Game Interface Design

Brent Fox

THOMSON
COURSE TECHNOLOGY
Professional ■ Trade ■ Reference

ISBN: 1-59200-593-4

Library of Congress Catalog Card Number: 2004111222

Printed in the United States of America

04 05 06 07 08 BU 10 9 8 7 6 5 4 3 2 1

Thomson Course Technology PTR, a division of Thomson Course Technology
25 Thomson Place ■ Boston, MA 02210 ■ http://www.courseptr.com

SVP, Thomson Course Technology PTR:
Andy Shafran

Publisher:
Stacy L. Hiquet

Senior Marketing Manager:
Sarah O'Donnell

Marketing Manager:
Heather Hurley

Manager of Editorial Services:
Heather Talbot

Senior Acquisitions Editor:
Emi Smith

Senior Editor:
Mark Garvey

Associate Marketing Manager:
Kristin Eisenzopf

Marketing Coordinator:
Jordan Casey

Project Editor/Copy Editor:
Estelle Manticas

Technical Reviewer:
Les Pardew

PTR Editorial Services Coordinator:
Elizabeth Furbish

Interior Layout Tech:
William Hartman

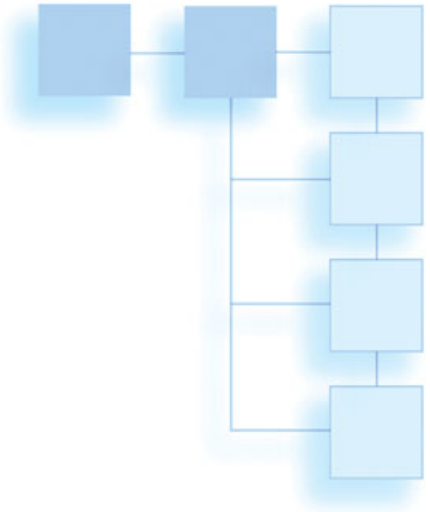
Cover Designer:
Mike Tanamachi

CD-ROM Producer:
Brandon Penticuff

Indexer:
Kelly Talbot

Proofreader:
Gene Redding

Chapter 2



Planning Menu Flow

Planning is vital to a successful interface. If budget or time requirements are tight and corners must be cut, then cut something else. If you cut out the planning stage, your project will probably end up taking longer and costing more than it would have with careful planning. If you really want to complete an interface design quickly, spend more time planning. A large-budget project may afford the luxury of more experimentation and trials, but with a lean project, you need to get it right the first time.

Why Is Planning So Valuable?

The best way to speed up the design process is to only do things once, and careful planning gives you a shot at getting things right the first time. It may seem impossible to get a perfect interface on the first attempt, but if you don't plan your process, then you almost guarantee that things will need to be redone. You know that you will make changes, but that doesn't mean you should avoid planning. If you don't plan, you will end up having to make even more changes.

Without good planning, it is hard to know what needs to be done. It is easy for artists to waste time creating art for things that will never appear in the game. How many screens are needed? What pieces of art can be re-used in different areas? What information must be displayed in game? All of these questions should be answered in the planning stage. Good planning will generate a list of assets that are needed and there will never be a question of what art needs to be created.

Just like with everything else in the game development process, you should strive for the best but plan for

the worst. In the ideal scenario, all of the details can be planned in advance and never changed. But in reality, some changes will always be necessary. Plan time for revisions but do all you can to avoid them. A trap many interface designers fall into is, when they see that time is planned for making changes, they use this "extra" time to justify incomplete work. "I can always fix it later," is a bad attitude. What usually happens in these cases is that the final polish is never added, and the game ships with an inferior interface.

Solid planning can also help determine schedules. If the design requires 100 screens, each with unique art, it may just take too long to create a fully animated 3D scene for each of these screens. If the design can be simplified and the resulting design only requires a handful of screens, then more time and attention can be given to each screen. More time will be required if the game design is so complicated that a large number of options and a lot of information must be displayed. You can make more reliable time estimates if the quantity of art needed for the game can be determined.

If I know how many screens will be needed for the front-end menu, then I can simply do a little math and know how much time I have for each individual screen. This will make it easy to determine if I am on schedule or if I am falling behind. You will need to ramp up and expect that the first tasks will take longer than the tasks at the end. If you have designed five other screens, it will be easy to make a sixth screen that fits into the design of the first five. Of course, make sure not to cut it too tight—add some time for revisions and adjustments. Without a good plan, it is difficult to know how much time you can spend on a task and if you are ahead or behind.

Accurate scheduling can reduce the panic level at the end of the project. There will always be a push at the end of the project, but you will be able to better manage things if you have a plan early on—you can attack problems early. A common result of poor planning is a string of "all-nighters" at the end of a project. This is one of the situations that cause your significant other to insist you find a job outside the video game industry. While crunch time is part of the industry it can be greatly reduced by smart planning early in a project. This will only be beneficial if you are willing and able to react and adjust to the risks you identify early in the project.

Creativity in Planning

At first glance, the planning process may not seem very creative. It might even seem boring. Charts and graphs that are kept intentionally visually plain are often used in the planning stage. Little or no art is being created, and it can seem very tedious to someone who is bursting with creative energy.

If you have a complete understanding of the entire interface design process, this attitude will vanish. There are some very important and potentially creative decisions that are made in the planning process. Truly creative and innovative ideas can be conceived at this point. You can ask questions like, "What if we tried it this way?" Arguably, these ideas can be more important to the ultimate game than pretty art. It may be more important that you decide to have a 3D animated character guiding the user through the interface than it will be if you take

the same approach as in every other game and just make a cool logo. In the planning stage, you can make these types of decisions. You are choosing the most important places in which to put your time and effort. If a 3D guide is not going to significantly add to the user's experience, you may find it better to spend your time creating cool, animated transitions between screens.

An interface designer who truly understands interface design can plan for creative and elegant concepts in the planning stage. It is much harder to find a designer who understands this concept and can come up with creative ideas that are also possible under the budget. These designers are more valuable than those who can create only cool-looking art. Creative planning is an area wherein an interface designer can become truly great. If you can come up with great ideas and you can skillfully execute these ideas, you will be in demand.

