The JACK PRINCIPLES

of Jellyvision interface and design

The collective vision from
the interactive design experience at Jellyvision

First Edition
©1997, Jellyvision Inc.

• • •

written by

Harry Gottlieb

Table of Contents

A disclaimer:

Some of the arguments and design concepts herein may seem almost ridiculously obvious . It is, indeed, the simplicity of *The Jack Principles,* that gives us hope that they may be broadly relevant and helpful to the growth of the interactive industry.

A Definition
for Jellyvision Programs

The majority of television programs have three qualities: they have moving pictures with synchronized sound, constant pacing and communicate primarily through words.

Specifically, we define Jellyvision programs as follows:

> **An experience delivered through a screen where sound & picture are synchronized, pacing is constant, and the program responds to the individual input of each user primarily through words.**

The Jack Principles exist to help our designers create Jellyvision programs based on the above definition. Throughout this document, when "jellyvision" is mentioned, it is specifically referencing this definition. If this is sometimes confusing--differentiating the name of our company and this form of communication – sorry: we couldn't agree on another name.

The Defining Quality—Modeling Human Conversation

A jellyvision program has a particular defining quality that separates it from all other forms of communication: it feels like someone is talking with you. Indeed, a jellyvision program appears to create a continuous conversation between the character in the program and the human sitting in front of the screen.

This doesn't mean the conversation can be about *anything* the way a real human-to-human conversation can. The topic is constrained by the goals and design of the program's creators. If the host of a program asks you:

> "So, do you like books about politics?"

As the user of the program, you can't respond by saying, "Speaking of politics, what do you think about this silly confirmation hearing going on in the Senate?" You can only respond with one of the choices the program recognizes (which in this case might be "Yes", "No" or "Sort of").

The Suspension of Disbelief

It appears, however, that in a well designed jellyvision program, the audience will accept those inherent limitations without question. They accept those limitations so that they can buy into the illusion that the character is really talking *to them*. At a movie, we find ourselves watching a sweet, little orange alien with a penchant for Reese's Pieces and allow ourselves to forget that such a creature doesn't actually exist. We do this quite naturally. It allows us to be entertained by the film. This phenomenon is commonly known as "the suspension of disbelief."

Jellyvision programs can work because a user will suspend her disbelief and buy into the illusion that the character is really talking to her.

> "So, do you like books about politics?"

You select "No."

> "No? Not even historical politics? Lincoln-Douglas debates...that sort of thing? Any interest in that?"

You select "No."

> "O.K., no problem. I won't recommend any books on politics....
> ....How about sci-fi? Are you interested in science fiction at all?"

In this case, you are responding by just saying "no." If it were a real conversation with a human, you'd probably say "no" and start explaining why you don't like politics. With a jellyvision program, however, it appears people will naturally accept limitations on their responses. Thereby, the suspension of disbelief becomes possible.

This is true even when the questions are more open-ended. When a character in a program asks:

> "Of the movies that came out this year, which was your favorite?"

Most people will accept typing out the answer to the question directly, as opposed to changing the subject.

First time users of interactive programs that model human conversation often test the limits of the program's ability to respond. People type in swear words and all sorts of stupid things. Sometimes a program can handle this intelligently, but you can always find some way to answer an open ended question so that the program can't respond intelligently. Quickly, however, the novelty of "cracking" the technology wears off. Most people will soon get caught up in the flow and real purpose of the program, suspend their disbelief, and allow themselves to feel that the character is actually talking to them.

Shared Control

In a jellyvision program you are not so much controlling what the program does, as you are constantly reacting to prompts from a character. Control is *shared* between the user and the program.

Every moment in a jellyvision experience has been pre-defined and every path has been prefigured by the creative design team. You, however, impact which moments you experience and the path you go down. You don't definitively *decide* the path, but rather your reactions to a program's characters *influence* the path down which the program takes you.

In the earlier example, if you had responded that in fact you enjoyed books on politics, the program would have played back a different piece of dialogue:

> "Yeah? You're into politics? How do you feel about, say, autobiographies on contemporary politicians? Would you be into something like that?"

You respond "no."

> "No? How about books on the history of political campaigns?"

You respond "yes"

> "Great. I've got an excellent one I can review for you..."

Your responses have changed the sequence of events from above. You didn't consciously *decide* to hear these pieces of dialogue, but the responses you gave *influenced* the program to select this sequence. If you want to hear the host repeat that last line of dialogue over and over again, you can't. Then again, you usually can't get a human being to do that either.

Shared control also manifests itself in the way the program limits the options it gives you. A television program gives you no options at all. The Web and multimedia programs usually allow you to go anywhere at any time you want. A jellyvision program falls between these extremes. It will only allow you to do a relatively small number of things at any one time (like responding to a single question). What the program allows you to do at any moment is up to the designers of the program, not you. Reciprocally, as you can see from the example above, how you respond to the program will then influence what other things the character in the program asks you to do and possibly the order in which he asks you.

So, you are not without *influence* over what you will experience, although you cannot completely *decide* what you will experience.

This models the dynamic of talking to a human being. In a conversation, you can't unilaterally decide what gets discussed. The other person is not a machine. He can place his own limits on the conversation. He can steer the conversation in one direction, just as much as you can. The control of the conversation is shared.

For jellyvision, the sharing happens between the creative design team and the individual user. The design team arranges for all the possible experiences. The individual's actions determine which experience actually transpires.

The "feel" of jellyvision

For the sake of precision, the definition of jellyvision programs is rather formal. In terms of the "feel" of a jellyvision program though, think of a continuously flowing "conversation" between you and a character in a program whose voice is synchronized with his image and / or other visual effects.

