 255
Combo 12	Dark Purple: 102 0 153 Light Purple: 153 102 255 Lavender: 204 153 255 Pure White: 255 255 255

<i>Color Combination</i>	<i>Colors (RGB Values)</i>
Combo 13	Dark Green: 0 102 0
	Medium Green: 51 153 51
	Light Green: 102 204 102
	Bright Green: 153 255 153
	Very Light Green: 204 255 204
Combo 14	Dark Olive: 102 102 51
	Medium Olive: 153 153 102
	Light Olive: 204 204 153
	Very Light Olive: 255 255 204
	Pure White: 255 255 255
Combo 15	Dark Magenta: 102 51 102
	Medium Magenta: 153 102 153
	Light Magenta: 204 153 204
	Very Light Magenta: 255 204 255
	Pure White: 255 255 255
Combo 16	Dark Cyan: 51 102 102
	Medium Cyan: 102 153 153
	Cyan: 102 204 204
	Light Cyan: 153 255 255
	Very Light Cyan: 204 255 255

The lack of more than six shades for any one color and the overabundance of primary colors in the Java/Netscape palette make it largely unsuitable for creating certain types of images. Instead, it is strongly recommended to only use this palette to create retro- or cartoon-style artwork. Coming to this realization will save you time and effort later.

Usually, the palette's suitability can be determined by examining the colors at your disposal. Specifically, look for essential "indicators" or factors that can negatively affect your creativity as:

- **Available shades and tints**—As mentioned in Chapters 7 and 8, a palette with too few shades and tints per unique color prevents it from being used to effectively render realistic objects.
- **Available color types**—Many objects, particularly realistic ones, require intermediate and neutral colors. A palette with too few of these types of colors effectively rules it out for rendering realistic or complex objects.
- **Available color intensity**—A palette with an abundance of highly saturated colors makes it largely unsuitable for rendering realistic or detailed objects. Similarly, a palette with too little intensity can have the same effect.

In addition, other factors can influence a palette's suitability, including:

- **Contextual requirements**—These are the color requirements of the object(s) in question, i.e., green sky vs. blue sky, etc. You may have no choice but to use certain colors because the context of the situation demands their use. This should be one of your most important considerations, especially when trying to mimic real-world characters and objects in your game artwork. A palette is worth considering as long as it meets your contextual requirements.
- **Aesthetic requirements**—These consider issues like which colors look best against certain backgrounds and with other objects. This is essentially the color harmony of the palette. If the palette provides your game objects with sufficient coverage, then the palette is worth considering.

On the positive side, the Java/Netscape palette can produce some very attractive game artwork when used properly. This means sticking to the color combinations described in Table 12-13 and only using them to create cartoon- or retro-style artwork. Because of this, the artwork in *Fish Dish* does not use a realistic design style.

Commentary: The Color Palette in *Fish Dish*

As an online game, we really had no choice but to use the Java/Netscape palette for *Fish Dish*. However, despite the palette's obvious disadvantages, the colors in it more than meet our contextual and aesthetic color requirements. This means that it will do the trick.

Determining the Order of Element Creation

One of the challenges of designing arcade game graphics is figuring out exactly what to do first. Do you design the game's characters and animation first? Do you create the backgrounds before the title screen? As you can probably appreciate, these are important questions, especially when you consider the fact that you may be under a deadline and only have a limited amount of time to create everything for your game.

There are a number of factors to consider in determining which part of your game project to tackle first. These include such things as:

- **The quantity of objects**—For obvious reasons, the number of objects in a game can have a big effect on whether you choose to work on this part of the game's artwork first. This is simply due to the fact that more objects will take more time to design.
- **The design style**—Certain design styles require more design effort than others. For example, games with realistic artwork are more complex than those with retro artwork, and so on.

- **The type of artwork**—Some artwork is simply more difficult and time-consuming to design than others. For example, animated objects will usually take more time to create than static objects.
- **The color palette**—The color palette used can significantly affect how long it takes to create a piece of game artwork. For example, some palettes impose specific limitations on the number or types of colors you can use. This will force you to spend more time coming up with workarounds.


Given these factors and issues, it is suggested that you approach creating your game objects in this order:

1. Game sprites
2. Game backgrounds
3. Game titles and menu elements

Game sprites and any animated objects should be created first because they tend to be the most complex as well as the most numerous parts of your game. Therefore, you're bound to spend the most time creating them. The sooner you get them done and out of the way, the better off you are.

Game backgrounds can be very complex but rarely as much as sprites. Also, there tend to be fewer of them so they should not take you nearly as long to design and create.

Finally, always create your title screen and menu screen elements last. This is because they are considered game icing. That is, they are embellishments to the game and not central to it. Your game sprites and backgrounds are far more important. Far too many designers make the mistake of focusing on these elements first when they should really be creating them last.



NOTE: If you are working with other designers, some of these steps can be done at the same time. For example, you can have one artist focus on creating the title screen while another focuses on designing the backgrounds. This is typically how larger, commercial game projects are done as this arrangement maximizes resources and productivity. However, the order of element creation presented here will work for pretty much any solo game project.

Commentary: The Order of Element Creation in *Fish Dish*

Fish Dish is sufficiently complex enough that it requires the bulk of the design time to be used creating the game's various game sprites. The backgrounds are very simple, and therefore only minimal time will be allocated to them. In addition, they will be done after the main game objects have been completed. The title screen, because it is an embellishment, will also be done last.

Phase II: The *Fish Dish* Design Plan

This section of the chapter documents the various technical details associated with designing the artwork and animation for *Fish Dish*'s game objects and elements.

The *Fish Dish* Game Summary

As discussed back in Chapter 11, it is the purpose of the game summary in our design plan to help us flesh out the overall creative appearance of our game (in this case, *Fish Dish*). In addition, this is the place where we define and establish such crucial game elements as the characters that we will need to eventually design.

Game Back Story

Poor Salmon Fishdie has just hatched and is very hungry. All alone in the deep blue ocean, Salmon must rely on his speed and wits to snag a meal and grow big and fat. Unfortunately, life in the wild isn't easy because the ocean is a tough place. Salmon must constantly be vigilant and watch out for bigger, more powerful fish and even worse—sharks! Can you help Salmon fill his belly without him filling someone else's?

Game Description/Game Concept

Fish Dish is a humorous online, maze/chase version of Darwin's "survival of the fittest" theory. In it, the player controls an on-screen character in the form of a small fish named Salmon Fishdie. The player must maneuver Salmon so that he can gobble up any fish that are smaller or equal in size to Salmon while avoiding larger, predatory fish that periodically appear. The more fish that Sal eats, the larger he grows. Unfortunately, for Salmon, his predators grow as well, making it progressively more dangerous for Salmon to survive.

The goal of *Fish Dish* is very simple: the player scores points by eating other fish. When the player exhausts all of Salmon's lives, the game ends.

Game Object Inventory

As designed, *Fish Dish* includes a number of different game objects. This group of objects contains the game's *core game characters*. Core game characters are game objects central to the theme or plot of the game and usually include the player-controlled character as well as the computer-controlled enemies. In *Fish Dish*, these objects include:

- **Salmon Fishdie**—The player-controlled character. Salmon starts the game as a small, round fish with two large eyes on the same side of his face. Salmon's

main ambition in life is to eat. As such, he travels the ocean looking for fish to eat that are smaller or equal to him in size while avoiding bigger and meaner fish. The more Salmon eats, the more he grows. In fact, Salmon can grow up to 7x his original size. However, even at his largest size, Salmon manages to retain the boyish good looks he had when he was a small fry. In other words, he looks more or less the same as he did at the start of the game. The player controls Salmon with the arrow keys and can move Salmon in four possible directions—up, down, left, and right. To help make Salmon easier for the player to spot in a sea of fish, he is colored pink. In addition, he is the only fish in the game that moves both his fins and mouth as he swims.

- **Salmon Fishdie Angel**—This divine, winged fish appears whenever Salmon “gives up the ghost” and becomes another fish’s meal. Salmon is inherently good so he gets to go upstairs to that big ocean in the sky and swap his fins for a pair of wings.
- **Bad Tuna**—A potential predator or meal, depending on Salmon’s size at the time of contact. Bad Tuna are a particularly tasty species of tuna that are known for their voracious appetites for Salmon. Bad Tuna are easily identified by their purple and lavender color scheme and large fins. Unlike other fish, Bad Tuna never swim alone.
- **Blue Angels**—A species of uncanny beauty, and like many of the fish described here, they are both a threat and meal to good ol’ Salmon. Blue Angels are triangular in shape and blue in color. Despite having large eyes for their size, Blue Angels are virtually blind, making them easy prey for a hungry, ever-growing Salmon.
- **Gold Diggers**—A small species of fish known for their brownish-gold coloring and tendency to feed at the bottom of the sea. Gold Diggers are very fond of smaller Salmon but the feeling is not mutual; Salmon hates the taste of Gold Diggers. However, if hungry, Salmon considers Gold Diggers to be excellent sources of protein, vitamins, and minerals and will eat them if given the chance.
- **Gray Sharks**—A small and notoriously cowardly species of Shark. Gray Sharks are sharks only in name, as they tend to startle easily. Although they often prey on young Salmon, Salmon just as often preys on them. Despite their meek demeanor, Gray Sharks freely roam the ocean and rely on their great speed to avoid danger.
- **Green Meanies**—Another potential predator and meal of Salmon’s. Plentiful and hearty, Green Meanies are one of Salmon’s favorite dishes. When not being hunted, Green Meanies are one of smaller Salmon’s greatest threats. Green Meanies are small, green, and round and are easy to spot no matter how many fish are in the sea. Green Meanies prefer to swim in the upper reaches of the ocean, near the surface where the water is warm.

- **Happy Clams**—A species of giant clams that are occasionally found on the ocean bottom. Although immobile, Happy Clams are impervious to attack due to their hard outer shell.
- **Head Hunter Fish**—A scourge of the sea, Head Hunter Fish are feared far and wide for their razor-sharp teeth and even nastier composure. Head Hunters love to eat Salmon, especially if he's big and fat. Head Hunter Fish are big, round, and tan in color. They can be found in all parts of the ocean and always hunt alone.
- **M.C. Hammerhead**—The oversized bully of Salmon's underwater neighborhood. M.C. Hammerhead is a giant Hammerhead Shark who knows no fear and who will eat any fish he comes across. A skilled hunter, M.C. always appears when you least expect him to. Gray in color like most sharks, M.C.'s size is equaled only by his obsession for Salmon.
- **Red Devils**—One of the smallest fish in the ocean and Salmon's easiest meal. Dark red in color, Red Devils are a small and stupid species of fish that wanders the ocean aimlessly. No one knows why or how they got such a horrible name but to Salmon they are simply known as "dinner."
- **Starfish**—Starfish are frequently found at the bottom of the ocean, alone and in pairs. When hungry, they float up to the surface in the hopes of finding food. A lazy species, they only eat what fish they run into. Unfortunately for Salmon, they would prefer to eat him over other types of fish.

In addition to these core objects, several items in *Fish Dish* can alter or enhance the abilities of the player. These objects are often referred to as *pickups*. In *Fish Dish*, the pickups include:

- **Bubbles**—Bubbles are transparent spheres that appear in various sizes and emanate from the ocean depths. Although they usually contain pockets of air, on rare occasions, some bubbles actually contain magical items that Salmon can use to his advantage in his quest to eat.
- **Pause Bubbles**—Pause Bubbles are special bubbles that contain the power to stop time for exactly three seconds. During this time, Salmon is the only fish in the ocean that can move. All other fish are stopped dead in their tracks regardless of their size or speed. Pause Bubbles make it very easy for Salmon to pig out and eat without worrying about being eaten. In some cases, they can also help get Salmon out of a tight spot. Pause Bubbles look like ordinary bubbles with miniature stoplights in them.
- **Shield Bubbles**—Shield Bubbles are special bubbles that contain the power to make Salmon invincible to bigger fish for exactly five seconds. During this time, Salmon cannot be eaten. Any fish that tries will simply bounce off Salmon, frustrated and hungry. Shield Bubbles are useful for protecting Salmon when he is small so that he can freely eat without having to worry

about becoming someone else's snack. Shield Bubbles look like normal bubbles except for a small shield icon inside of them.