Getting Approval

Another reason for good planning is to get proper approval before starting to create art. A well-planned interface offers the game designer, project manger, or other team members the opportunity to give the go-ahead or voice concerns. Time can be wasted if the decision-makers decide halfway through the project that everything should be done differently. A producer could decide that the user should be able to choose a weapon before starting a level. If the producer has a chance to review the plan early, he could point this out.

You won't be able to avoid changes from the person in charge, but you can reduce the amount of changes you get. You can make it much easier for the producer to give you direction early if you present him with a good plan. He may not even know what he wants himself, and planning can help him to figure it out.

As you are presenting your plan to your boss or publisher, make sure that he understands that you are seeking approval and that this is the best stage to provide feedback. If the person in charge glosses over the plan, it can cause problems later. If you can make adjustments now, everything will go much more smoothly.

Getting approval can avoid your being blamed later because you can always refer back to the approved plan. Once you have received approval, it should never be used as a threat. This will make everyone hesitant to give you approval. You don't want to give the impression that you will fight against any reasonable changes after your plan has received approval, but the person giving approval needs to understand that there is a certain level of commitment in the approval.

Get as much information as you can before you even start to plan. This will help you get approval faster. If you understand the expectations, it is much easier to get approval. The more difficult problem comes when the publisher or producer doesn't know or can't articulate what he wants. This is where good communication skills come in. It is your job to understand what the producers or publishers are looking for and give it to them, even if they don't know what they want.

After a plan has been laid out, sufficient time must be spent evaluating the details of this plan. This should be

done by anyone who has veto power. You can help make this clear by asking something like, "I need to get your approval on this layout to make sure that everything is the way you want it. This will help avoid changes." A polite request like this one conveys the idea that you expect this to be final approval. Just because flow charts don't look pretty doesn't mean they don't deserve serious consideration.

Interface Planning Helps Game Design

A detailed plan for a video game interface can really help drive game design. Fleshing out all of the details in the menus and the HUD will force many game-play decisions to be made early. It may also bring up important issues that may not have been considered until later in the game-creation process. The game designer may change actual game-play based on serious consideration of the interface design.

You can ask a lot of good questions in the planning stage. These questions can stimulate the imagination of the game designer. For example, a seemingly simple screen wherein the user chooses an environment can prompt questions that will help determine the game's ultimate design. Will the user be able to choose between different environments? How many choices will he have? Will some environments be locked and not available to the user until he completes certain tasks? Can the user choose an environment in every mode of the game or are there some modes that will dictate environment choice? Will environment choices, in the menu, be affected by other choices made during game-play? The questions could go on and on. They are questions that affect the flow of the menu. It is easy to see how game-play can be affected when serious thought is given the interface. It is difficult to have a solid plan if the game design is underdeveloped.

Don't forget to show your plan to a programmer. The planning stage is a great time to get feedback and suggestions from the programmer who will be working on the interface code. The features planned in an interface greatly affect the programming schedule. Something that may seem easy to an artist can provide a big headache for a programmer. If you plan on playing full-motion video in your menu while animating buttons on top of the movie, you will need to see if the engine has this capability. If you plan on using real-time 3D in the front-end menu, the programmer will need to make this possible. Do your best to work well with the programmer. Both of you will have to work as a team to get a functional interface—No one can go it alone.

Game Design Goals

A good way to for a game designer to make decisions about the features of a game is to have goals. If the interface designer also understands these goals, it will be much easier for him to make decisions about the interface. It's not always easy to define the overall game goals, but if the game designer takes the time to create concise goals for the entire game and the interface designer clearly understands these, many decisions will be easy to make. Goal-oriented design produces great results.

You may be wondering, "What kind of goals do I need to set when designing an interface?" *Make the coolest inter-*

face ever may be the first thing that comes to mind. This goal sounds great, but it probably shouldn't be the first priority. As much as everyone wants a cool interface, there may be other things that are more important. If you are making a kids' game, for example, it might be more important that the menu is easy for a six-year-old to use than that it looks cool. Prioritization is key to using these goals to guide your design.

The game designer, publisher, and project manager may need to work out the game goals. Everyone will need to agree on the goals. Each of these people has a different role and may have a different prospective. If they can agree on a prioritized list of goals, it can help everyone work together.

Some goals may not be what you would expect. For example, the first goal of the game may be to reach a broad market. If you are creating an online game that will be used to promote soap, the client's goal may be to reach a broad audience. This goal may not lend itself to the type of design that hard-core gamers might consider the coolest interface ever. You might choose a completely different art style because of this goal. You might use a little less rust and grunge than you would use in a game aimed at hard-core gamers, for example.

Possible Game Goals

Below is a list of possible goals that a developer or publisher may have. These goals may not match perfectly with your personal goals, but it is important to understand the goals of the guy in charge. This list is by no means a complete list of goals that could ever be used for game design. In fact, it is a very brief list. This list is just meant to stimulate thought about the real goals of your next game.

- Promote an existing license or famous personality.
- Capitalize on an existing license or famous personality.
- Meet a particular schedule.
- Reach a particular audience.
- Create something completely unique.
- Outdo a competing game.
- Capitalize on the success of a competing game.
- Continue a successful series.
- Sell another product (other than the game itself).
- Promote a moral issue.
- Create a buzz using controversy.
- Create an educational experience.
- Pass the approval process of the console manufacturer.
- Please the Marketing department.
- Tell a story.

Don't treat goals lightly. As an interface designer, if you are not given goals for the game, you might want to present the goals that you think everyone would agree on. This will give you a chance to see if you understand what everyone else is expecting. Ask questions about the target market, subject matter of the game, budget, schedule requirements, what the most important feature of the game is, what makes the game unique, what other games might be similar, and so on. A clear understanding of all aspects of the game will make it much easier to understand the overall goals of the game.