The shorthand definition

The definition at the top of this section is a mouthful, but precise. If you need to quickly describe the key qualities of a jellyvision program you would say:

"It's cinematic and makes you feel like you're really talking to someone."

An Argument
for this definition of jellyvision and this form of interactivity

Why create programs like this? Why try to model human conversation? Why is it worth creating programs that facilitate the suspension of disbelief? Why would you give people less control? Why create interactive programs based on this definition?

We set forth this idea because we believe it describes a form of communication that by its very nature can *successfully engage anyone and everyone* to the extent that television does today.

Like television, radio, literature and websites, jellyvision programs as defined above are a true *form* of communication: it is not limited to any one kind of subject matter or service. It can be used to create games, to teach math, to sell tennis racquets, to inform people about the weather and on and on.

Our belief in its appeal comes from the fact that it "steals" for itself qualities that make TV shows so engaging while adding the allure of viewer participation.

Let's briefly review why these qualities of television are so engaging.

Synchronized Picture And Sound

Many would argue that there's nothing better than sprawling out on a big, comfy couch with a great book. However, if you've got a movie—based on that book—playing on your television in the same room, it's nearly impossible for most people to keep reading. Even with the same basic story, one form of communication nearly demands your attention over the other. The movie may not be as good as the book, but the movie, through the synchronization of picture and sound, simultaneously engages our two strongest senses. Biologically, we are predisposed to focus our attention on things that move and make noises.

Constant Pacing

Television doesn't wait for *you*. You have to keep up with *it*. To know what is going on, you have to continually pay attention. The images and sounds are constantly changing. The frame of the television screen, more often than not, reveals a world that is much more sensorally dynamic than the room in which the television sits. Leaning back in our living rooms, we get mesmerized looking into this little window where everything is constantly changing.

Communication Through Words

Most television shows are based on expression through words, typically the spoken word. Almost any time you turn on television and flip through the channels, you will find someone talking. Often, they are performing words written by someone else.

There are, of course, exceptions. There are television shows based primarily on the expression of images, music and sound effects. Music videos are an obvious example. These programs are no less engaging by their nature than programs based on words; some are more engaging.

Still, the overwhelming majority of television programs heavily rely on words because words are the dominant way human beings communicate with each other. Words allow us to convey complex and subtle ideas and emotions. Words allow us to discuss almost all subject matter. A picture is certainly worth a thousand words. Yet, often it is those words that put the picture into a context, point to its intricacies and discuss its implications and makes the picture infinitely more meaningful and relevant to a human being. The words help us understand. The words help us care.

Therefore, it is no surprise that newspapers, magazines, books, radio, film, theater, television, and even much popular music, are based fundamentally on the use of words. If there is ever to be an interactive medium that will capture a mass audience on a par with television, it will almost certainly be based on the use of words.

Distinct from Multimedia Programs and Websites

Multimedia programs and websites are, indeed, often fundamentally based on the use of words. Yet, such programs, while occasionally containing video clips, rarely have synchronized picture and sound with constant pacing moment-to-moment throughout the experience. Typically, the experience is navigational—directing the program where you want to go. The pacing is intermittent because the program needs to wait for the user to tell it what to do. There's nothing wrong with this: the ability to quickly locate desired information is what has made these two forms of communication revolutionary. On the other hand, the resulting lack of pacing means such programs are fundamentally different from television.

Distinct from Video/PC Games

Video / PC games do have synchronized picture and sound with constant pacing. By and large, however, the use of words is tangential to the core interactive experience in such games. Words are used to introduce action sequences or as side commentary to action sequences. Other games allow networked players to communicate with words—but this is not the same as the program itself using words to communicate. There are also word games, where the words are indeed the content—but the *genre* of word games can't be expanded into an entire *form* of communication.

Rarely, if ever, do video / PC games respond to the user's input *with words* moment-to-moment throughout the experience. Again, there's nothing wrong with this: the fun of most video / PC games is seeing the physical movement of objects on the screen responding to your decisions. Words are completely secondary to what makes video / PC games exciting. On the other hand, the lack of words means such programs are fundamentally different from television. Indeed, the lack of words means that video / PC games are not really a form of communication at all.

———

The reason to pursue this form of interactive programming, what we are calling jellyvision programming, is because it may be a much more engaging and effective way to communicate, not just to a technology literate audience—but to everyone.

The Jack Principles

The Principles serve one purpose: to give designers a specific set of guidelines in order to create jellyvision programs.

We define jellyvision programs as follows:

An experience delivered through a screen where sound & picture are synchronized, pacing is constant, and the program responds to the individual input of each user primarily through words.

The Principles are not abstract theoretical concepts. They are design principles that have evolved organically out of our interface work at Jellyvision. We now use these design principles intuitively with each new program we develop.

The Principles break-out into three basic sections:

MAINTAINING PACING

Principles that help a designer maintain the pacing of a jellyvision program.

CREATING THE ILLUSION OF AWARENESS

Principles that help a designer *create* the illusion that the characters on the screen are actually aware of the person sitting in front of the screen. Human conversation is modeled using these principles.

MAINTAINING THE ILLUSION OF AWARENESS

Principles that a designer must follow to *maintain* the illusion that the characters are actually aware of the person sitting in front of the screen. The audience's suspension of disbelief depends on adhering to these principles.

The Jack Principles

Maintaining Pacing

A jellyvision program has available to it almost all of the same techniques that television and film use to achieve pacing, but must also employ some techniques which are specific to jellyvision programs.