Every game also has a few supporting objects that include everything from status indicators to title screens. In *Fish Dish*, these items are:

- **Title Screen**—A title screen that features a *Fish Dish* logo and a simple, game difficulty menu. The game's logo is located at the top of the screen and consists of a drawing of a fish skeleton while the game title is in thick, pink, cartoon-like lettering. The difficulty menu consists of three small buttons that resemble check boxes. The check boxes are labeled Easy, Medium, and Hard, respectively. The menu itself is positioned at the bottom of the screen.
- **Level 1 Background**—This is the background screen for all levels of the game. This screen consists of two parts: a light blue field and a dark brown field. The light blue field occupies the upper 85% of the screen and represents the ocean while the dark brown field that occupies the remaining 15% of the screen represents a seabed. The seabed itself consists of rocky and craggy mounds of earth rendered in various shades of brown.
- **Level 2 Background**—Same as in Level 1, except the game difficulty would increase.
- **Level 3 Background**—Same as in Levels 1 and 2, except the game difficulty would increase.

Game Functionality Overview

Fish Dish was originally conceived to be an online game. As such, it can be implemented using any existing programming technology and can run on virtually all platforms from Windows to the Macintosh. It is also perfectly feasible to use *Fish Dish*'s artwork for an offline game as well.

As designed, *Fish Dish* supports these features:

- Runs on any Windows 95, 98, NT, or 2000, Linux, or Macintosh computer capable of running Java or *Shockwave* (if implemented as an online-only game).
- Allows the player to oppose ten different creatures, each with a unique look and behavior.
- Allows the player to obtain temporary advantages with two pickup objects.

The Game Action Sequence

The game action sequence was also described in Chapter 11. In case you've forgotten, the purpose of this section of the design plan is to chart or "map out" the flow of action throughout *Fish Dish*. It is through this section that we gain an understanding of what game elements will be needed to be designed and how and where they will be used.

The Game Action Flowchart

Because *Fish Dish* is a simple arcade game, it has a relatively straightforward game action sequence. This sequence is shown in Figure 12-7.

Fish Dish Game Action Flow

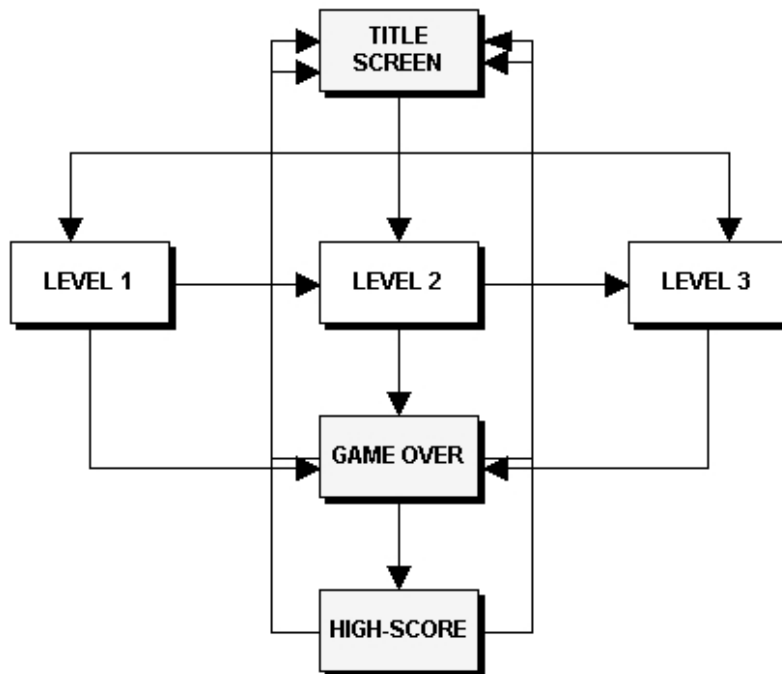


FIGURE 12-7: The *Fish Dish* Game Flow Sequence

The Screen Summary

Altogether, *Fish Dish* has five different game screens. However, only four of them are unique. The last screen, the Game Over screen, is essentially just a playing level that displays the text “Game Over” and a button that allows the player to play another game. This is a common feature of arcade-style games and is done to save space. As *Fish Dish* is an online game where space is at a premium, incorporating such efficiencies should be expected.

The direction of the arrows in Figure 12-7 indicates the flow of action between the game’s various screens. The game’s action breaks down as follows:

- From the title screen, the player selects a difficulty level. The choices available are Easy, Medium, or Hard.

- If the player chooses the Easy difficulty, they go to Level 1. Once Level 1 is beaten, the player progresses to Level 2. If they lose all of their available lives before completing the current level, they go to the Game Over screen.
- If the player chooses the Medium difficulty, they go to Level 2. Once Level 2 is beaten, the player progresses to Level 3. If they lose all of their available lives before completing the current level, they go to the Game Over screen.
- If the player chooses the Hard difficulty, they go to Level 3. As implemented, there is no ending or winning screen. *Fish Dish* simply continues on, filling the screen with more and more fish until it becomes virtually impossible to play. However, there is no reason why you cannot add one as an exercise.

In any case, here is a summary of the game screens in *Fish Dish*:

- Title Screen
- Level 1 (Easy difficulty)
- Level 2 (Medium difficulty)
- Level 3 (Hard difficulty)
- Game Over

Title Screen

This is the first screen that the player sees upon starting *Fish Dish*. Figure 12-8 shows what the actual title screen looks like.

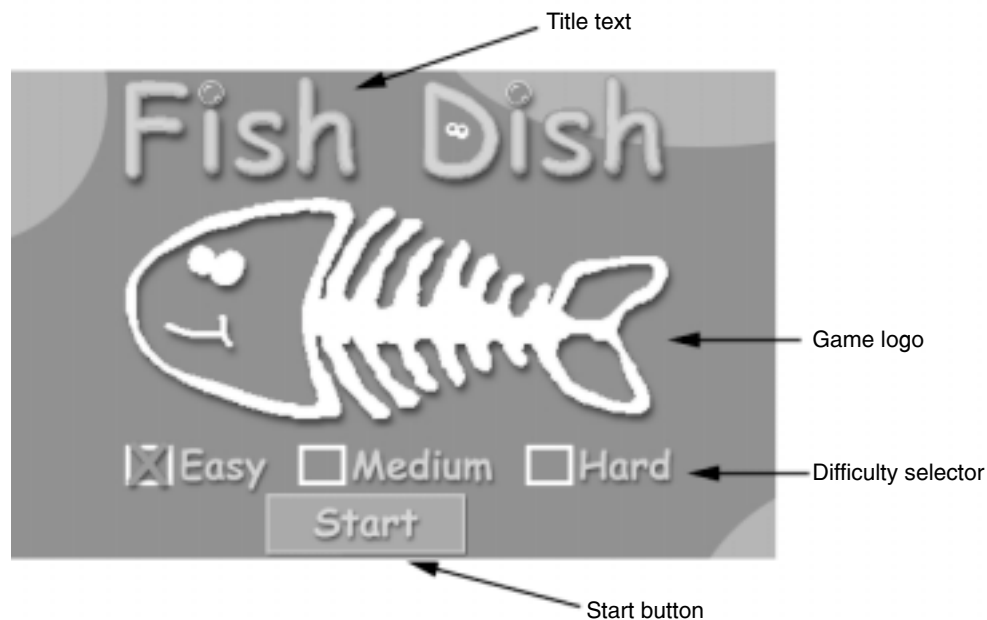


FIGURE 12-8: The *Fish Dish* Title Screen

In most games, this screen contains the program's title, logo, credits, and/or any related copyright information. It is here where the player can control the difficulty of the game as well as begin a new game. This is true regardless of whether the game is online, offline, maze/chase, or another type of arcade game.

In *Fish Dish*, the title screen contains four different elements. These are:

- The title text
- The game logo
- The difficulty selector
- The Start button

The title text is simply the game's name represented graphically. Most games, regardless of their type, use some sort of stylized title that ties back into the game's theme or plot. In *Fish Dish*, the title text is rendered using bright colors and includes elements from the game (such as bubbles and cartoon-style eyes). Not only do these elements give the title text character, but they help further reinforce their relationship to the game's plot.

The game logo is a graphical element that represents the theme or plot of the game in question. Many games feature logos that somehow tie into the game by using characters or items that are featured in the game. Other games use more abstract elements that are not in the actual game but still manage to tie into it nonetheless. *Fish Dish* falls into the latter category. The logo element consists of a cartoon-style fishbone. This was done to emphasize the general theme of the game, i.e., eating other fish and surviving. Several other concepts could have been executed in its place and would probably have the same effect, including putting a fish on a plate so that it would tie in with the game's name. Yet, to keep the example simple while effective in illustrating the point, only the fishbone motif was used here.

The difficulty selector is a menu that allows the player to determine the game's difficulty setting. The vast majority of arcade-style games usually place such menus on a separate screen; however, in order to maximize space and for the sake of illustration, it was placed on the title screen in *Fish Dish*.

As implemented, the difficulty selector consists of three check boxes in which the player merely has to click on one of them to choose a setting. Although other mechanisms could just as easily have been used, check boxes are probably the best choice. This is because they offer the player *affordances*, or subtle design characteristics that convey the correct use of an object. Therefore, people understand their purpose and know to click on them. As a rule, regardless of the type of game involved, always try to give the player as many affordances as possible.

The final item on the title screen is the Start button. This is a simple rectangle that the player clicks on to begin the actual game. Like the difficulty selector, the Start button contains such affordances as a drop shadow, a unique shape, and

position on the screen so that players have no problem understanding the button's purpose within the game.



NOTE: We will examine the construction of *Fish Dish*'s title screen later in this chapter.

Level Screens (Easy, Medium, and Hard)

These are the actual game screens where all of *Fish Dish*'s activity occurs. Figure 12-9 shows an example of what a typical *Fish Dish* game level looks like.

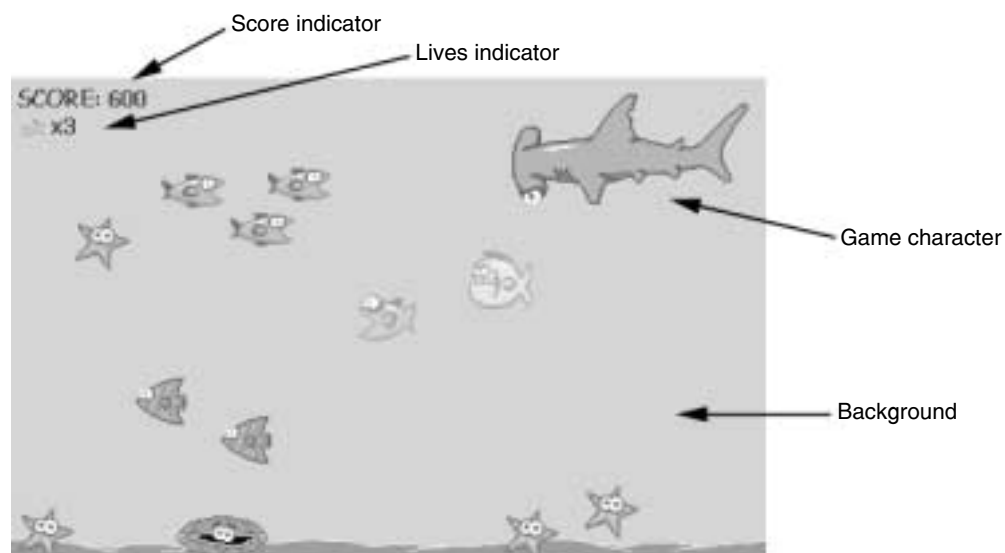


FIGURE 12-9: Example of a *Fish Dish* Game Level Screen

In most arcade games, the average level screen consists of a simple playfield along with any relevant scoring information, level information, and the current count of the player's available lives or chances.