If you have input on creating the game goals, be honest with yourself. Get honest information from the game publisher. It may take a little coaxing to get a publisher to give you the direction you need. It is, however, very important to understand the publisher's vision. He is paying for the project, and therefore his goals and opinions are always the most important. It won't do any good to pretend that the number-one goal of the game is to be innovative when the number one goal of the publisher really is to reach a sales goal.

If you are working on a game that has a movie tie-in, the number-one goal may be to get the game on the shelf at the same time as the movie is released. This may be more important than adding cool new features. In such a case, the schedule should rule. Any feature that has the potential to delay the project may need to be scrapped, even if it would make the game "super cool." When making decisions, it will be easy to throw out anything that would jeopardize the schedule and choose the features wisely.

Breaking Down Your Goal into Specifics

Avoid the temptation to set one large goal that is actually several goals in one. This is often the easy way out—it is more difficult to articulate specific goals than it is to generalize. But a goal like "Make a cool game" is not nearly as clear as "Add three new and creative features that are not found in competing racing games." You could even break down this goal into several more detailed goals: Create one new feature that appeals to avid racing game fans and add two new features that appeal to the more casual gamer. All of these goals could be a subheading of "Create a game that will sell more copies than the last version." The point is to define useful goals that will provide direction during development. Understanding the motive behind the goal is very important.

How Priorities Affect Decision-Making

Now think about how priorities can affect decisions. Let's say that the first priority of a game is to be educational and teach kids about animals. A secondary goal is to make the interface intuitive and easy to use. This secondary goal is important, but it is lower on the priority list. If you are armed with this information about the game's priorities, decisions will be easier to make. For example, a lot of text describing the difference between amphibians and reptiles may be used in the interface. Using all of this text may not support the goal of a simple interface, but as the first goal of education is more important, the text must be kept. This does not mean that the lesser goal of a simple interface can be dumped. There may be ways to work around the text and create the best possible solutions that includes the text.

In the case described above, you may need to be a little creative and include the necessary text but not clutter an interface. One solution could be to create an information button with a recognizable icon. This button could be placed next to animals. When the information button is selected, a pop-up window appears with all of the text. This window can be closed and

other information buttons can be selected. This way, the user can easily access all of the information, but the screens with the images of the animals can be kept simple.

Charting Methods

The menu system that appears before the game begins is often referred to as the *front-end*. This term helps distinguish this menu from all of the in-game and pause menus that can appear in a game. The best way to plan and organize a front-end menu is to create a flow chart. This will give you the chance to organize your ideas. Once you have a chart, it is also easy to give this information to others for approval or feedback. A great flow chart can even allow the programmer to begin programming the interface with temporary art, before the final art is completed. A flow chart makes it easy to see what tasks need to be done.

The important thing to remember when charting a menu is to be consistent and clear. The purpose of creating a flow chart is to be organized. Clear communication of the flow of the interface is the number one goal of a flow chart. Don't worry about what the chart looks like so much as what information and options will be displayed on each screen. Just get the flow on paper and make it easy to understand. You can waste time making pretty borders and cool-looking backgrounds. Simplicity and clarity should be the governing factors.

There are many software programs that can help you create charts. I prefer to use Adobe Illustrator because I know it well and use it for other parts of game development. There are programs, like Visio, that are specifically made for creating flow charts. These programs can be much more efficient than a standard art program. Making changes should be easy. If you don't plan on using a program that was made for creating flow charts, I would strongly suggest at least using a vector program, like Illustrator, and not a raster program such as Photoshop. Vector files will be much more flexible when it comes to editing; they create smaller files and print clearly. I have seen some cool-looking flow charts created in Photoshop, but they were big files that were hard to send in an e-mail, and it was much harder to make changes to them.

I have come up with a charting method that works well for me. There are many other methods that would work equally well, but I'll share my method as an example. Feel free to develop your own methods and symbols. The important thing is that your chart includes all of the information discussed here and is easy for others to understand.

Start by creating a box that represents the first screen that is seen when the user starts the game. Make a box that is large enough to fit several lines of text. All of the options for that screen should be listed in this box. Place a title at the top. As you choose the size of the document and the size of all of the elements that appear in this flowchart, you should take several things into account. Most likely, this chart will be printed at some point. The text and images will need to be large enough to be easily read on paper. Think about the total number of screens that will appear in the flow chart, and make sure that the spacing and size will allow for all of these items to fit (See Figure 2.1).

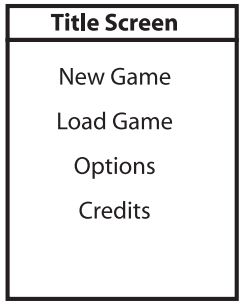


Figure 2.1
Create a box that represents the title screen. Carefully plan the size and spacing of your chart.

Tip

Your menu may become too complicated to fit on one page. Most software programs provide the option to print one document on multiple pages—when the chart is printed, these pages can be pasted together to make one big chart.

Charting may not be easy. It is very likely that you'll have questions that aren't easily answered during this charting process, and you'll need to make changes to your chart once these decisions have been made. Don't let this intimidate you. The best way to root out problems is just to get started. Take your best guess at the options that should appear in this first menu and type them into your chart. As you move onto other screens, you may discover something that will cause you to come back to the first screen and make changes. This is a natural part of the charting process.

Once you've listed the options, you need to decide where each of these buttons will take the user. Create new boxes, just like the first box, for each of these new screens. Type in all the options that should appear on these screens and decide where each of the buttons in these menus will send the user. Use this method to chart out the entire front-end menu. If important information or images (such as a tournament bracket in a sports game) will need to be displayed, make sure to list these items. Make sure that they are visually distinct from the items that will be interactive. I have created a sample of a chart that could be used for a sports game interface. All of the items that are not interactive have been italicized in this example. (See Figure 2.2.)

The next thing you need to do is create arrows that show the flow of the interface. Use these arrows to connect all of the various screens. Don't forget the Back buttons. Use a different colored arrow to represent the transition when the user presses a Back button. This will help keep things clear.