Television uses music, sound effects, movement of characters and objects on the screen, camera movement and all manner of editing techniques to give a program pacing. The way characters exchange dialogue and the unfolding of the show itself through the plot also create a feeling of movement that draws the audience into the program. Pacing means paying attention to the timing of events.

YDKJ carefully times dialogue, music and sound effects to screen movement. Like a television game show, it pulls the audiences' attention through the program in part with the anticipation of a climatic final round.

YDKJ and all jellyvision programs have an additional challenge that television programs do not. The audience has to interact with the program. The program does not have complete control of its pacing. It has to share that control with the audience.

In a jellyvision program, the audience must be *drawn* into the overall pacing of the program. This can be accomplished by use of all or any combination of the following principles:

Jack Principles to Maintain Pacing

1. **Give the user only one task to accomplish at a time**
2. **Limit the number of choices the user has at any one time**
3. **Give the user only meaningful choices**
4. **Make sure the user knows what to do at every moment**
5. **Focus the user's attention on the task at hand**
6. **Use the most efficient manner of user input**
7. **Make the user aware that the program is waiting**
8. **Pause, quit or move on without the user's response if it doesn't come soon enough.**

The following pages examine each of these principles in detail.

Maintaining Pacing

Give the User Only One Task to Accomplish at a Time

By requiring only one task to be accomplished at any one moment of interaction, the user can make a series of brief decisions quickly, and thus maintain the momentum of the program.

The beginning of YDKJ gives a simple example of this principle in action. Cookie, the stage manager, welcomes you to the show, then asks you how many players there are, then asks you for your name and then inquires whether or not you want to play a 7 or 21 question game. He asks one question at a time. A program could easily and efficiently get the same information graphical user interface style. A web page, for example, would accomplish the task using a form with several checkboxes and text fields for typing. You'd probably click a "submit" button when you were finished filling in everything. This method would get all the same information as we do in YDKJ, however, there would be no pacing. You would feel like you were on a computer rather than interacting with a show.

———

In that the basis of jellyvision programs is asking you to *react* moment to moment, often, this principle will translate to: ask the user only one question at a time. By doing this rather than placing all the questions on the screen at the same time, the program can achieve a sense of momentum —a sense of pacing.

Maintaining Pacing

Limit the Number of Choices the User Has at Any One Time

At any one moment of interaction, the fewer choices the user has, typically, the less time it will take to make a decision. Pacing is thereby maintained.

Often, web pages have dozens of links and various other options on a single page. It takes too much time to decide what you want to do, so any semblance of pacing gets lost.

In YDKJ, for each question there are only three categories to pick from and four multiple choice answers. Users can make decisions very quickly and keep the momentum of the program going.

In combination with the first principle, this will often mean: Ask the user only one question at a time, and limit the number of possible answers to each question.

Freeform Input

Limiting choices does not mean that all jellyvision programs have to be strictly multiple-choice programs (i.e. choosing a number or clicking on a button or text). On the contrary, the method of input can be anything at all. There are a number of questions in YDKJ where input is Fill-in-the-Blank. Typing always takes more time than multiple choice . Nonetheless, the program can still narrowly focus what the user is *supposed* to be typing.

In YDKJ's Fill-in-the-Blank questions, to be successful, there is only one real choice: type in the correct answer. At the beginning of the game, Cookie asks you to type in your name. There's only one real choice: type in a name. Of course, you can type whatever you want; the opportunity for response is "freeform." Nonetheless, a program can still draw the user into the pacing of the program, if it *narrowly directs what a user should do* for Fill-in-the-Blank, voice-recognition and other types of freeform input.

Here's an example: Imagine a jellyvision program that reviews books and helps you purchase them. The first question the host of the program asks is:

> "So, do you have a specific book in mind or are you looking for a recommendation?"

Let's say the method of input for this jellyvision program is voice recognition. You have the ability say anything you want. The program, however, shows you two multiple choice responses:

> I've got a specific book
> I need a recommendation.

You say aloud into a microphone, "I've got a specific book." The host of the program responds:

> "Oh, O.K. What's the name of the book?"

Well, there's only one right answer to that question: the name of the book. You've already said that you know what book you want, so you should be able to quickly announce the name of the book to the program. The set-up to the question has narrowly directed what the user should do, so pacing should be maintained.

Obviously, the program then has to have a database of all the books it has available along with audio recorded by the host in order to respond appropriately. If you say the book in which you are interested is "The Catcher is the Rye," the host might respond:

> "Catcher in the Rye by J.D. Salinger. Great. Total classic. I think you'll like it."

You might wonder what happens if the person has a specific book in mind, but doesn't know the title. One design approach would be to combine a set of multiple-choice answers with freeform input. As the host asks you for the name of the book, the text cues on the screen could read:

> The title of the book is...
> The title includes the words...
> The author's name is...
> The subject of the book is...

You would simply speak the opening phrase and add the end of the sentence (e.g. "The author's name is Salinger.") Obviously, the program's ability to respond appropriately will depend directly upon the depth and breadth of its matching database and the host's recorded responses.

You can effectively limit the user's choices through multiple choice or limit the user's choices by narrowing what the user should do at any one time (even if in reality the user's choices are as expansive as selecting a single book out of thousands). Application of this principle draws the user into the pacing of a jellyvision program and keeps it moving.

Maintaining Pacing

Give the User Only Meaningful Choices

Do not give the user a choice if the program can successfully make a decision in the user's behalf and thus continue the flow of the program.