In addition to hosting all of the game's action, each game level offers the player new game experiences. These experiences can range from introducing visual elements such as new characters or backgrounds to increasing the difficulty over the previous level.

Despite this possibility, every game level usually shares elements and objects in common. In the case of *Fish Dish*, there are four basic elements that never change, regardless of what challenges a new level brings. These items include the:

- Score indicator

- Lives indicator
- Game character
- Background

The score indicator does just what the name implies, it shows the player's current score during the course of the game. For a score indicator to be effective in its role, it must meet three essential requirements:

- It must be highly visible.
- It must be well placed on the screen.
- It must never be distracting or obtrusive to the player.

Although it might defy common sense, these three criteria are not always mutually inclusive. This is because some designers, in the attempt to make a game aesthetically pleasing, forget these requirements and sacrifice usability for the sake of appearance. This should never happen because if it does, the game may become unplayable.

A score indicator that is highly visible is, by definition, easier for the player to see. There are many ways to do this but all of these methods involve one or more of the object differentiation techniques described earlier. For example, rendering the score indicator in a large font with a light color and displaying it against a dark background can make it very easy to see when compared to using a small font and a dark color.

A score indicator should be well positioned on the game screen so that the player can easily spot it on a crowded game screen, especially during a frantic game session. Remember, for something to be seen, it must be well inside the player's field of vision. The best places for a score indicator are at the top left and bottom right of the screen. Table 12-14 provides a summary of the various places where a score indicator can appear and explains the rationale behind their placement.

TABLE 12-14: Summary of Score Indicator Positioning

<i>Score Indicator Position</i>	<i>Recommendation</i>	<i>Comments</i>
Top left	Highly recommended	Many players, particularly those in Europe and the United States read from left to right moving from the top down. Placing the score indicator at this part of the screen ensures that it is one of the first game elements the player sees.
Top right	Recommended	A good secondary position for the score indicator, especially when placement at the top left of the screen is impractical or impossible.

Score Indicator Position	Recommendation	Comments
Middle	Not recommended	Never place the score indicator (or any indicator for that matter) in the middle of the screen. Placing the score indicator here not only makes it difficult to see but can cause a potential conflict with the game's other objects.
Bottom left	Recommended but with reservations	An acceptable position but not preferred since this locates the score indicator outside of comfortable viewing.
Bottom right	Highly recommended	An excellent position for the score indicator because it is the last thing the player sees on the screen. This provides the player with an important visual clue as to the score's relative importance on the screen.

The third requirement of a good score indicator is to not be obtrusive or distracting to the player. In effect, this requirement directly relates to the previous two in that the score indicator's object differentiation methods and placement on the screen can become distracting if done improperly. As long as the first two requirements are addressed properly, the score indicator will never appear obtrusive to the player.

In *Fish Dish*, the score indicator meets all of these requirements. It uses a dark color that is rendered in a unique but readable font, and is positioned at the top left of the game screen. All of these factors ensure that the score remains highly visible to the player without being distracting or disruptive.

The purpose of the lives indicator is to show the player how many lives or chances they have at any point during a game. It is interesting to note that much of the same logic that applies to score indicators applies to lives indicators as well, with one significant addition: presentation. Traditionally, lives indicators in arcade-style games are iconic in nature. That is to say, they use small graphic elements, usually in the form of a smaller version of the main player-controlled character, to display the number of lives a player has left. In most cases, this device works well, as players are able to associate the icon with their on-screen character. However, in others it does not, especially when this icon runs the risk of confusing the player. For example, if a game uses a small spaceship for its lives indicator and then displays several small spaceships on the game screen, it stands to reason that some players will mistake their lives indicator for another game object. While this situation is usually not fatal, it is considered bad design. No status indicator (i.e., score indicator, lives indicator, etc.) should ever favor form over function. In other words, they should always work better than they look. One of

the best ways to do this is to simply use text to represent the actual number of player lives that remain.

It was for this very reason that *Fish Dish* takes a hybrid approach. It combines an icon of a small fish (Salmon) with a numeric counter as shown in Figure 12-9. This allows the lives indicator to look good while minimizing the potential for player confusion. In addition, placing the lives indicator in *Fish Dish* close to the score indicator tells the player that it is a status device and not an active game object they can manipulate or interact with.

Game characters are the graphical objects controlled by both the player and the computer. They usually consist of animated sprites that can interact with a variety of on-screen elements. As such, they are the focal point for all of the game's action. Due to their importance, all game characters must meet three essential requirements:

- They must be easily differentiated from each other.
- Their function or purpose within a game must be visually obvious.
- They must be consistent with the game's chosen design style across all levels.

The fact that game characters should be easily differentiated from each other should go without saying. Without a means for the player to distinguish between a game's different game objects, game play would be nonexistent. Game characters should always be designed using the various differentiation devices in mind. Whether you design your characters using differences in color, size, shape, or animation does not matter as long as the differences are sufficient for the player to tell objects apart.

Fish Dish uses several differentiation devices to ensure that its game characters are uniquely identifiable to the player. This gives the game more variety and visual appeal, and reduces the potential for player confusion.

One of the greatest challenges we face as designers is to convincingly render small grids of pixels as recognizable, real-world objects. For example, does the player recognize the graphic we made as a doorway? To make matters worse, due to a variety of resolution, size, and color restrictions that various platforms impose, it can become very difficult to accurately represent certain objects. For example, does the graphic we made actually look like a door? This in turn makes it very difficult for the average player to understand the context of how an object is used within a game. For example, does the player know that they can use the door to exit the current level? So lies our dilemma. It is our job to make it as easy as possible for the player to understand what is going on in a game at all times. This extends to how objects look and work. Therefore, you should maximize your efforts to always make sure that any game objects or characters you design for a game leave some clue as to their role in the game. There are a number of ways to do this. One of the easiest and most useful techniques is to simply carry over as

much of the real-world object's shape and color as possible within your particular size and color constraints. For example, if an apple is red in the real world, make sure that it is red in your game and shaped like an apple. Usually, such clues are enough to help the player make the connection as to what the object is.

Fish Dish addresses this issue by using this technique. All of its game characters are recognizable as fish so the player immediately has some understanding of what they are and how they fit into the context of the game.

It is essential that the game characters you design be consistent across the game's various screens and levels. In particular, this means the design style you chose should be reflected in all of the game's objects and not change. For example, if you design your game characters using a cartoon-like design style, make sure that all of the game's objects appear this way. The importance of consistency should not be underestimated. Simply put, players will associate the lack of consistency in your game and its characters with poor quality and bad design. Therefore, always strive to maintain whatever design convention you establish throughout your game.

Fish Dish follows this rule to a tee. Every game character that appears in it is drawn in the same design style using the same color shading techniques. This gives the game a coherent and professional look compared to a game that does not follow this requirement as closely.

Backgrounds are the actual playing area where all of the game's action occurs. Backgrounds can come in many styles and can range from the absurdly simple to the amazingly complex. Regardless of this, in order to be effective all arcade game backgrounds must follow these three rules:

- Backgrounds must provide sufficient contrast so that foreground elements stand out.
- Backgrounds must stylistically fit in with the foreground element used.
- Backgrounds must be designed with efficiency and flexibility in mind.

To be aesthetically pleasing, all game background screens must provide ample contrast. As already mentioned, contrast makes graphic elements stand out against a colored or complex background. Backgrounds that fail to provide sufficient contrast can potentially destroy the player's experience by making the game's artwork difficult to see and interact with. For example, a game with dark foreground elements should have a light background screen and vice versa. In addition, backgrounds that provide good contrast can improve the appearance and perceived quality of a game's artwork. The key to determining whether or not a given background screen has sufficient contrast can be done via the light/dark test first mentioned in Chapter 8.

Fish Dish uses a light background since most of the game's foreground elements are rendered in medium to dark colors. This produces a screen with the proper level of contrast.

Not only must a background screen have the proper contrast, but it must also be stylistically consistent with the various foreground game objects used. This means, for example, that if a game uses retro-style foreground elements, so should its backgrounds. Game backgrounds that do not stylistically fit in with a game's established look and feel will seem poorly executed to the average player. Many inexperienced designers make this mistake and, for example, use photo-realistic backgrounds with cartoon-style foreground objects. Do not make this mistake.

Figures 12-10 and 12-11 illustrate this important concept. In Figure 12-10, item A is rendered in a realistic design style while item B is rendered in a cartoon design style. It should be obvious that the two objects are stylistically incompatible with each other and should not be used in the same game. Meanwhile, in Figure 12-11, both items A and B are rendered using a realistic design style. This makes them stylistically compatible with each other and therefore they can and should be used together in a game. Incidentally, this logic applies to both background and foreground objects.

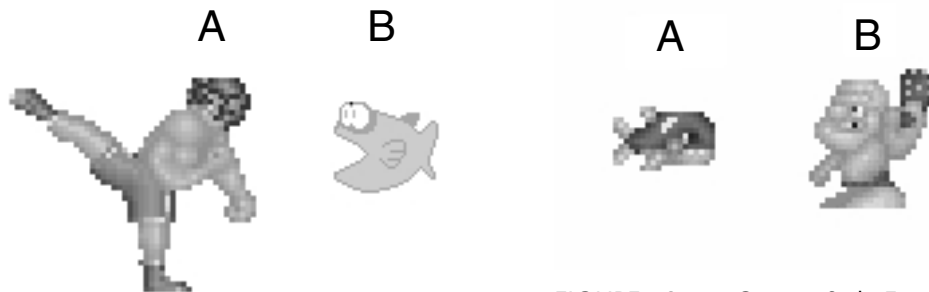


FIGURE 12-10: Incorrect Style Example

FIGURE 12-11: Correct Style Example

As expected, the backgrounds in *Fish Dish* closely match the style established by its game characters. This creates a unique synergy between the game's various elements to which the player can relate, which in turn helps them perceive *Fish Dish* as a high-quality production.

Although it is often overlooked by many designers, it is extremely important that your game backgrounds be designed with efficiency and flexibility in mind. This means that backgrounds should be designed to optimize well when compressed. Such efficiencies can be achieved by using large areas of solid colors, reducing the

number and arrangement of complex patterns, and using several pieces to represent a background rather than a single image. As you can imagine, practicing background screen efficiency can greatly reduce the file size of a game, especially if it is to be distributed or played online.

Flexible backgrounds make it possible to expand the type, number, and variety of levels in a game. Often, you can create more variations by breaking a background screen into smaller pieces and then arranging them in mosaic tile fashion. For example, by drawing parts of trees such as leaves, branches, and trunks one could create an entire forest in less space and with more variations than if each forest image was created individually. Incidentally, this technique has been used for years to good effect in video game systems.

The backgrounds in *Fish Dish* are designed with such efficiency and flexibility in mind. For one thing, they consist of large areas of solid colors. This makes them compress very well. Furthermore, the backgrounds are flexible because they include a number of elements that can be reused to construct additional background screens as needed.



NOTE: We are skipping the screen mockups that would ordinarily have been part of our design plan because they were presented as part of this section of the chapter.

Game Over Screen

The Game Over screen appears at the end of an arcade game, usually when the player exhausts all of their lives or chances. Figure 12-12 shows an example of what the Game Over screen in *Fish Dish* might look like:



FIGURE 12-12: The *Fish Dish* Game Over Screen

Some arcade games make the Game Over screen into a fancy affair and include additional artwork or animation that ties into the game's plot in order to tell the player that they lost.