Video games often have items that are locked and are only accessible after certain tasks have been performed in the game. Levels may need to be unlocked by completing a previous level. The user may only have access to certain cars until he has won enough races. Most games have some items that are not available, or locked, at the beginning of the game. This is a good time to identify these items. Once an item is identified as one that is initially locked, it will be easier to visually design this screen later. (See Figure 2.3.) If you know that in the end 20 different soccer fields will be available, but in the beginning the user can only play in one of the three available fields, this will affect your design.

You may want to let the user know how many options are possible to unlock. In the example above, you might want to display all 20 fields and just mark some as locked. This is important to know before you start to design these screens.

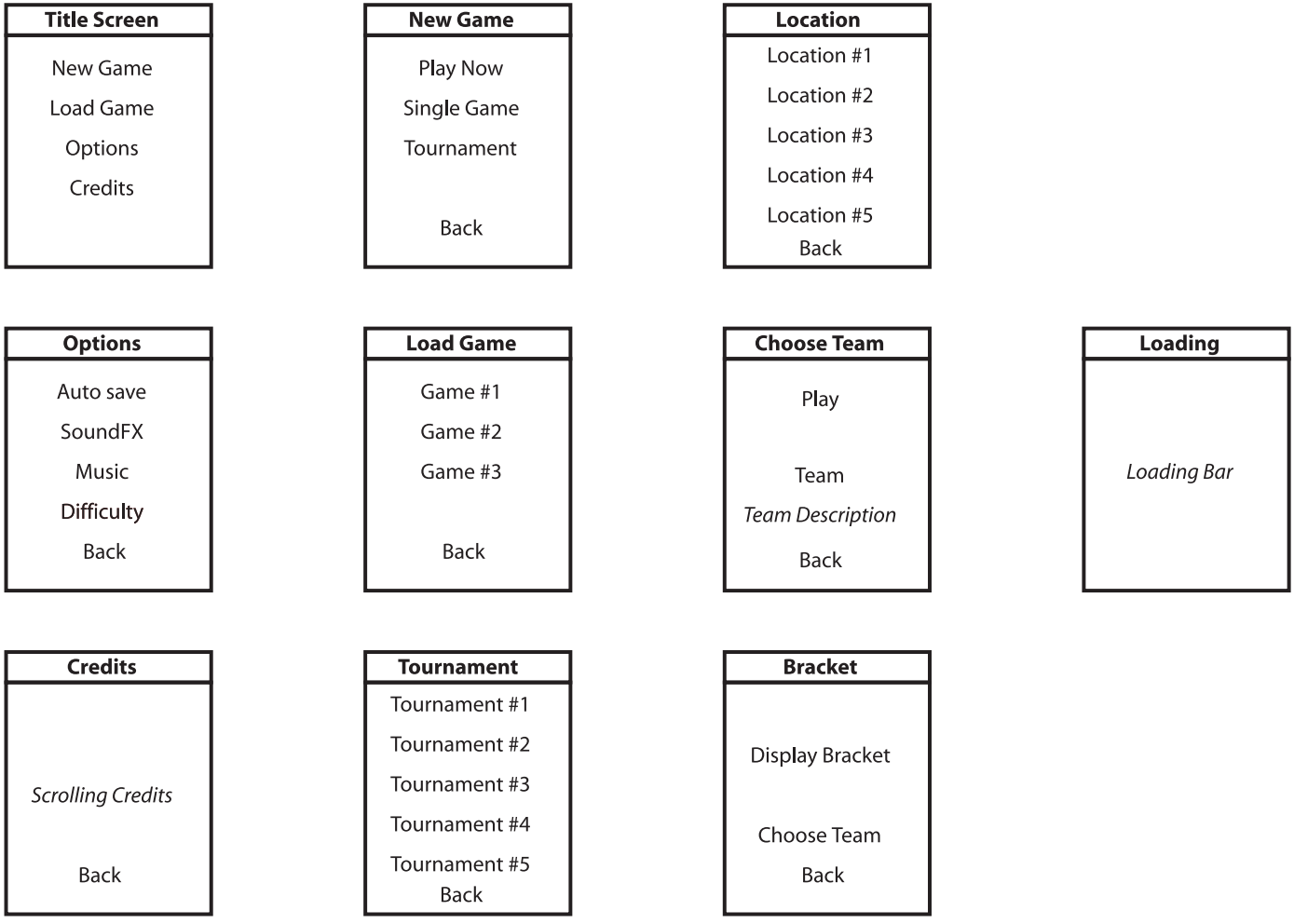


Figure 2.2 Create boxes for every screen in the menu and list all of the options on each of the screens.