In the final "speed round" in YDKJ, the JackAttack™, the host reviews the instructions for playing the question. Taking up to 18 seconds, these instructions are rather long. The user has the opportunity to skip the instructions by pressing the spacebar.

If, however, the users choose to play another game without first quitting the application, in the next JackAttack the host automatically abbreviates the instructions and goes right to the questions. The program already knows that the people who are playing the game have already played once, so it makes the decision to skip the instructions in their behalf, without asking them. This helps maintain the pacing of the game.

Meaningful Choices at the ATM

A real world example of when this principle could be applied was noted by one of our directors at Jellyvision, Terry Hackett. He observed that every time he goes to get money out of his local ATM the screen says: *"Hello Terry Hackett"* and then asks him if he wants to conduct the transaction in Spanish or English. For four years, probably 200 trips to this one ATM, Terry has always, always said that he wants to conduct the transaction in English. Not only that, he only takes money out of his checking account and almost always takes out the same amount: $150. If the ATM machine was adhering to this design principle of giving the user only meaningful choices, after a few uses of the machine, Terry would put in his card and the program would respond *in English:*

> "Hello Terry Hackett. Would you like to take $150 out of your checking account?"

It would completely skip over three questions. The pacing of the experience would not be unnecessarily slowed by asking three questions for which the program can assume the answers successfully.

Now, is it possible that Terry might want to take out $200 one day? Is it possible that he might want to use the ATM machine to brush up on his Spanish? Sure. So when Terry is asked whether he wants to take out $150, he would respond "no." At that point, the ATM would back-up and ask all the original questions.

In this case, it would actually be asking one more question than otherwise. If, however, this is bound to occur one time out of ten, then it might well be worth the trade-off. This becomes a subjective judgment on the part of the designer of the program.

Let's take it a step further. What if Terry occasionally used the ATM to deposit various amounts of money into his savings account? Let's say he also regularly transferred either $500 or $1000 out of his savings into his checking account. The ATM could note that Terry does one of these four transactions more often than others. The ATM could then greet Terry by asking in English:

Hello Terry Hackett. Would you like to:

- withdraw $150 out of your checking account
- deposit money into your savings account
- transfer $500 from your savings to your checking
- transfer $1000 from your savings to your checking
- or perform a different transaction?

Now, *most of the time,* the transaction that Terry wants to perform is right there up front. If Terry's transaction patterns change over time, the program can update itself. Redundant tasks are likely to be skipped so they don't unnecessarily slow down the pacing of the program.

Use User History Files to Make Intelligent Decisions

As you can tell, maintaining a record of the user's history of using the program is critical to the application of this principle.

In YDKJ, the "Press spacebar to skip instructions" option is the one annoying break with *The Jack Principles* in the game (this principle in particular). The program could eliminate this option all together and always make the decision as to whether to play or skip the instructions. To do this, the program would need to maintain a history file that allowed it to know if a user had ever played the game before. If according to the history file any of the contestants were new, the program would automatically make the decision to review the instructions. If the contestants had played before, it would skip the instructions automatically. In short, through simple reasoning and perhaps some user testing, we could get rid of that choice by making a decision in behalf of the users that would be accurate the vast majority of the time.

The maintenance of this sort of history file causes other interface complications, however. When first signing in, how does the program tell two people apart with the same name? Do you add an additional task to Cookie's interview to find out if anybody is new? (thus slowing the pacing into the game). Despite these complications, history files will be essential to jellyvision designers who need to make intelligent decisions in the user’s behalf.

Use “Artificial Intelligence” to Make Intelligent Decisions

What is sometimes falsely called “artificial intelligence” can also be of great benefit in customizing the program to the user automatically without interrupting the pacing.

Let’s go back to our jellyvision book review program. Let’s say every week that the program tells you what new books are out. If there are forty new books that come out every week, you may not have time to listen to all forty reviews. One thing the program can do is interview you before each program to get a sense of what to show you. The program could also have done an interview when you first started using the program and maintained that in a history file. The program can also do away with both of those interviews and simply “observe and remember” what books you choose to look at or ignore and what books you select to purchase. Then each week, based on a set of rules (known as an algorithm), the program will intelligently and automatically prioritize the order in which it shows and reviews the new books. It will do this based on your history of likes and dislikes. A good algorithm will constantly refine its accuracy as the user’s interests evolve. An even better algorithm will combine what it “understands” about the individual user and what it understands about other users who have an extremely similar history to that person.

The danger here is that a program can also make bad decisions. It can frustrate the user, because the user wants to see one thing and the program assumes that he doesn’t. There is a fuzzy psychological phenomenon that enters here.

The Psychology of Proacting vs. Reacting

With most computer software, users are *proacting*; the program doesn’t do anything until the user tells it to act. In this context, users expect a great deal of control. After all, from a programming standpoint, most multimedia programs and the Web itself are fundamentally tools—navigational tools. You control a tool; the tool does not control you. Television is the complete opposite. You don’t have or expect any control at all. Jellyvision lives in the gray area in between television and computers. In jellyvision programs, control is essentially shared between the user and the program. The user is *reacting* to the program. Psychologically, this may give a jellyvision program a greater degree of latitude to act on certain assumptions about the user without being questioned. For example, a user might react totally positively to a human character in a jellyvision program that said:

> “Hey, I’ve got a bunch of new books, but there are four in particular I think are just for you.”

With that, the character tells the user a little about all four books, asking if the user is interested in going more in depth after each book review. In contrast, the same user might be frustrated if she was forced to click through four book reviews selected by a web site instead of immediately having the control to narrow the search herself by clicking off a bunch of checkboxes.