The vast majority of arcade games do not need to go this far. In fact, one can get away with a Game Over screen that just displays a message that says "Game Over." However, in order to be truly effective, any Game Over screen should really meet two requirements:

- The Game Over message should be obvious and visible.
- The Game Over screen should provide a means of letting the player start a new game.

Nothing is more confusing and frustrating for a player than to not know when a game is actually over. Sure, we might assume that the lives indicator would cue them to this fact, but as designers, we cannot always make such assumptions. Therefore, we must always let the player know that the game is over. Knock them over the head with the message so that there can be no doubt in the player's mind. This can be done in several ways, such as using large text, certain color combinations, animation, certain placements, or even sound. As long as you catch their attention with the game's Game Over message, there will be no doubt in the player's mind that the game is indeed over.

Players hate it when a game ends and they have no obvious mechanism for starting a new game. Why not make it easy for them and provide a way for them to immediately play again? The easiest and most obvious way to do this is to simply include a Play Again or New Game button on your game's Game Over screen. This will save the player frustration and encourage them to keep playing your game.

Fish Dish meets both of these requirements. First, it includes a *Game Over dialog*. This is simply a graphical box that displays a "Game Over" message positioned at the center of the screen so there can be no doubt in the player's mind that the game has actually ended. In addition, it includes a large button that is labeled "New Game" so the player has an easy method for playing again.

The *Fish Dish* Graphics Specification

The next several pages contain the graphics specification for *Fish Dish*.

Game Creative Statement

Creatively speaking, *Fish Dish* artwork is drawn using a cartoon design style. In addition, bright colors were used to further highlight the light-hearted and comedic nature of the game.

Artwork Orientation

The artwork in *Fish Dish* is rendered using a *profile* orientation, i.e., the objects appear as if viewed from the side.

Arcade game graphics can be oriented in one of three ways: from the side (profile), from above (top-view), or in simulated 3D space (isometric). Figure 12-13 illustrates these orientations.

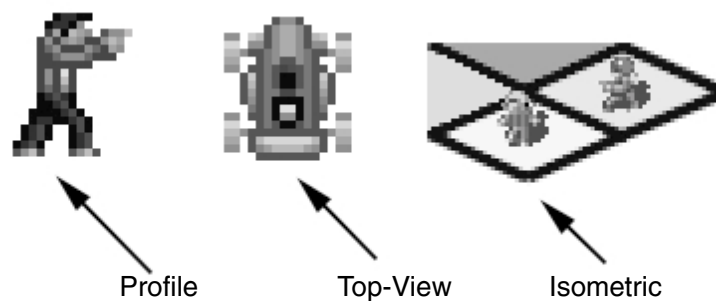


FIGURE 12-13: Object Orientation Examples

An arcade game should only use the profile or top-view orientations for their artwork. Although they can produce some very striking visual effects, isometric orientations tend to work best for non-arcade type games such as RPG (role-playing games) and simulations.

In addition, you should only use one orientation for your artwork at any one time, as no orientation scheme is complementary with another. For example, it would look very strange for an arcade shooter that uses a profile orientation to suddenly introduce objects that use a top-view orientation and vice versa. Therefore, determine which orientation to use and stick to it for the duration of the project.

Target Platform

As mentioned in several places in this chapter, *Fish Dish* is designed as an online game capable of running on virtually any 32-bit Windows, Linux, or Macintosh machine with the appropriate language support (i.e., Java or *Shockwave*). However, it is not limited to this and can be easily adapted to an offline format with minimal changes.

Estimated Object Count

In all, there are about 27 different objects in *Fish Dish*, including all variations of Salmon Fishdie and the various supporting elements (i.e., title and background screens).



NOTE: The graphics for *Fish Dish* were designed to be extensible enough to allow virtually anyone to extend the basic graphics set to include additional objects not described in the game object inventory.

Artwork Screen Resolution and Playfield Size

All of *Fish Dish*'s graphics were designed in a 640x480 screen resolution. Because the artwork was conceived with online play in mind, the active game area of *Fish Dish* was implemented using dimensions of 470 pixels wide by 300 pixels high. See our discussion of online game considerations earlier in this chapter.

Artwork Color Depth

All of *Fish Dish*'s graphics were designed for use in an 8-bit (256-color) display mode. This allows them to be used on a variety of different systems whether they support high color display modes or not. It also enables *Fish Dish* to take advantage of color cycling and other effects that palette-based display modes support should they be implemented by the game's programmer(s).

Refer to Chapter 2 for more information on the subject of color depth and Chapter 8 for more information on the topic of palettes and color cycling.

Artwork File Format(s)

All of *Fish Dish*'s artwork is stored using GIF 89a files. This makes its graphics assets compatible with the widest range of graphics programs and development tools, especially if its artwork is incorporated into an online game.

Artwork File Naming Scheme

The graphic assets in *Fish Dish* use a file naming scheme that is very similar to the example described in Chapter 4 of this book. Be aware that all of the game's assets were named so that they would be compatible with both Windows and Macintosh systems (i.e., maximum of 31 characters with file extensions).

Here is a breakdown of the suggested file naming scheme for *Fish Dish*:



NOTE: The object dimensions used for these filenames are for demonstration purposes only.

Backgrounds:

- bkg-level1-470x300.gif—Background screen for level 1
- bkg-level2-470x300.gif—Background screen for level 2
- bkg-level3-470x300.gif—Background screen for level 3

Title Screen:

- img-titlescreen-470x300.gif—Completed title screen

Sprites:

- spr-sal-470x300.gif—Sprite animations for Salmon
- spr-badfish-470x300.gif—Sprite animations for the other fish

Supporting and Miscellaneous Elements:

- mnu-easybutton_unsel-131x38.gif—The unchecked button for the easy difficulty setting
- mnu-easybutton_sel-131x38.gif—The checked button for the easy difficulty setting
- mnu-medbutton_unsel-131x38.gif—The unchecked button for the medium difficulty setting
- mnu-medbutton_sel-131x38.gif—The checked button for the medium difficulty setting
- mnu-hardbutton_unsel-131x38.gif—The unchecked button for the hard difficulty setting
- mnu-hardbutton_sel-131x38.gif—The checked button for the hard difficulty setting
- mnu-startbutton-123x38.gif—The game Start button
- img-fishdish_title-346x72.gif—The game title text
- img-fishfish_logo-346x148.gif—The game logo
- img-title-back-470x300.gif—The title screen background pattern
- img-gameover-160x118.gif—The Game Over dialog box

Artwork Color Palette

The *Fish Dish* color palette uses the Java/Netscape system palette. This color palette was then modified to include the 20 Windows reserved colors described in Chapter 8. Doing this allows the artwork in *Fish Dish* to be used as both an online and offline arcade game.

The palette is supplied in the Microsoft .PAL format. The file is called `fishdish.pal`.

The transparent color for this palette is index (palette entry) 0 which is also RGB: 0,0,0 or black.

The actual RGB values for the color palette can be found on the book's accompanying CD-ROM. Please see Appendix B for additional details.

Artwork Gamma Level

Fish Dish uses a 2.2 gamma level since its artwork was designed on a Windows system. However, because the Java/Netscape color palette contains color definitions that are the same across Windows and the Macintosh, gamma-related issues do not have a significant impact.

Artwork Object Dimensions

Fish Dish uses several different grid sizes to contain its various objects and elements. All of the main game characters use sprite grids that range in size from 16x16 to 40x40. This was done to support the “growing” effect of the game's main character. In addition, several of the larger objects use sprite grids up to 148x72 pixels in size.

These sizes were selected because they maximize the amount of object detail that can be displayed while remaining relatively efficient in terms of their screen performance and file size.

In any case, here is a breakdown of the sprite grid sizes of the different objects in *Fish Dish*:

Salmon Fishdie:

- 16x16, 20x20, 24x24, 28x28, 32x32, 36x36, 40x40

Salmon Angel:

- 16x16

Bad Tuna:

- 36x36

Blue Angels:

- 32x32

Gold Diggers:

- 20x20

Gray Sharks:

- 28x28

Green Meanies:

- 24x24

Happy Clams:

- 66x30

Head Hunter Fish:

- 40x40

M.C. Hammerhead:

- 148x72

Red Devils:

- 16x16

Starfish:

- 36x36


Bubbles and Pickups (All):

- 24x24

Frames per Object

Due to its simplicity, most of *Fish Dish*'s objects are animated using minor animation techniques like the swing and open/close primitives. This results in animations that use a relatively small number of total frames.

Here is a breakdown of the different game objects:




NOTE: Only objects that require animation are included in this list.

- **Salmon Fishdie:** Moving—5 frames (x7) sizes = 35 frames total
- **Salmon Angel:** Flying—3 frames (x1) size = 3 frames total
- **Bad Tuna:** Moving—5 frames (x1) size = 5 frames total
- **Blue Angels:** Moving—5 frames (x1) size = 2 frames total
- **Gold Diggers:** Moving—5 frames (x1) size = 2 frames total
- **Gray Sharks:** Moving—5 frames (x1) size = 2 frames total
- **Green Meanies:** Moving—5 frames (x1) size = 2 frames total
- **Happy Clams:** Moving—1 frame (x1) size = 1 frames total
- **Head Hunter Fish:** Moving—5 frames (x1) size = 5 frames total
- **M.C. Hammerhead:** Moving—1 frame (x1) size = 1 frame total
- **Red Devils:** Moving—1 frame (x1) size = 1 frame total

- **Starfish:** Moving—1 frame (x1) size = 1 frame total

Object Actions and Facings

This section lists the actions and directional facings for the key game objects.

 **NOTE:** Only objects that require animation are included in this list.

- **Salmon Fishdie:** Moving—2 directions (left and right)
- **Salmon Angel:** Flying—1 direction (up)
- **Bad Tuna:** Moving—2 directions (left and right)
- **Blue Angels:** Moving—2 directions (left and right)
- **Gold Diggers:** Moving—2 directions (left and right)
- **Gray Sharks:** Moving—2 directions (left and right)
- **Green Meanies:** Moving—2 directions (left and right)
- **Happy Clams:** Moving—2 directions (left and right)
- **Head Hunter Fish:** Moving—2 directions (left and right)
- **M.C. Hammerhead:** Moving—2 directions (left and right)
- **Red Devils:** Moving—2 directions (left and right)
- **Starfish:** Moving—2 directions (left and right)

Game Text Font(s)

Fish Dish does not display a significant amount of textual information on its screens. However, it does use one font:

- **MS Comic Sans**—This font appears throughout *Fish Dish*. It is used extensively on the title screen where it appears in the game title text, difficulty selector, and Start button. It is also used in both the score and lives indicators. Finally, it appears in the Game Over dialog. This font was chosen because it has a distinctive appearance that is reminiscent of the type that is used in cartoons and comics. It was thought that this characteristic would blend in perfectly with the rest of *Fish Dish* given its humorous plot and engaging cartoon-like elements.

Phase III: The *Fish Dish* Design Execution

This part of the chapter focuses on how the various objects in *Fish Dish* were designed and created.

Artwork Templates

The fastest and most efficient way to create artwork for an arcade game is to *templatize*, or systematically develop a process for constructing each game element. Doing this allows you to break your artwork into a series of simple and consistently repeatable steps, which reduces the possibility for making mistakes while helping you maintain a good level of control over image quality.

Although not every game project lends itself to such a system, most do. In *Fish Dish*, two sets of templates were used, a design template which guided the creation of the game artwork and an animation template which guided the development of the related object animation.

Table 12-15 shows the basic components of the *Fish Dish* design template.