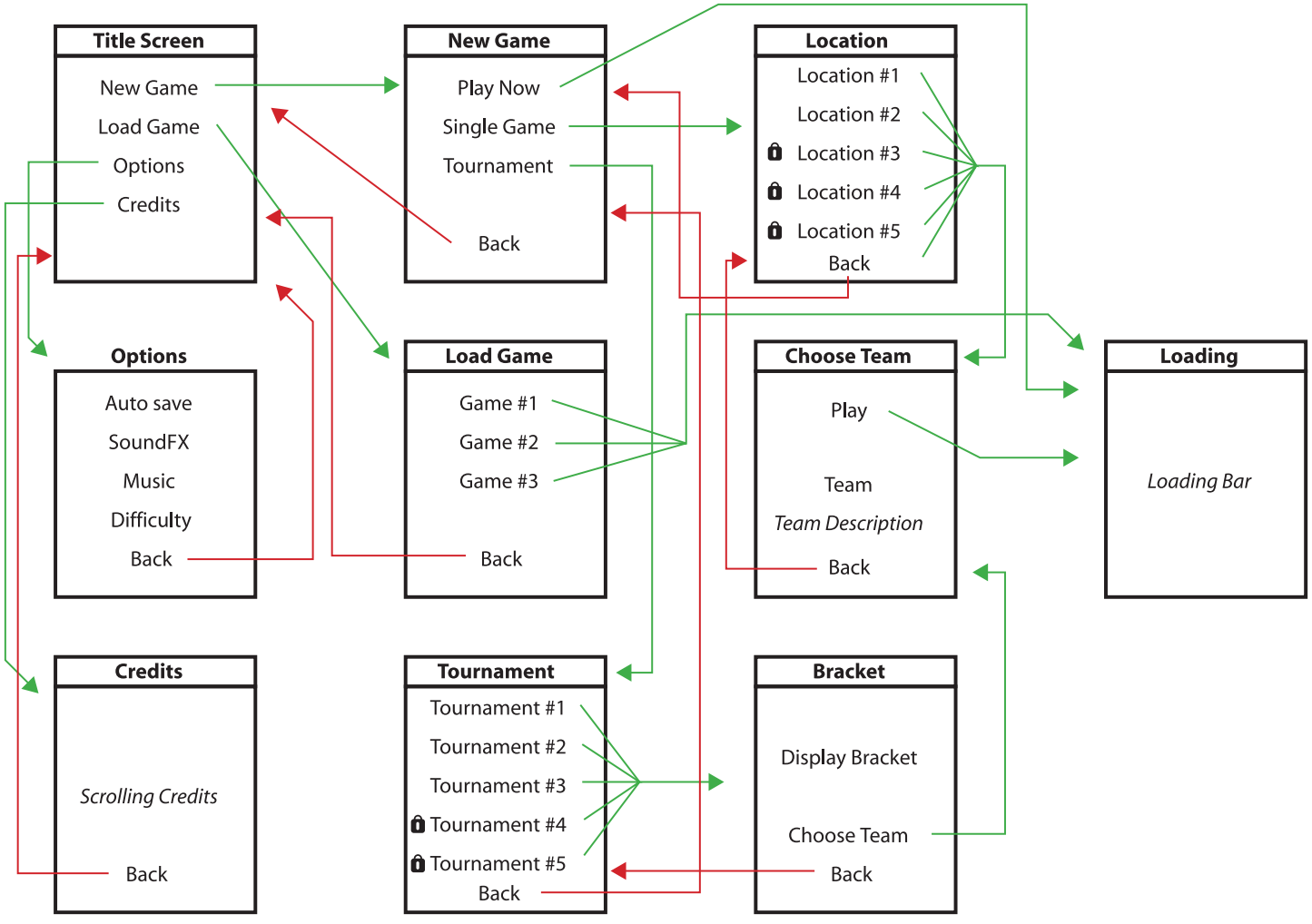


Figure 2.3 Notice the transitions and the locked items. This is a relatively simple menu.

The charting process may seem simple, but it can take a lot of thought to get it right. This first example is relatively simple and straightforward. It doesn't take long for a menu to become very complex, however. Take a look at Figure 2.4 and see how this same game, but with a bigger budget and more features, is more complex and will take more time to plan.

Notice another type of screen in Figure 2.4. These are pop-up menus. The traditional pop-up menu is different from a regular screen in two important ways. These menus are not stand-alone screens; they appear on top of the current screen. When they are activated, they "pop up" over the current screen. The screen that was visible before the pop-up appeared can often be seen in the background. The pop-up menu only covers part of the screen. Pop-up menus often appear when there is a small amount of information that needs to be displayed and this information does not justify a full-screen menu.

A common technique, used to avoid confusion, is to darken the previous menu, which is in the background. This way, the user does not think that the buttons on that part of the menu are active. Occasionally, a pop-up menu can advance a user onto a brand-new screen, but it is much more common to have these pop-up menus close and return the user to the screen he just came from—they are sort of "dead ends" in the menu flow. Because they are often small, they typically do not contain as many options and information as a full screen does.

Button Types

This is a good stage at which to examine the interactive aspects of the menu. How does the user make adjustments or choices? Most menus have several different ways of accepting user input. Buttons, sliders, and toggles are very common and are all used in many menus. The trick is to choose the appropriate input method for each item. What information does the user need to input, and what is the simplest method to get this information?

I will assume that the user will be using a mouse or a controller as an input device. There are, however, many other options. Voice recognition technology has improved and there are some games that use a microphone as an input device. Some systems use other input devices such as guns or drums. Not as many interfaces have taken advantage of these interesting input devices. This does not mean that there are not any great solutions that use this non-standard hardware.

The way to get input from the user is to detect which buttons have been chosen by the user. The user makes choices that advance him through the menu. Every time the user makes a selection, the game engine detects these choices and the appropriate changes are made when the game starts. Simply put, the user chooses features such as Single Player mode and Easy difficulty level by pressing the buttons or pressing Select on the controller. When the game starts, these settings take effect.