There is no definitive answer on how a user will respond to having less than complete control in any one situation. It will depend primarily on the ability of the designer to (a) give the user the most logical and meaningful choices at any one moment of interaction, and (b) to craft the overall context leading into that moment so that the user wouldn't imagine wanting to do anything else anyway.

Eliminate Unnecessary Choices from the Basic Design

So far, this section has discussed how the program can make decisions for the user by watching what the user typically does and accordingly omitting unnecessary tasks. On the other hand, a good jellyvision designer will omit all unnecessary and meaningless tasks from the basic design anyway.

In YDKJ, questions are worth between $1000 and $6000 depending on the round. The values correspond to difficulty levels. During the original design of YDKJ, we considered letting players select the value level for each question. This would have added another task throughout the program that would have slowed the pacing. We decided that this choice was unnecessary; the randomness of the value levels was simply to be part of the game.

For other forms of software, the more control you give the better. This is *not* the case with jellyvision: just because the program *can* give the user more control doesn't mean it *should.*

This fundamental design break with most other forms of software is most evident with "setting preferences."

Don't Make the User Manually Set Preferences

Almost all software programs have a screen or set of screens that allow you to set preferences. You can adjust certain settings, activate or deactivate certain features, rearrange things, etc. Preferences are user adjustable settings applied throughout a program.

In most software programs, the more preferences you can adjust the better. It gives you more control. On the other hand, it also makes the program more complicated to use. For jellyvision, simplicity is at a premium. The more preferences the user has to manually adjust, the worse. Take note: television sets have different settings, but television *shows* do not.

In order to effectively interact with the program, if the user has to stop and adjust preferences, then you can kiss pacing good-bye.

Set the Preference Without Asking the User

Unlike other forms of interactivity, simplicity is more important than control. Jellyvision programs are not tools. Control is *shared* between the user and the program. Through good intuition, user testing or history files, a good creative team can often choose preferences in the user's behalf.

We also wrestled with the issue of difficulty levels in *That's a Fact, Jack* (TFJ). Given that TFJ is an educational program, the notion of difficulty could not ignored. It would've easily made sense for the teacher or the student to be able to set an overall preference for the difficulty level of a game, depending on the student's ability.

In fact, the program does not give this choice. Rather, it dynamically adjusts the difficulty level of the questions in real-time based on how well the student is doing. If the student gets a question correct, the difficulty goes up. If the student gets a question wrong, the difficulty goes down or remains the same. Neither the teacher nor the student is ever asked how difficult they want the questions to be, the program intelligently "sets that preference" for them.

———

The only questions a program should be asking are ones that *only* the user can answer. Every task and every choice should be meaningful. Designers should seek ways to craft the program so that it doesn't need to offer choices that are anything other than at the heart of the interactive experience.

In watching people play YDKJ, to our amazement, many don't even want to take the time to type in their name. They'll type things like "ddd" or just anything to get to the game as quickly as possible.

The nice side effect to following this principle is that the program is less likely to waste the user's time asking questions that the user feels are unimportant or annoying.

Maintaining Pacing

Make Sure the User Knows What to Do at Every Moment

If at any moment of interaction in a program, the user does not know what she is supposed to do, pacing will be compromised. Moreover, the user will probably also be frustrated.

Television shows are very "user-friendly." You don't need a manual to watch *Rosie*. You don't need a hint book to watch *Nightline* . You never have to call an 800 tech support number to get through an episode of *Seinfeld*. Television's "ease of use" factor is pretty high since changing channels and staring are the only required activities—and staring comes pretty naturally to most of us.

Anything interactive requires some level of instruction since people are actually participating. How an interactive program offers instruction, therefore, is critical.

Clearly, if the user ever has to pause to look up the instructions in a book, you can forget about maintaining pacing.

The program itself has to give instructions to the user in real time. Fortunately, since a key feature of jellyvision is creating the illusion that a character in the program is talking to you, it is well adapted to personalized instruction.

Give Instructions when they are Needed: Just-in-Time Instructions

The further away instructions are given from the time they are to be used, the more likely the user will forget them. Give instructions just before they are to be used: "Just-in-Time Instructions." In YDKJ, for Gibberish™, Disordat™ and JackAttack questions, the program waits to give instructions until that type of question is about to be played.

In doing user testing on YDKJ, we noticed that people regularly forgot to "buzz-in" before hitting the number of the correct answer. They'd keep pressing the number key, while the program was waiting for a Q, B or P key to be pressed to indicate who was answering. So we had the host give instructions to the user right at that moment:

"Gotta Buzz-In First!"

If the user still keeps pressing the number of the answer, the host blurts out again:

> "Hey, I said BUZZ IN!"

The instructions are given right at the moment they are needed. The user is told what to do and can immediately respond. Pacing is maintained.

Limit "Universal Instructions "

A program can't always give Just-in-Time instructions. A program may allow or require the user to perform certain interactions continuously throughout the program. In YDKJ, the use of the buzzers and the 'screw your neighbor' feature are examples of this. The program couldn't reasonably stop and give instructions at every moment where someone is supposed to buzz in or can use a screw. Such interactions require "universal instructions" given at an earlier point in the program. In YDKJ, universal instructions are given during the introduction. The fact still applies: the further away the universal instructions are from when they are going to be used, the less likely the user will remember them.