TABLE 12-15: The General *Fish Dish* Design Template

Step #	Procedure	Description and Purpose
1	Create the general object shape	The basic shape of the object is created, usually from a paper sketch or a simple pixel-by-pixel drawing.
2	Establish object coloring	The object begins to be filled in with a base color and the basic boundaries of the different parts of the object begin to become visible.
3	Add expressive and other functional elements	This is where the various expressive elements such as the eyes, eyebrows, and teeth are created as well as where functional design elements such as the fins are added.
4	Shade and complete	The object is completely colored and shaded and the final detailing is added.

Using this scheme, the average game object in *Fish Dish* can be completed in just four steps. A few objects even require just three.

Once the game object is completed, it must be animated. To simplify this process, we have devised an animation template for use in *Fish Dish*. Table 12-16 shows the basic components of the *Fish Dish* animation template.

TABLE 12-16: The General *Fish Dish* Animation Template

Step #	Procedure	Description and Purpose
1	Draw starting movement	The starting key-frame of the animation (usually a swing or open/close primitive) is created.
2	Draw the ending movement	The ending key-frame of the animation (usually a swing or open/close primitive) is created. All animated objects in <i>Fish Dish</i> use two to three frames of animation and no in-betweens in order to keep the animation sequences very simple.

Step #	Procedure	Description and Purpose
3	Test the animation	Once the animation sequence is completed, it is tested. If it looks good, we go on to step 4, otherwise, we revisit steps 1 and 2 until we get it right.
4	Horizontally flip the object frames	Where applicable, completed animations are flipped horizontally so we can have objects that appear to move from both left to right and right to left. Doing this is a real time saver as it allows us to only draw one set of images for both directions.

Step-by-Step Game Object Design

At this point, we are now ready to examine the design and construction of each game object in detail.

From here on in, each game object will be visually broken down into its basic components to better illustrate how the final object was designed and created. For the purposes of clarity, every character is provided with detailed figures that show the progression of the object's design and that maps directly to a unique step in the design template. There is not a one-to-one mapping of the animation sequences to the animation template shown in Table 12-16. Instead, the exact frame-by-frame sequence is provided. However, you are encouraged to follow the process established in Table 12-16.

Because this book is printed in black and white as opposed to color, a special visual color map diagram is also provided so that you can see where the colors for a given object are supposed to be applied.



NOTE: The master files for all of *Fish Dish*'s artwork can be found on the book's CD-ROM in the `EXAMPLES/FISHDISH` directory.

Before proceeding, you should familiarize yourself with the basic visual elements of the characters in *Fish Dish* as most of the game's objects and characters share one or more of these elements in common. They are identified in Figure 12-14.

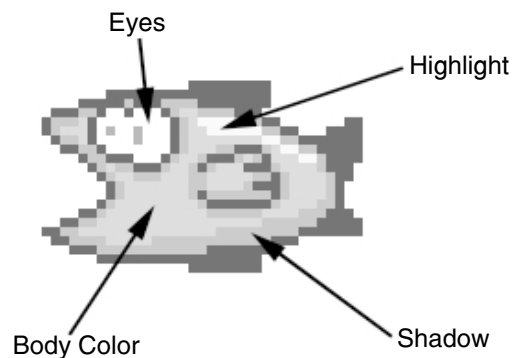


FIGURE 12-14: General Object Element Map

Salmon Fishdie Object Creation

TABLE 12-17: The Salmon Fishdie Color Scheme

Generic Color Name	Palette Entry #	RGB Value	Usage
Magenta	62	204 51 153	The primary object bounding and detailing color
Medium pink	19	255 102 104	The shadow color
Hot pink	12	255 153 155	The main body color
Light gray	43	204 204 204	The eyeball shading color
Pure white	255	255 255 255	The eyeball and body highlighting color
Medium blue	168	51 51 255	The eye iris color. This color is standard for all game characters that appear in <i>Fish Dish</i> .

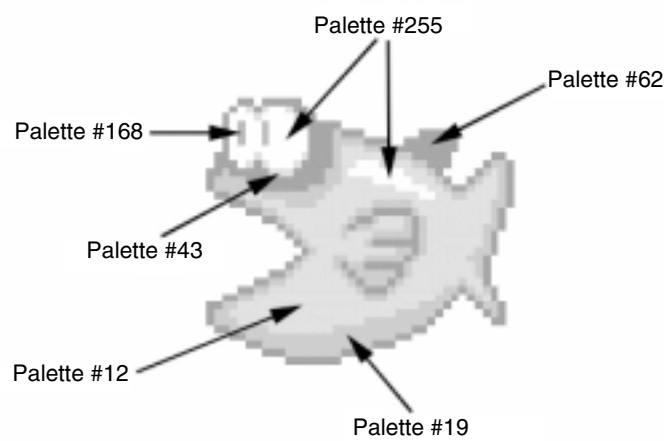


FIGURE 12-15: The Salmon Fishdie Visual Color Map

The Salmon Fishdie Design Templates

Figures 12-16 through 12-22 illustrate the progressive development of the various Salmon Fishdie objects.

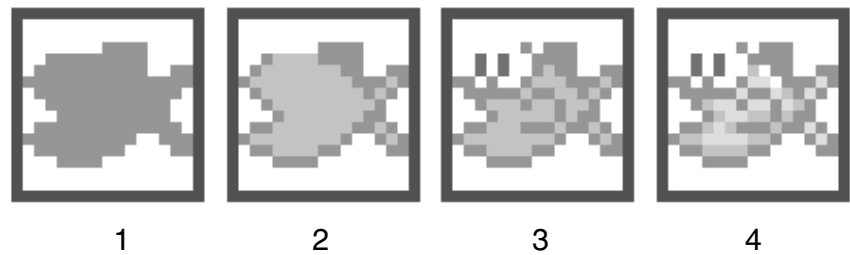


FIGURE 12-16: The 16x16 Salmon Fishdie Design Template



FIGURE 12-17: The 20x20 Salmon Fishdie Design Template



FIGURE 12-18: The 24x24 Salmon Fishdie Design Template

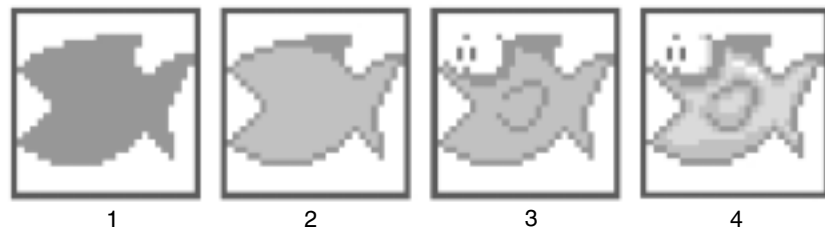


FIGURE 12-19: The 28x28 Salmon Fishdie Design Template

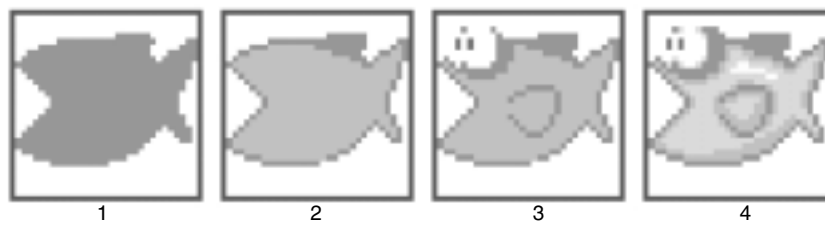


FIGURE 12-20: The 32x32 Salmon Fishdie Design Template



FIGURE 12-21: The 36x36 Salmon Fishdie Design Template



FIGURE 12-22: The 40x40 Salmon Fishdie Design Template

The Salmon Fishdie Animation Templates

Figures 12-23 through 12-29 show the animation sequences of the various Salmon Fishdie objects. There are five frames in each sequence and each sequence uses cycling and loops. These animations are also excellent examples of the open/close primitives described in Chapter 9.

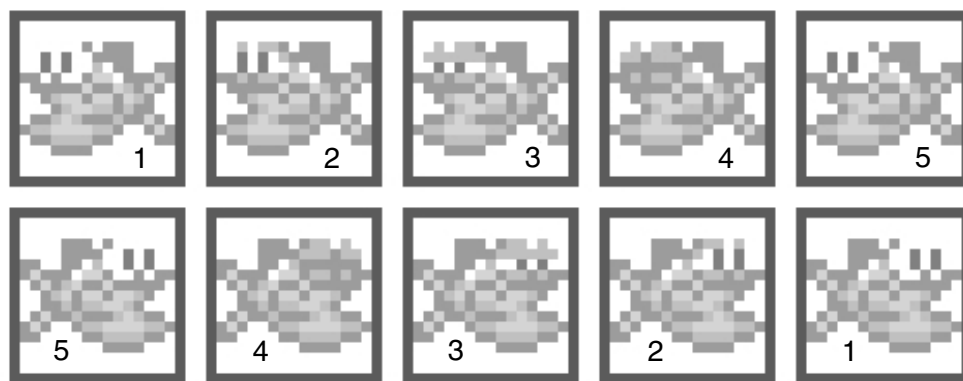


FIGURE 12-23: The 16x16 Salmon Fishdie Animation Template



FIGURE 12-24: The 20x20 Salmon Fishdie Animation Template



FIGURE 12-25: The 24x24 Salmon Fishdie Animation Template



FIGURE 12-26: The 28x28 Salmon Fishdie Animation Template



FIGURE 12-27: The 32x32 Salmon Fishdie Animation Template



FIGURE 12-28: The 36x36 Salmon Fishdie Animation Template



FIGURE 12-29: The 40x40 Salmon Fishdie Animation Template

Red Devil Object Creation

TABLE 13-18: The Red Devil Color Scheme

Generic Color Name	Palette Entry #	RGB Value	Usage
Medium red	107	153 0 0	The primary object bounding and detailing color
Light red	64	204 51 51	The main body color
Pure white	255	255 255 255	The eyeball color
Medium blue	168	51 51 255	The eye iris color. This color is standard for all game characters that appear in <i>Fish Dish</i> .

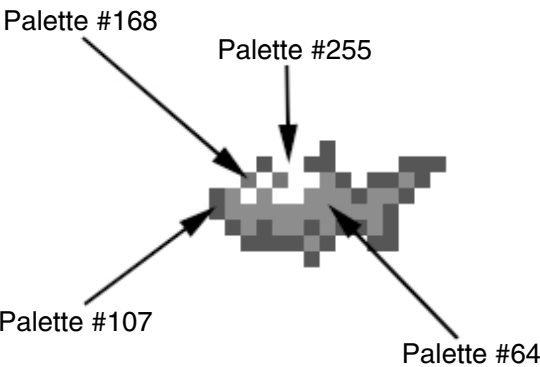


FIGURE 12-30: The Red Devil Visual Color Map

The Red Devil Design Template

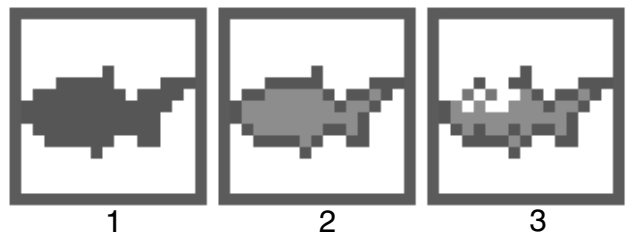


FIGURE 12-31: The Red Devil Design Template

Gold Digger Object Creation

TABLE 12-19: The Gold Digger Color Scheme

Generic Color Name	Palette Entry #	RGB Value	Usage
Dark brown	137	102 51 0	The primary object bounding and detailing color
Medium brown	95	153 102 0	The shadow color
Light brown	53	204 153 0	The secondary shading and detailing color
Very light brown	10	255 153	The main body color
Pure white	255	255 255 255	The eyeball and body highlighting color
Medium blue	168	51 51 255	The eye iris color. This color is standard for all game characters that appear in <i>Fish Dish</i> .