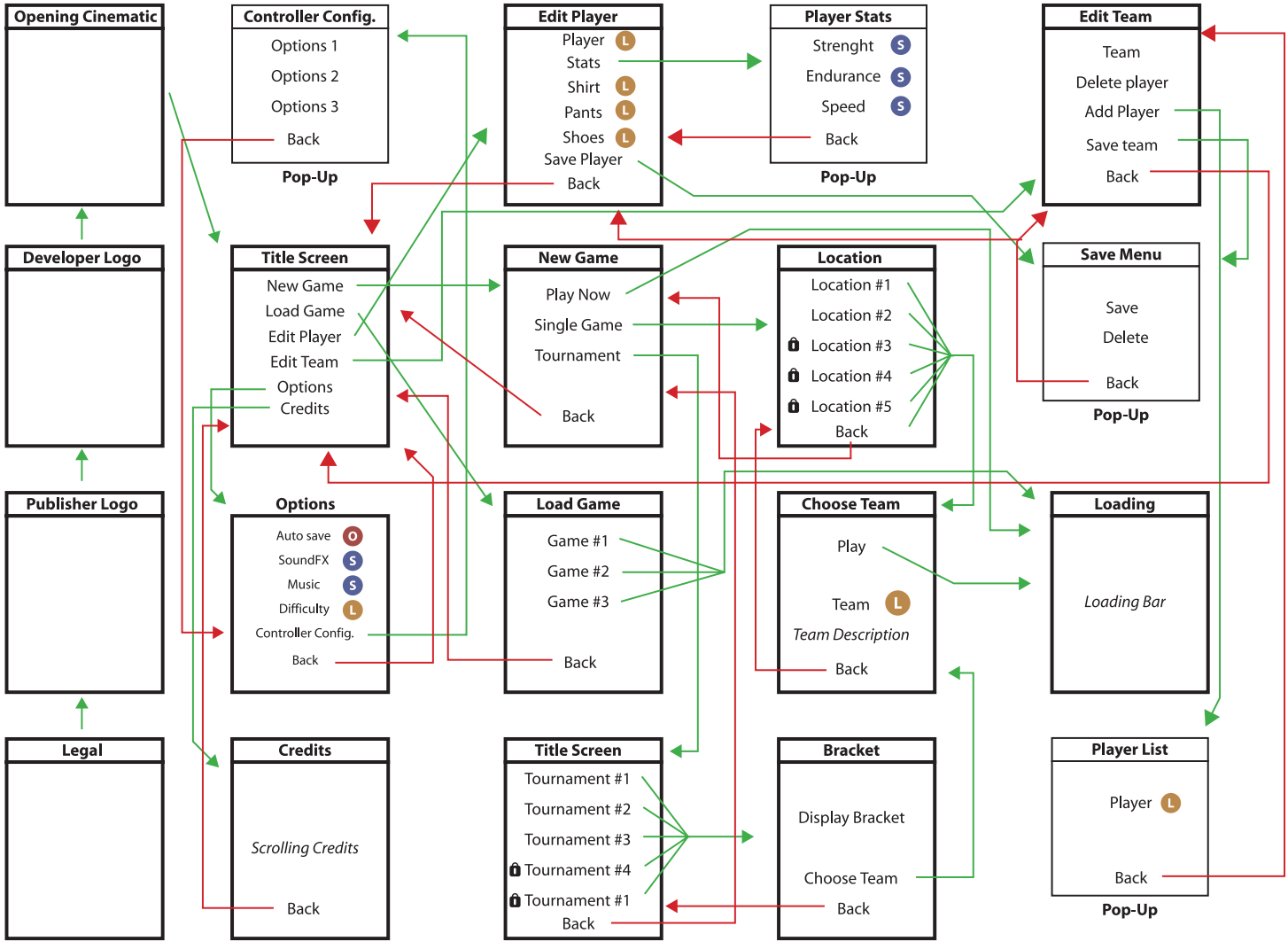


Figure 2.4 It is easy to see why a flow chart is necessary to visualize how the menu will work.

Sliders

Sliders are a great way to adjust values that have a wide range of possibilities. If the value that is being adjusted only has a small number of distinct choices, such as Easy, Moderate, and Difficult, then other methods work better. Numeric values like a range from 1 to 10 or 0 to 100 percent work great for sliders. Music volume is another good example of an input that could use a slider—the user can move the slider left and right and get a wide range of volumes. When using sliders, the settings are typically remembered by the game engine. If the user sets the volume at 3, and leaves the menu, the volume should remain at 3. If the game system has a method to save information on the hard drive or on a memory card, this setting should also remain the next time the user plays.

Just because you're using the standard input method for a slider doesn't mean there isn't any creativity involved. All sliders don't need to look the same. This is another area where you are only limited by your creative ability—the only thing that is important is that the user recognizes the control as a slider and instinctively knows what to do with it. (See Figure 2.5.)

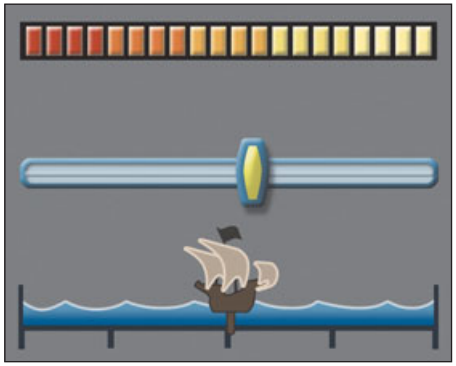


Figure 2.5 Be creative when designing the look of sliders. Just be sure not to confuse the user!

Toggle Switches

You can use a toggle input method when there are two possible states, such as On and Off. Methods for displaying these two states and getting the user input can vary. As the word *toggle* implies, the user can change from one state to the other by *toggling* or changing the switch.

A radio button is a commonly used toggle switch that is placed next to text or an icon. The text or icon represents one of the two states. When the box is checked or the button filled, this means that the state represented is on. If it is empty, the state is off. For example, a radio button may be placed next to a line of text that reads "High Level of Detail." If the circle is empty, then High Level is off. If a dot appears in the circle, then this option is turned on. A radio button does not always have be a circle with a dot. For example, the same functionality could be accomplished with a check box. It is important that the empty version looks empty and the filled version looks filled. The user should be able to recognize it as a radio button.

Changing a toggle switch can be accomplished using several different methods. When playing a console game, the user can make a change by moving the control stick left and right when the option is highlighted. The toggle can also change as the user presses the Select button when the

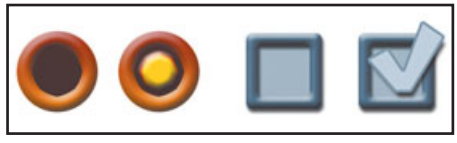


Figure 2.6 These are some standard looks for a radio button.

option is highlighted. Look at the methods used by other games and see what will work best for your game. The goals are to use what you think will come naturally to the user, and to be consistent throughout the entire menu.

Lists

Lists are used in many different ways. If all of the screen options can be seen on the screen at once, then the user can move the selection indicator to the option he wants and the options remain in place. (See Figure 2.7.)

If there are too many options to display all at once, you can handle the situation in a couple of different ways. One method is to keep the selection indicator stationary and the options scroll on and off screen and move into place. (See Figure 2.8.)

You can use a combination method, but this can become confusing if it's not executed well. In the combination method, the cursor moves from option to option if it is visible on screen. Once the selection reaches the last visible option on the screen, and the user continues to move it in the same direction, then the selection indicator remains still while the options move onscreen. (See Figure 2.9.)