Give "Reminders" of Universal Instructions

Even if the program requires universal instructions, it can still "remind" users of them closer to the related moment of interaction. In YDKJ, reminding the users to buzz-in when they forget is one example of this. Here's another: during the break between rounds, if none of the players have used their screws, the host reminds them:

> "Hey, none of you used your screws! Next round I want to see you guys start screwing each other! Remember: Buzz-in and hit the 'S' key..."

We could have given the players a "reminder" about the screws even closer to the moment of interaction. After the host finishes reading a question, if nobody is buzzing in and time is running out, the host could yell:

> "C'MON!! Screw your neighbor!! Somebody buzz-in and hit the 'S' key. "

Even if you need to use universal instructions, give them as close to the first related moment of interaction as possible and remind users of them if it appears they've forgotten. This helps insure that people will know what to do at every moment in the program, which in turn, helps preserve the pacing.

Beware of Hot-Keys

Many software tools and PC games employ "hot-keys." Hot-keys are key combinations that a user can type at any point to quickly give the program a command. Hot-keys typically substitute for pulling down a menu or some other more cumbersome procedure. Hot-keys are a god-send to people who have to call the same command over and over again. In many PC games, typing various key combinations is in fact the way you play the game.

The problem with hot-keys and intricate key combinations is that you have to memorize them. You learn them from universal instructions. The 'S' button in YDKJ is a hot-key for screwing your neighbor. You need to memorize it. This means that you can accidentally forget it and thus find yourself not knowing what to do. Keep the number of hot keys in a jellyvision program to an absolute minimum. One hot key is already too many. The program needs to do extra work to remind users of hot keys throughout the program, as with the 'S' key in YDKJ. Two or three hot-keys could be unwieldy and confusing.

Put All the Currently Available Choices on the Screen

Putting a hot-key on the screen when it is available solves that problem. In YDKJ, unlike the screws, the screen graphics actually show the buzzer keys, 'Q', 'B' and 'P'. If a player forgets, the choice is right there in front of his face.

In general, if the user has a choice of some type of interaction, make sure the choice is actually on the screen. If there are too many choices to put up on the screen at any one moment, then you probably need to rethink your design and take a look at Jack Principle #2: "Limit the number of choices the user has at any one time."

Set the Preference By Asking the User in the Flow of the Program

In the previous section, there was a discussion of setting preferences without asking the user. If, however, the program cannot successfully select a preference in the user's behalf, then it should ask the user to set the preference in the flow and context of the program itself, rather than going to a separate window. There are two reasons for this: (i) If the user has to stop and go the preferences screen, it will slow down the pacing of the program. (ii) the user may not even know that she has to go to some preferences screen to make a certain adjustment. She'll waste time trying to figure out what to do and pacing will be lost.

Our example book review program could have a preferences window where you could specify that you only want to see books that are available in paperback. In a jellyvision program, however, the host of the program would ask you about this the first time you used the program.

> "So, before we begin, let me ask you a few questions to get a sense of how I can best help you. First of all, as you know, hard bound books are more expensive than paperbacks. Now, most of

> the time, the hard bound books come out first. If you'd prefer, however, I can make a point of just reviewing books that are already in paperback. Would you like me to do that?"

You respond "Yes" or "No" and the preference gets set, but the flow and pacing of the program is still preserved.

What happens if the user, who had specified paperback only, later changes her mind and now is ready to buy hard cover books as well?

Let's say the user is looking for a book on a particular topic and among the relevant books is one only available in hard cover. After reviewing the books in paperback, the host can mention:

> "You should know that there's an excellent book on this subject that just came out. But it's only available in hard cover, so it is a bit more pricey. Do you want to hear about it?"

If the user responds that she does and then buys the book, the host could then ask:

> "So is it all right if I start showing you books that are still only available in hard cover?"

Thus, in the flow of the program, the user has another opportunity to reset the preference.

Now, if the user didn't even want to look at the hard cover book—and makes this same choice in several similar situations—perhaps the program adjusts itself so that it doesn't even mention hard cover-only books for several months. Then it again gives the user another chance to change her mind. On the other hand, if the user wants to hear about the book, but doesn't buy it, perhaps the program continues to just mention hard covers after reviewing the paperbacks first.

Is this solution far more complicated for a designer than a preferences window? Absolutely. However, it is far *less* complicated for the user, who doesn't have to remember anything. She only has to answer a simple question asked at an appropriate moment. The user always knows what to do, and pacing is preserved.

Place All Manually Adjusted Preference Controls on the Pause Screen

If the program cannot set the preference without asking the user, and cannot ask the user in the flow of the program to set the preference, then the preference must be set in a separate window. Many programs have multiple windows for preferences. In a jellyvision program, where the "manual adjustment" of preferences should be minimal, there should be only one place where preferences are set: a screen that pops up when the user puts the program into pause.

The concept of “pausing” is universal to any non-broadcast, time-based media. For videotapes, CD-players, and most video / PC games, there is almost always some simple button to press to pause the program. On the other hand, newspapers, magazines, websites and most multimedia programs don’t move through time, so the concept of pausing is irrelevant. Jellyvision, being a time-based medium requires a pause function. The input device on a jellyvision program could be virtually anything, but somewhere, somehow, there needs to be a way to pause the program.

Since pausing is a universal concept for all jellyvision programs, it makes sense to put all manually adjusted preferences on to a screen that appears when the program is put into pause.