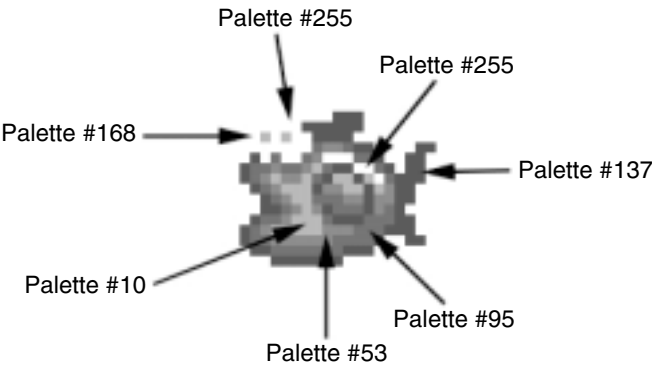


FIGURE 12-32: The Gold Digger Visual Color Map

The Gold Digger Design Template

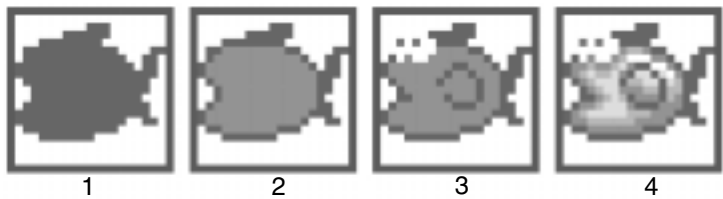


FIGURE 12-33: The Gold Digger Design Template

The Gold Digger Animation Template

Figure 12-34 contains the complete animation sequences for the Gold Digger character. Notice that this animation consists of five frames, makes use of cycling and loops, and uses the open/close animation primitive.

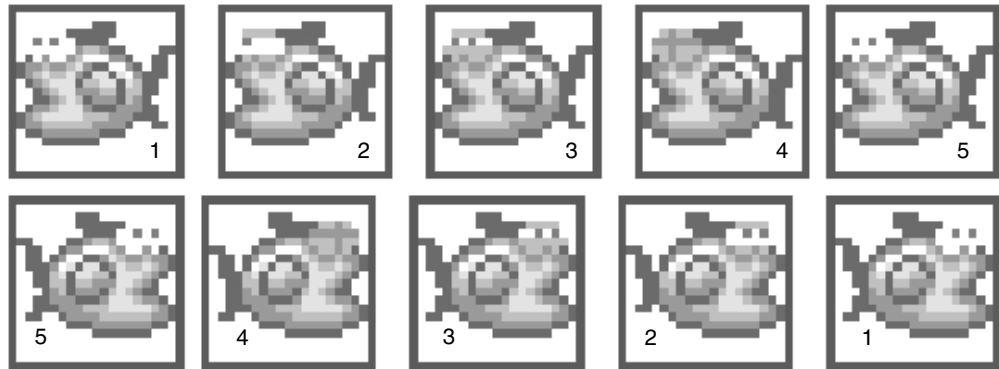


FIGURE 12-34: The Gold Digger Animation Template

Green Meanie Object Creation

TABLE 12-20: The Green Meanie Color Scheme

Generic Color Name	Palette Entry #	RGB Value	Usage
Dark green	203	0 102 0	The primary object bounding and detailing color
Medium green	197	0 153 0	The shadow color
Light green	191	0 204 0	The main body color
Very light green	113	102 255 0	The body highlighting color
Pure white	255	255 255 255	The eyeball color
Medium blue	168	51 51 255	The eye iris color. This color is standard for all game characters that appear in <i>Fish Dish</i> .

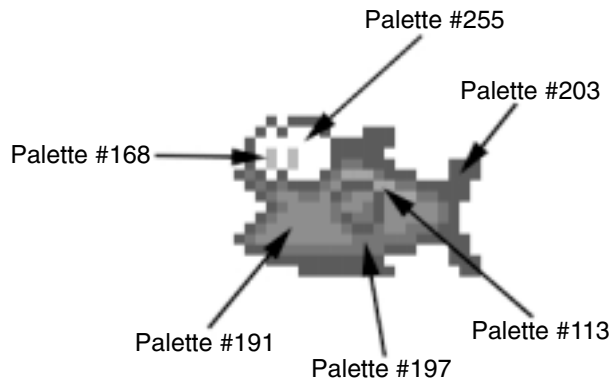


FIGURE 12-35: The Green Meanie Visual Color Map

The Green Meanie Design Template

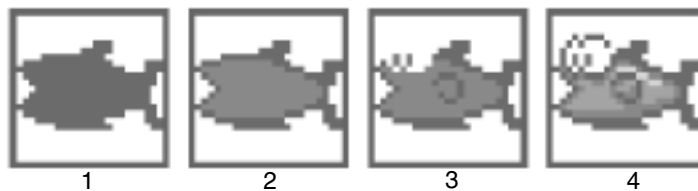


FIGURE 12-36: The Green Meanie Design Template

The Green Meanie Animation Template

Figure 12-37 contains the complete animation sequences for the Green Meanie character. Notice that this animation consists of five frames, makes use of cycling and loops, and uses the open/close animation primitive.

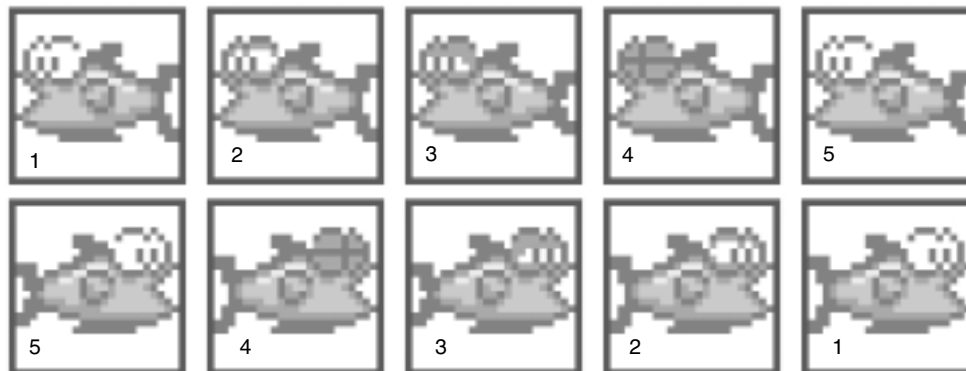


FIGURE 12-37: The Green Meanie Animation Template

Gray Shark Object Creation

TABLE 12-21: The Gray Shark Color Scheme

Generic Color Name	Palette Entry #	RGB Value	Usage
Dark gray	172	51 51 51	The primary object bounding and detailing color
Medium gray	129	102 102 102	The shadow color
Light gray	86	153 153 153	The main body color
Very light gray	43	204 204 204	The highlighting color
Pure white	255	255 255 255	The eyeball color
Medium blue	168	51 51 255	The eye iris color. This color is standard for all game characters that appear in <i>Fish Dish</i> .

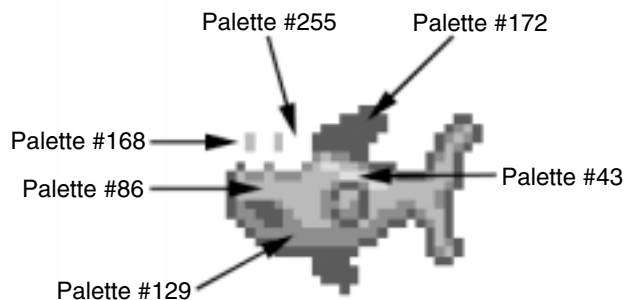


FIGURE 12-38: The Gray Shark Visual Color Map

The Gray Shark Design Template



FIGURE 12-39: The Gray Shark Design Template

The Gray Shark Animation Template

Figure 12-40 contains the complete animation sequences for the Gray Shark character. Notice that this animation consists of five frames, makes use of cycling and loops, and uses the open/close animation primitive.

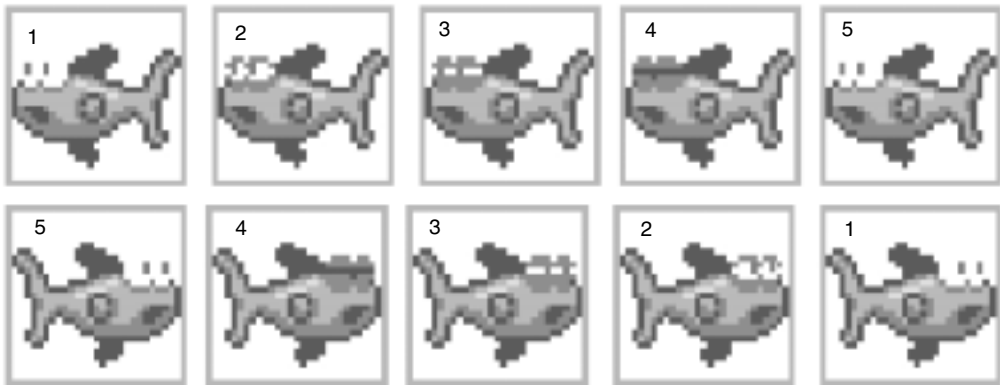


FIGURE 12-40: The Gray Shark Animation Template

Blue Angel Object Creation

TABLE 12-22: The Blue Angel Color Scheme

Generic Color Name	Palette Entry #	RGB Value	Usage
Dark blue	206	0 51 153	The primary object bounding and detailing color
Medium blue	198	0 102 255	The base body color
Light blue	192	0 153 255	The foundation body color
Sky blue	150	51 204 255	The body highlighting color
Pure white	255	255 255 255	The eyeball color
Medium blue	168	51 51 255	The eye iris color. This color is standard for all game characters that appear in <i>Fish Dish</i> .

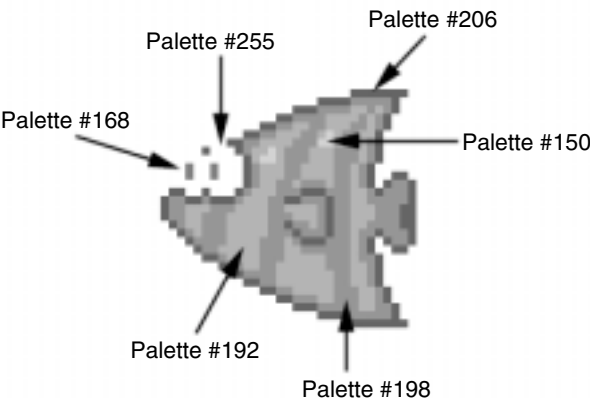


FIGURE 12-41: The Blue Angel Visual Color Map

The Blue Angel Design Template

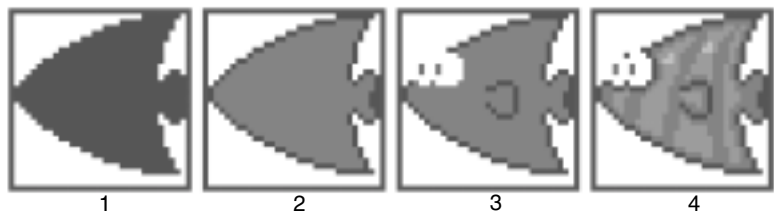


FIGURE 12-42: The Blue Angel Design Template

The Blue Angel Animation Template

Figure 12-43 contains the complete animation sequences for the Blue Angel character. Notice that this animation consists of five frames, makes use of cycling and loops, and uses the open/close animation primitive.