Input Text

In some cases, the user may need to input text. For example, the user may be given the option to enter a name for his character. This can be a simple process when playing a game on a PC, but it can become very complicated

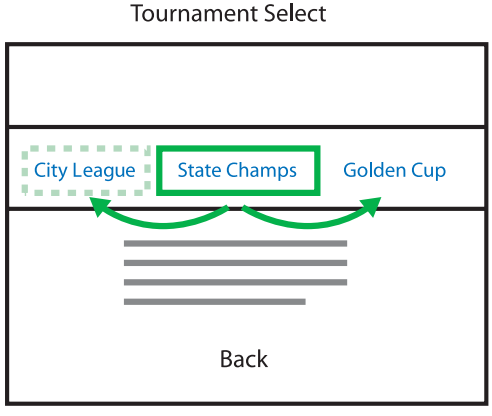


Figure 2.7 The selection indicator moves, and the options stay stationary.

Figure 2.8 The selection indicator remains still and the options move.

without a keyboard, as when playing a game on a console.

This challenge has been addressed in basically the same way ever since the old arcade games asked the user to enter initials after receiving a high score. The most common method is to make three or four rows of text. This list should include every letter of the alphabet and any characters that can be used for input. There are also two additional options: Delete and Done (the Done button is only needed if the number of characters is not pre-determined). One of the letters is always highlighted, and the user can move this highlight left, right, up, and down. When the correct letter is highlighted, the user presses the Select button on the controller and this letter is added. If the user makes a mistake, he can select the Delete button and delete one letter. Each time the Delete button is selected, another letter is deleted. This method is effective and it can work with a very simple controller, such as a joystick and one button. The problem with this method is that it can take a long time to enter a name—it is much slower than typing. Figure 2.10 shows an example of this method for entering text.

Another approach is to start with one letter blinking. If the user moves the joystick up, the list cycles backward through the alphabet. If the user moves the joystick down, it moves forward through the alphabet. In both cases, the letter cycling wraps around and continues when at the end or beginning of the alphabet. When the

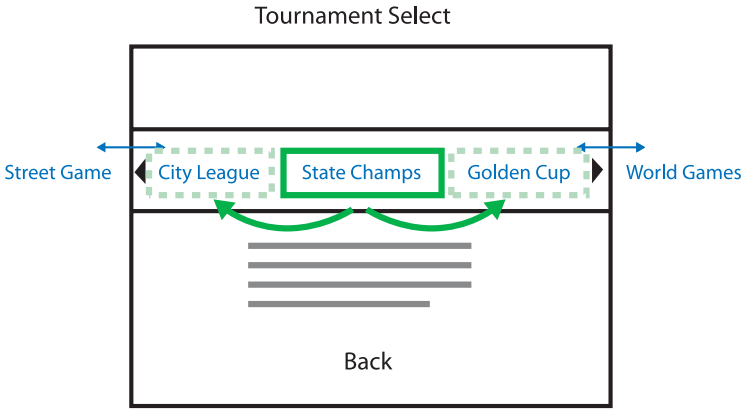


Figure 2.9 The selection indicator moves until it reaches the edge, and then the options move onto the screen

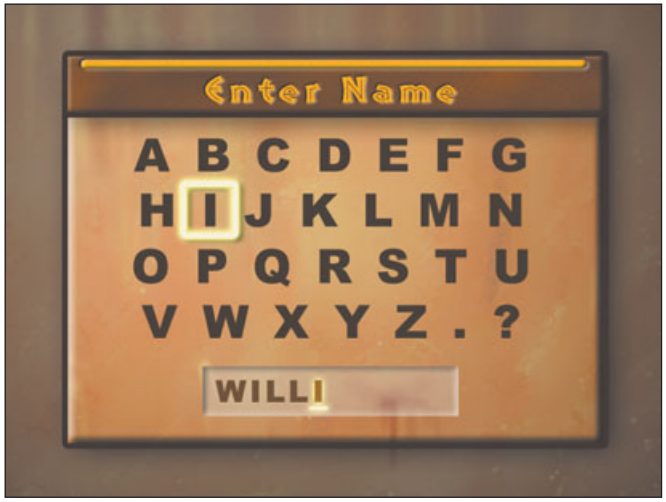


Figure 2.10 This is the traditional solution for entering text on a console or arcade machine.

Figure 2.11 This solution is also common. The advantage to this solution is that it uses much less space than the one shown in Figure 2.10.

user moves the joystick right, another letter is added. When the user moves the joystick left, a letter is deleted. Pressing the Select button on the controller finalizes the input and advances the menu.

These are just some of the most common solutions for allowing users to input text. There are many more that have been used, and I have heard some solutions described that would make the entry process much faster. I challenge interface designers to come up with a better text-entry solution for a console game, but I do so with caution. A new method can cause confusion and frustration in users. It must be very intuitive or explained really well. Neither of the methods described here is simple. The first time someone uses these methods to enter text, he is usually a bit confused. These methods do, however, have a huge advantage: Almost everyone who's played a game has used one of these methods and is familiar with them—in fact, users have come to expect them.

Drop-Down Menus

Drop-down menus are commonly used as an input method in PC games, but they are seldom used on a console game. Technically, a drop-down menu is just another way to use a list of items as input. It is just hidden until the user selects the correct option. Drop-down menus are a common solution in most PC operating systems and are very standard with application software. There aren't many software applications that don't use drop-down menus extensively.

While drop-down menus can be useful, remember that you are making a game. A lot of the Windows conventions may be very familiar to the user, but they aren't much fun. In general, it is better to get away from the feel of the operating system and make your game *feel* like a game. Drop-down menus typically don't make you feel like you're playing.

If you determine that a drop-down menu is still best for your game, you can at least do some variations. These menus can drop to the side or "drop" up. Interesting animations used for the transitions can also help give these menus a fun look. Just remember that you are making a game.

Other Variations

Not all of the methods for accepting input are listed in this book. In fact, there are many still waiting to be con-

ceived. This is an area where vision and creativity can have a great impact on game design. Just be very careful when implementing new methods—if a new method makes the interface confusing, then it is not a better solution, even if it looks really cool. The user should know what to do without having to think very much. Stick with generally accepted methods most of the time; the user is familiar with these methods and will expect them. Use new and innovative methods sparingly and only when they will have a positive impact on the design.