In YDKJ, the pause screen allows one to adjust the volume, quit or restart the game. Those are the only manually adjusted commands. If there were others, however, they would be located on that screen. There are four reasons for putting all manually adjusted preferences on to the pause screen:

(1) It makes it easy to find the preferences because even if you don’t know that they are on the pause screen, you’re bound to pause the program at some point and stumble upon them anyway

(2) It makes it easy to remember where the preferences are

(3) Getting to the pause screen is always a quick, single button push away

(4) When you’re not sure what to do, your first reaction will be to pause the program. The pause screen is therefore the best place to locate any additional controls that might help them.

———

A jellyvision program should be certain that the user knows what to do at every moment of interaction within the program. It can do this by giving instructions as they are needed, periodically reminding users of instructions if their actions suggest they’ve forgotten, limiting the number of hot keys that the user can forget, setting any preferences in the flow of the program, and placing all “manually adjusted” preferences on the pause screen, one universal place where they will be easy to find.

Maintaining Pacing

Focus the User's Attention on the Task at Hand

When the program asks users to interact, focus their attention on the information that is relevant to their decision. If the user is distracted looking or hearing information that is peripheral to the decision they must make, it will slow down their response and thus the pacing of the program.

Multimedia programs and web pages were created to be viewed on a computer screen where the user is typically two to three feet away. These forms of interactivity were also born out of the software industry.   The software industry was founded on producing tools where control and thus, lots of features, are the benchmark for value. The combination of the user's proximity to the screen and the value placed on quantity of features means that it often makes sense for multimedia programs and websites to have screen lay-outs with many intricate visual elements. Some elements have functionality, some not. When a user is sitting two feet away and pacing isn't of the essence, then it can be a wonderful experience looking around from one side of the screen to the other at the different illustrations, text, banners, buttons and such. It's akin to the way we look at newspapers.

Television programs are the polar opposite. One sits eight to ten feet away. There is almost never a shot on television where the viewer does not instantly know where to look. The whole scene is taken in one glance. One exception to this would be a football game, where multiple and relevant things do happen in different parts of the screen at the same time. Still, the viewer naturally knows to keep his eye on the ball. If an illegal block happens down on the opposite side of the screen, the viewer will be shown an instant replay visually isolating that event. On television the images and sounds are constantly changing, but typically television only shows one event at a time.

Jellyvision programs need to be closer to television programs: one thing at a time. In general, when a program has pacing, there is often no time to look around at various small details. This is particularly true at the moment when the user must interact.

Focus with Visuals

YDKJ focuses the user's attention by keeping the screen as uncluttered as possible. There are many other techniques to focus the user's attention on the information relevant to their decision. If the information is visual, the kinds of things that can set it off include its color, shading, dimensionality and relative size compared to other parts of the screen; it can also be animated or more animated than other objects. For example, in YDKJ, when a person buzzes in, if they have a screw left, the screw starts to animate even faster than before. The goal is to temporarily grab the user's gaze to remind them that they do have a screw they can use right at that moment.

Focus with Sound

Sound can also be used to focus the user's attention. YDKJ does this continually by having the host read off the questions and multiple choice answers that appear on screen. The host's words help direct you to the part of the screen on which you should be focused.

There is a non-interactive moment in YDKJ, just before the question appears on the screen. It's essentially the prologue to the question. During this prologue, the host will occasionally say something that does not appear on the screen in text, but the user must hear it in order to fully understand the question. For example:

> —Before the question appears, the host says:
>
> "O.K. now listen up! You're in Springfield driving your 18-wheeler down I-88 to Sioux City. It's 3:20 in the afternoon and rainy. Your nickname is The Brown Banana."
>
> —Then the Question appears on screen:
>
> "What's your 10-20?"

As the host starts reading the dialogue before the question, all movement on the screen stops. There are few things on the screen to distract the user visually. Further the host even says: "O.K. now listen up!" The program thereby *focuses the users on listening* at that moment.

———

A jellyvision program should carefully use sound and picture to focus the user's attention on the information she needs in order to effectively and efficiently interact. Thereby the program helps keep the user in sync with the pacing of the program.

Maintaining Pacing

Use the Most Efficient Manner of User Input

Choose a method of input that allows users to indicate their choices most efficiently.

There are quite a few ways for a user to input information into a computer-driven device. Mouse, keyboard, remote control, joysticks and other game controllers are used widely. A microphone (with or without voice-recognition software), digital stylus (with or without handwriting recognition software), electronic gloves, headgear, and foot pedals are also used. A program's input requirements can get exotic. If the goal is to create a jellyvision program that will reach a mass audience, relying on a specialized input device is probably not a great idea.

Wisely Choose a Method of Input within Existing Limitations

The requirements of the program itself, in fact, may be less relevant than the hardware that is available, the physical environment in which the program will probably be experienced, and the number of people who will be using it simultaneously.

In building YDKJ, we knew that the first platform we were developing the game for was CD-ROM. We only had three real choices to allow people to make their selections: (1) have users press numbers on the keyboard, (2) use the mouse, or (3) use the arrow keys to highlight the selection and hit return to input it. As soon as you consider that three people would be huddled around the keyboard, requiring everyone to reach over each other for the mouse was clearly inefficient. Using the arrow keys to highlight a selection and hit the return key might take multiple key presses. Just hitting a number was easily the fastest. For answering Fill-in-the-Blank and Gibberish questions, well, there was only one option: people would have to type in their answer and hit return. So the number of people who might be playing (up to three), the hardware it would run on (a computer with a keyboard and mouse), and the environment in which it would be used (typically on a desk in an office or den), really limited our choices. However, we made the best choices we could for speed of input under those circumstances.