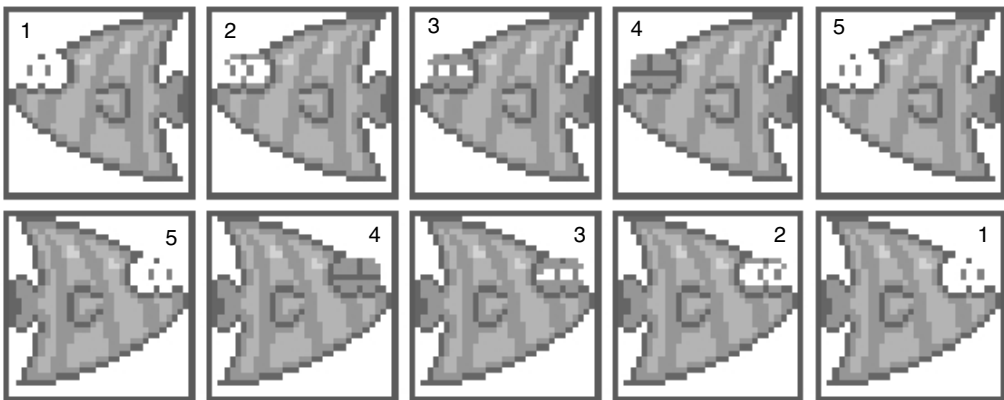


FIGURE 12-43: The Blue Angel Animation Template

Bad Tuna Object Creation

TABLE 12-23: The Bad Tuna Color Scheme

Generic Color Name	Palette Entry #	RGB Value	Usage
Dark purple	140	102 0 153	The primary object bounding and detailing color
Light purple	90	153 102 255	The shadow color
Lavender	48	204 153 255	The main body color
Pure white	255	255 255 255	The eyeball and body highlighting color

Generic Color Name	Palette Entry #	RGB Value	Usage
Medium blue	168	51 51 255	The eye iris color. This color is standard for all game characters that appear in <i>Fish Dish</i> .

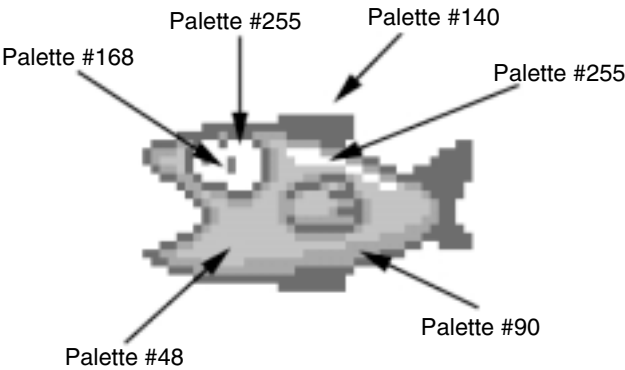


FIGURE 12-44: The Bad Tuna Visual Color Map

The Bad Tuna Design Template

Figure 12-45 shows the progressive development of the Bad Tuna object.

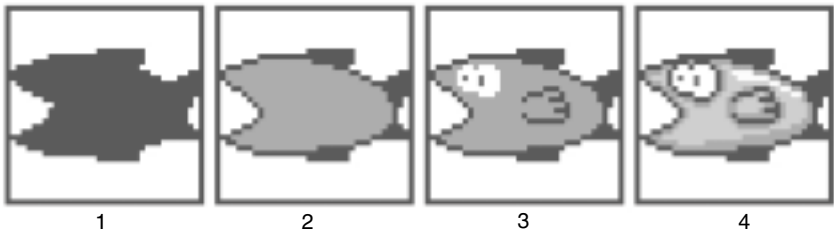


FIGURE 12-45: The Bad Tuna Design Template

The Bad Tuna Animation Template

Figure 12-46 contains the complete animation sequences for the Bad Tuna character. Notice that this animation consists of five frames, makes use of cycling and loops, and uses the open/close animation primitive.

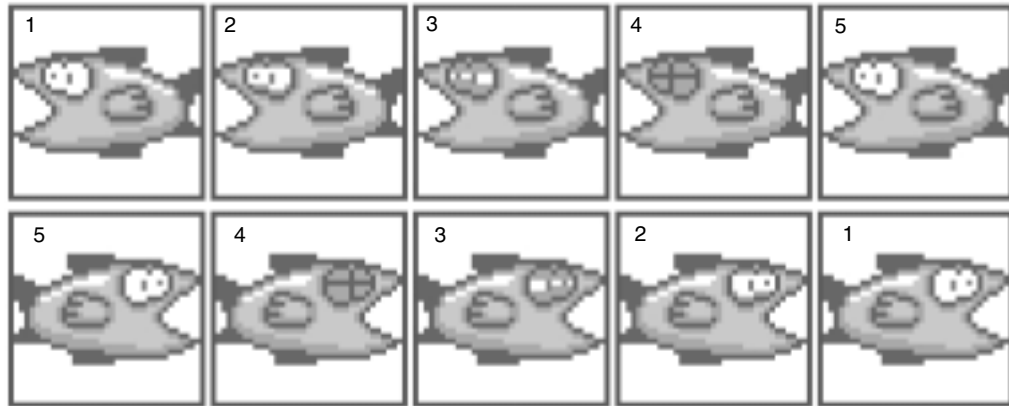


FIGURE 12-46: The Bad Tuna Animation Template

Head Hunter Object Creation

TABLE 12-24: The Head Hunter Color Scheme

Generic Color Name	Palette Entry #	RGB Value	Usage
Light brown	59	204 102 0	The primary object bounding and detailing color
Dark tan	53	204 153 0	The shadow color
Sand	10	255 204 51	The main body color
Medium gray	86	153 153 153	The highlighting color for the teeth
Pure white	255	255 255 255	The eyeball and body highlighting color
Medium blue	168	51 51 255	The eye iris color. This color is standard for all game characters that appear in <i>Fish Dish</i> .

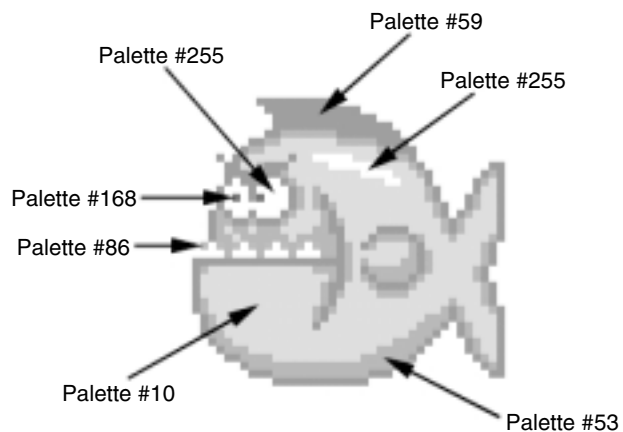


FIGURE 12-47: The Head Hunter Visual Color Map

The Head Hunter Design Template

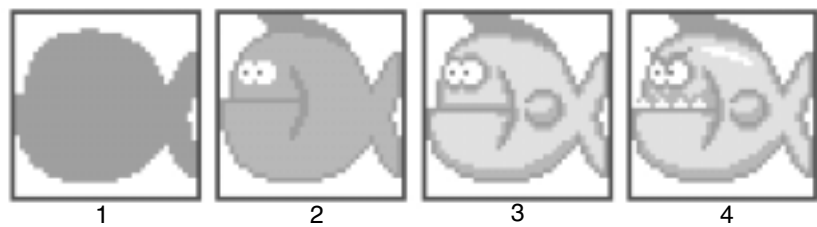


FIGURE 12-48: The Head Hunter Design Template

The Head Hunter Animation Template

Figure 12-49 contains the complete animation sequences for the Head Hunter character. Notice that this animation consists of five frames, makes use of cycling and loops, and uses the open/close animation primitive.

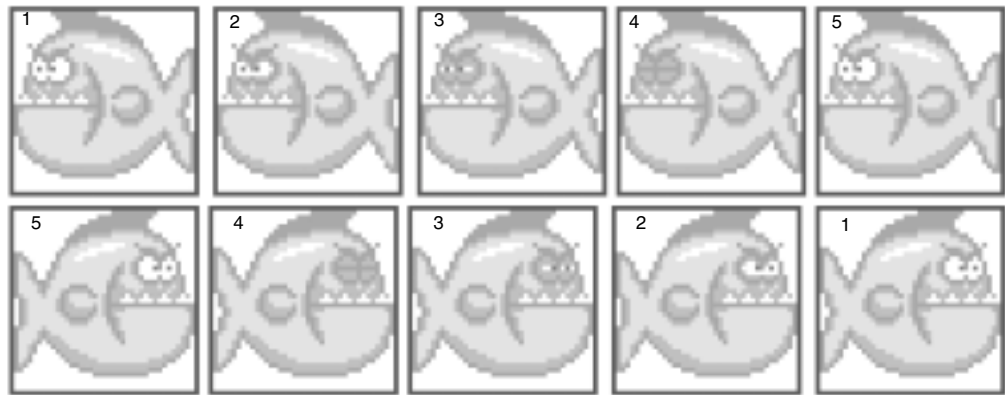


FIGURE 12-49: The Head Hunter Animation Template

Starfish Object Creation

TABLE 12-25: The Starfish Color Scheme

Generic Color Name	Palette Entry #	RGB Value	Usage
Dark red	107	153 0 0	The primary object bounding and detailing color
Medium red	64	204 51 51	The shadow color
Light red	21	255 102 102	The main body color
Very light red	14	255 153 153	The highlighting color
Pure white	255	255 255 255	The eyeball color

Generic Color Name	Palette Entry #	RGB Value	Usage
Light gray	43	204 204 204	The eyeball shadow color
Medium blue	168	51 51 255	The eye iris color. This color is standard for all game characters that appear in <i>Fish Dish</i> .

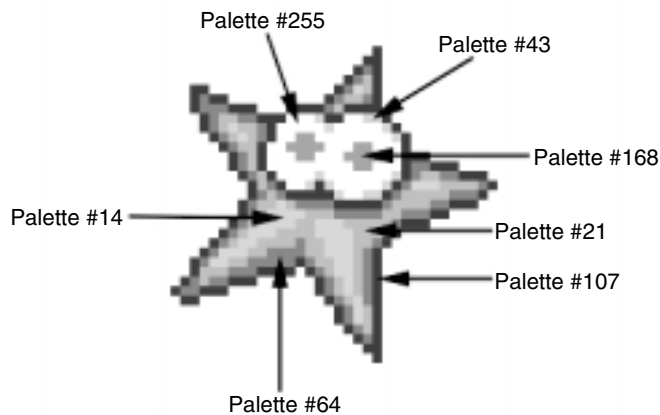


FIGURE 12-50: The Starfish Visual Color Map

The Starfish Design Template

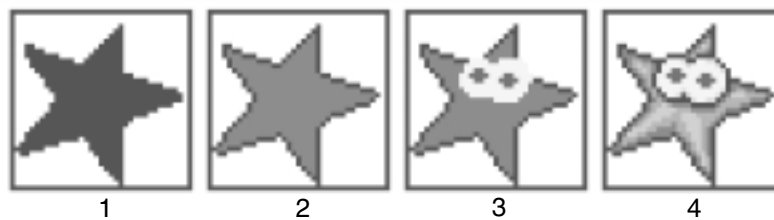


FIGURE 12-51: The Starfish Design Template

Happy Clam Object Creation

TABLE 12-26: The Happy Clam Color Scheme

Generic Color Name	Palette Entry #	RGB Value	Usage
Dark brown	137	102 51 0	The primary object bounding and detailing color
Medium brown	59	204 102 0	The shadow color
Sand	17	255 153 0	The main body color

Generic Color Name	Palette Entry #	RGB Value	Usage
Light gray	43	204 204 204	The eyeball shading color
Pure white	255	255 255 255	The eyeball color
Medium blue	168	51 51 255	The eye iris color. This color is standard for all game characters that appear in <i>Fish Dish</i> .
Black	0	0 0 0	The secondary body color

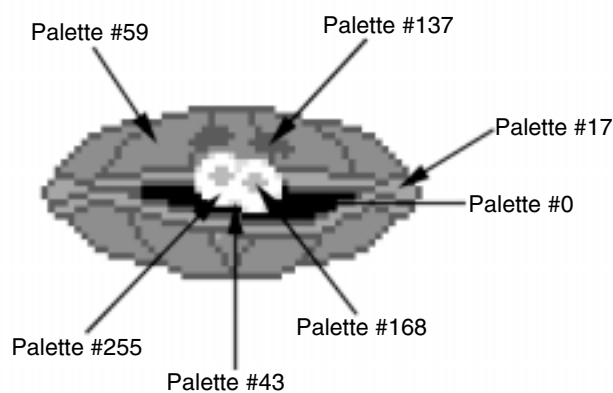


FIGURE 12-52: The Happy Clam Visual Color Map

The Happy Clam Design Template



FIGURE 12-53: The Happy Clam Design Template

M.C. Hammerhead Object Creation

TABLE 12-27: The M.C. Hammerhead Color Scheme

Generic Color Name	Palette Entry #	RGB Value	Usage
Dark gray	172	51 51 51	The primary object bounding and detailing color
Medium gray	129	102 102 102	The shadow color
Light gray	86	153 153 153	The main body color

Generic Color Name	Palette Entry #	RGB Value	Usage
Pure white	255	255 255 255	The eyeball color
Medium blue	168	51 51 255	The eye iris color. This color is standard for all game characters that appear in <i>Fish Dish</i> .