Common Menu Screens

Some of the sample menus that were charted previously were simple. A big-budget game with a lot of options can get very complicated. Take a look at Figure 2.4. You can see that creating a flow chart for a menu like this can be complicated. I've worked on games that had interfaces that are even more complicated than this sample.

Below is a list of some commonly used menu screens. Again, this is by no means an exhaustive list. There are many other screens that appear in many games. Looking at this list may help you to begin to make a flow chart for your next game. Think about which of these menus could be used in your game and what your game might need that is not listed here.

- **Legal screen**. This can be a short sentence or a screen full of text. It will include legal issues like copyright notices.
- **Publisher logo screen**. It can be a requirement to show this screen before the developer logo. There may also be a requirement for how long it needs to be displayed before the user can move on.
- **Developer logo screen**. This is where your company logo is seen.
- **Console logo screen**. Some consoles require or encourage developers to display the system logo.
- **Title screen.** The name of the game appears here. It can also include interactive options.
- **Options**. An option screen allows the user to change many game settings.
- **Credits**. This is where everyone who worked on the game is listed.
- **Environment or level select**. This is used in games where the user can choose a level of location to play.
- **Player editor**. A player editor will allow the user to change the look and attributes of characters in the game.
- **Information**. This screen can have extra information, such as story, maps, and so on.
- **Save / Load game**. This screen lists how many games are saved and allows the user to load and save games.

Simplicity versus Depth

After you have created a flow chart, it's time to evaluate the flow of your

menu. Before you move on to creating art is a good time to make changes to the flow. It is much easier to change your chart than to change elements after art is created and is working in the game. The goal is always to make the menu simple and easy to use. Ask yourself the following questions:

- How many options appear on each screen?
- Are they logically grouped?
- How fast can a new player get into a game?
- How fast can an experienced player get into the game?
- Which options will the player want to adjust often?
- Which options will be changed rarely?
- How does this menu compare with similar games?
- How many dead-ends are there? (When the user must back up to start the game.)

When designing interfaces, you should try to limit the number of screens you present. You should also try to limit the number of options on any one screen. These two goals sometimes conflict, and you must balance your solution. The user does not want to be forced to go through a lot of screens to accomplish what he wants to do. At the same time, if too many options are listed on a single screen, the menu can be more difficult to negotiate. It is much easier to quickly understand small amounts of information. This applies to both console and PC games, but too many options on a console can be even more problematic than on a PC. If there are too many options, screen resolution limitations can make them hard to read; it can also be tedious to scroll through a long list of items with a controller. A mouse allows the user to go directly to the option he is looking for. (The user will still need to read and comprehend all of the options on a PC game.)

A good test to see if your interface is designed well is this: When a user begins your game and just hits the Enter button repeatedly, can he get into the game? How many presses will it take to get into a game? Which options will be chosen with this default method? If your game has an option such as Multiplayer that will be used often but does not fall in this default path, how easy is it to choose this option and get into the game?

Make sure that the user is only forced to back up (go back to a previous screen) when he is adjusting options that most likely will not be accessed often. For example, music volume may only be set once and then left at the same level every time the user plays the game. Another option is to allow the user to start a game from what would be a dead end without a Start Game option. This doesn't always make sense, and it will only work if there aren't any other options that need to be selected before beginning a game.

Planning for HUD

What is HUD? It is not a replacement for a swear word. HUD is short for Heads Up Display, which refers to the interface that is displayed during game play—stuff like the radar, health meters, and score.

Because of the nature of the HUD, a flow chart is not typically necessary,

but organization is important. You will need to know all of the information that will need to be displayed during game play and you will need to know any options that might need to be accessed in a Pause menu. You may need to get the information from a game designer. Get it as soon as possible. Many other screens may be needed for a game. Pop-up menus may occur at various stages and the player may need access to other information that is not visible in the HUD while playing the game. The user may need to check inventory or look at a map, for example. Do your best to get all of the information and list all of the possibilities that could occur during the entire game.

Because of the nature of the HUD, no matter how well you plan in the beginning, the game will change at least a little before you're done. It changes because of the close tie with game-play. You will discover that the user needs information that you did not anticipate displaying. You may also learn that when playing the game, some information is not necessary and should be removed. Many of these problems won't be discovered until the game is at a stage where it can be played. You will need to be flexible and even look for ways to improve the HUD.

I have changed the HUD partway through the development process on most of the games I have worked on. A great example is a change we made to the game demo on the CD of this book. We began by displaying a number as a percentage for the rating of the station. We later discovered that this number was confusing—the test users were not sure what the number meant. We changed to a five star system. I created five empty stars and the stars filled in as the rating increased. The disadvantage is that the user did not know if he was at 3.2 or 3.3 stars, but this information could be checked in the Goals pop-up menu. We found that this was a much better solution. All of the users seem to intuitively understand what three stars meant.

Get our your pencil and begin to sketch your HUD early. What you need to design will vary greatly from game to game. It also can be a little harder to design than the front-end interface, because there are fewer ways to chart the flow HUD. Just because the HUD is a little harder to plan does not mean it should be left until later. Get all of the information you can. List everything that could be possibly displayed! Prioritize all of the display items and determine how the user will access information that is not typically displayed on-screen. Determine what will be automatically displayed and what will require a separate menu. Identify any game events that will change the HUD. Organize all of the information to be displayed into logical categories, and plan as much as possible!

Creativity versus Conventional Methods

Most interface designers are bursting with creativity. They want to do things better than has been done in the past. They want to discover and implement original ideas—this is what makes the video game industry fun! The passion and desire to continually improve are essential.

When designing an interface (or working on any aspect of a game), remember that thousands of creative

people have been doing the same thing for a long time. Most likely, someone else has already thought up what seems like a new idea to you. If it has never been implemented, then there might be a good reason. Be cautious when trying out new ideas—use your creativity wisely.

Video-game players have also come to expect the thing they have already experienced. Take advantage of this often by using these conventional methods. Don't give up on creativity, but don't think that something is better just because it is different. Make sure it really *is* better.