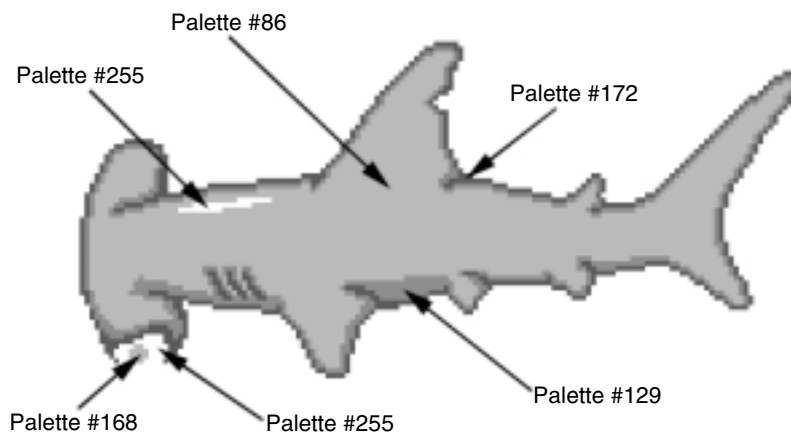


FIGURE 12-54: The M.C. Hammerhead Visual Color Map

The M.C. Hammerhead Design Template



FIGURE 12-55: The M.C. Hammerhead Design Template

Salmon Fishdie Angel Object Creation

TABLE 12-28: The Salmon Fishdie Angel Color Scheme

Generic Color Name	Palette Entry #	RGB Value	Usage
Magenta	62	204 51 153	The primary object bounding and detailing color
Medium pink	19	255 102 104	The shadow color
Hot pink	12	255 153 155	The main body color
Yellow	251	255 255 0	The halo color

Generic Color Name	Palette Entry #	RGB Value	Usage
Pure white	255	255 255 255	The eyeball and wing color
Medium blue	168	51 51 255	The eye iris color. This color is standard for all game characters that appear in <i>Fish Dish</i> .

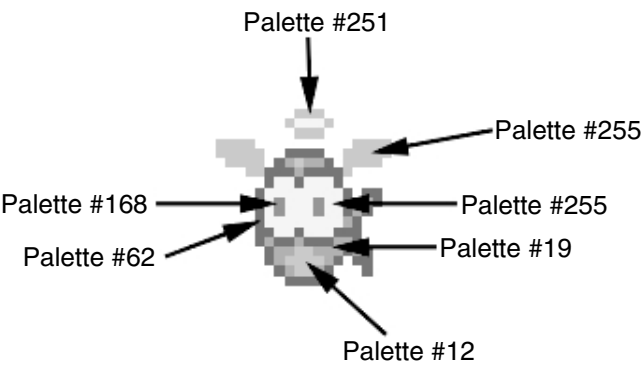


FIGURE 12-56: The Salmon Fishdie Angel Visual Color Map

The Salmon Fishdie Angel Design Template



FIGURE 12-57: The Salmon Fishdie Angel Design Template

The Salmon Fishdie Angel Animation Template

Figure 12-58 contains the complete animation sequences for the Salmon Fishdie Angel character. Notice that this animation consists of three frames, makes use of cycling and loops, and uses the swing animation primitive.



FIGURE 12-58: The Salmon Fishdie Angel Animation Template

Pause Bubble Object Creation

TABLE 12-29: The Pause Bubble Color Scheme

Generic Color Name	Palette Entry #	RGB Value	Usage
Pure white	255	255 255 255	The bubble color
Dark gray	172	51 51 51	The stoplight color
Medium red	35	255 0 0	The first light color
Medium green	161	51 153 0	The second light color
Medium yellow	17	253 153 0	The third light color
Black	0	0 0 0	The bubble background color

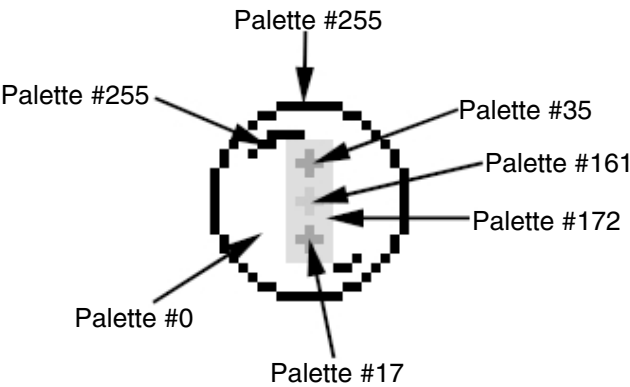


FIGURE 12-59: The Pause Bubble Visual Color Map

NOTE: The bubble outlines are white and the interiors are black as indicated by the palette numbers. However, they are shown in reverse in Figures 12-59 and 12-61.

The Pause Bubble Design Template

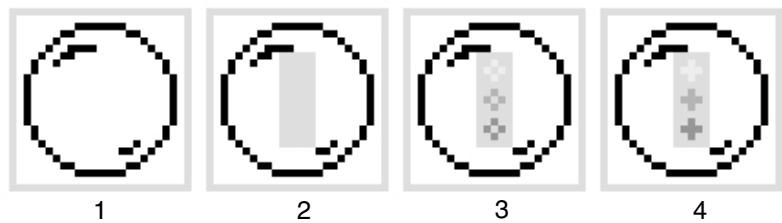


FIGURE 12-60: The Pause Bubble Design Template

Shield Bubble Object Creation

TABLE 12-30: The Shield Bubble Color Scheme

Generic Color Name	Palette Entry #	RGB Value	Usage
Pure white	255	255 255 255	The bubble color
Light gray	43	51 51 51	The outer shield color
Medium blue	207	255 0 0	The first inner shield color
Light blue	198	51 153 0	The second inner shield color
Black	0	0 0 0	The bubble background color

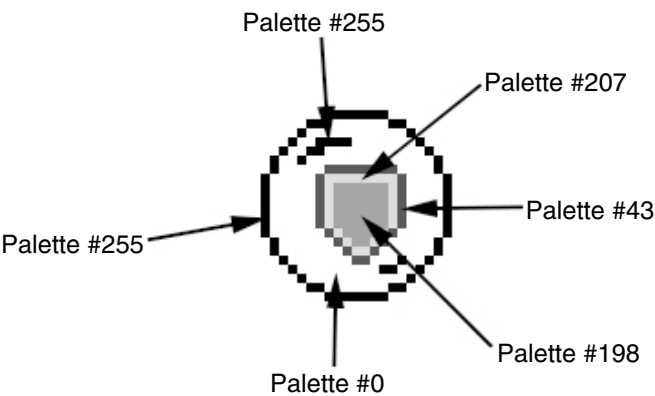


FIGURE 12-61: The Shield Bubble Visual Color Map

The Shield Bubble Design Template



FIGURE 12-62: The Shield Bubble Design Template

Background Screen Creation

TABLE 12-31: The Background Screen Color Scheme

Generic Color Name	Palette Entry #	RGB Value	Usage
Medium brown	95	153 102 0	The main surface color
Dark brown	137	102 51 0	The shadow color
Beige	52	204 153 51	The highlighting color
Sky blue	114	102 204 255	The water color

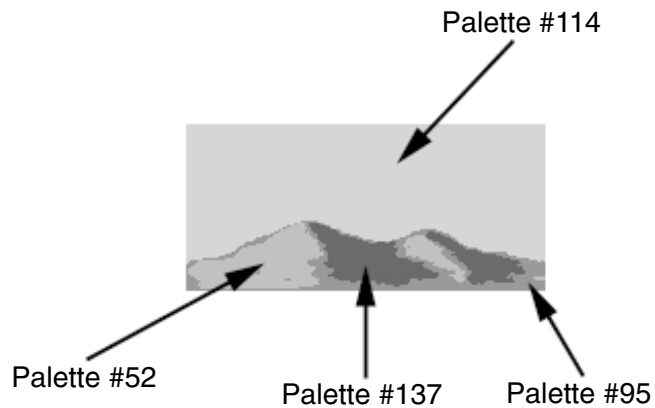


FIGURE 12-63: The Background Screen Visual Color Map

The Background Screen Design Template

Figure 12-64 shows how the seabed portion of the background was created. The simple but effective shading effect was done by using the Brush tool and Smear tool together. First, small areas of the seabed surface were painted using small brushes and then they were smeared and blended to produce the shading shown.

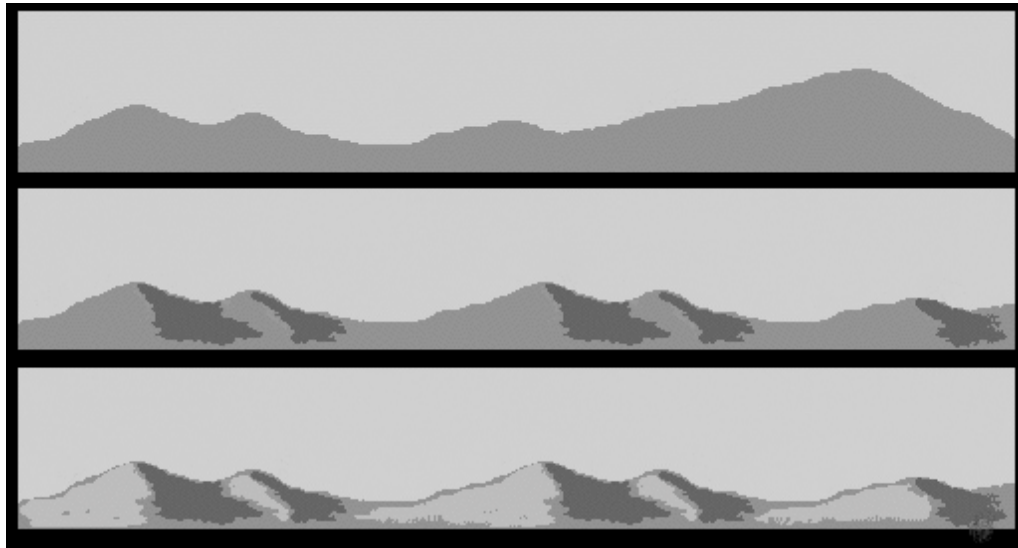


FIGURE 12-64: The Background Screen Design Template