Nonetheless, this keyboard method of input has three major drawbacks. First, it forces three people to huddle together around a single keyboard (although if you really like your opponent that could have other advantages). Second, typing out answers is slow no matter how fast one types. Third, it forces the user to regularly take their eyes off the screen to look down at the input device itself. An input device that allows you to keep your eyes on the screen is usually going to be more efficient and help maintain pacing.

The "Ideal" Method of Input

Far and away, the perfect input system for YDKJ would be one to three handheld remote control units, each with a button for a buzzer and attached microphone. The users could then sit back comfortably on a couch and not be crowded around a keyboard. Instead of punching numbers to make their selections, they could just say the number aloud into their microphone. For Fill-in-the-Blank and Gibberish questions, they would just say the answer. Depending on who buzzed in, the program would only "listen" to the response from that user's remote microphone. The pacing of both games would be even better than it is today, because the user would be able to indicate their decisions more quickly and more elegantly.

Moreover a voice activated interface, where people "talk" back to the screen and the characters on the screen respond intelligently, truly fulfills the metaphor of jellyvision: the user is *talking with the characters in the program.*

Besides, it's always funny to watch people talk to their television screen.

Hopefully, this sort of input hardware will be widely available and standardized when jellyvision programs can widely be used in the same room and on the same screen in which we watch television. At that point, we will retrofit YDKJ to take advantage of the technology. The programs themselves do not change, but developments in hardware allow us to seamlessly update the method of input, make it more efficient, and improve the user's contribution to the pacing of the program.

The options for how a user can physically input information to the program is normally limited by the hardware that is available, the physical environment in which the program will be experienced and the number of people who will be using it. Within those limitations, however, a program should utilize a method of input that allows the user to respond the most quickly. This facilitates maintaining the pacing of the program.

Maintaining Pacing

Make the User Aware that the Program is Waiting

When the user isn't responding quickly enough, *prod* the user into action with a sound, visual effect or dialogue or otherwise make the user aware that the program is actively expecting a response.

"Prodding" is used throughout TFJ and YDKJ. In TFJ, our educational program, the dialogue prods are a bit more gentle than in YDKJ, where they are nothing less than caustic. In YDKJ, when Cookie the stage manager asks for your name, if you don't type something fast enough, he might say:

> "Hey, Player 2! Wake the hell up and give me your name!"

In that both programs are game shows, we also use "countdown" music during questions to remind users that time is passing.

On Maintaining a Slower Pace

Maintaining pacing does not mean maintaining fast pacing. The pacing of a jellyvision program can be fast like YDKJ or calm and contemplative. It's up to the design team.

A program may very well allow for extremely long periods of input. For example, if our fictional book review program said:

> "So, after reading this book, you can't really recommend it, huh? Tell me why. If it's O.K., I'd like to share your thoughts with other folks who are interested. Take your time, let me know when you're done."

The program opens up a large text field for typing, plays a little undistracting instrumental music and doesn't interrupt at all, as long as the user doesn't completely stop typing for more than a minute or two. If that were to happen, the host might calmly remind the user:

> "Click 'I'm done' whenever you're finished, O.K.?"

Even if the pacing is leisurely, a jellyvision program does not act like a word processor. A word processor doesn't care if you suddenly stop in the middle of a sentence to run out to the store to pick up some talcum powder. It'll just patiently wait for you. It'll wait forever. It doesn't care if you ever come back. A jellyvision program cares. If you stop interacting for some period of time, it will let you know that it's waiting for you to finish.

To keep the pace of a jellyvision program, the program must keep the user interacting within a time frame. The program therefore needs to let the user know that it is expecting her response in a timely manner.

Maintaining Pacing

Pause, Quit or Move On Without the User's Response If It Doesn't Come Fast Enough

If a user's response doesn't come soon enough, the program should be able to continue without any user input and still make sense, elegantly quit or put itself into pause.

Ultimately the only way to truly insure pacing is to *force* the user to participate within a time frame. If there is no timely response, the program can't just do nothing—it has to respond in order to "train" users that they *must* respond in time.

Moving on without the User's Response

In YDKJ, if there is no response to Cookie's first request to indicate how many people are playing, Cookie will prod them for an answer. If there is still no response, he tells the control room:

> "Hey Helen, I don't think anybody's out there. I'm just gonna grab people from the audience."

At that point, the game starts, and three random names (actually picked from the credits) appear as players on the screen.

Another example: If a player doesn't pick a category in time, the host will say something like:

> "This isn't rocket science! How 'bout this one?"

The program then selects a category automatically. YDKJ can continue playing entirely by itself even if no one is sitting in front of the screen. Still, it is careful to use dialogue within the tone and context of the program that deals with the fact that there is no user input. It is also notable, that after the program has been running along without user input, if a user shows up, she can jump right in and start playing instantly.

Pausing or Quitting

For some jellyvision programs, it will not make sense for the program to just play by itself. Let's say in our book review program that the person wants to get a book of Impressionistic prints as a gift. The host might ask:

> "This type of book can get sort of pricey. How much are you willing to spend?"

If the user never responds after a couple prods, it doesn't make sense for the host to say:

> "You don't want to tell me? O.K. fine...then I say you're willing to spend $200."

In this case, the program should put itself into pause or quit out entirely. A good jellyvision program will stay in character while doing this. For example, the host of the program might explain that the program is quitting before actually doing it:

> "Ummm...last call here....anybody home? ...All right, well, I guess we'll just pick up where we left off later..."

With that, the program quits. The program might simply put up a title screen and go into pause, almost like what television stations do when they are having technical difficulties.

No matter what type of jellyvision program it is, it must respond definitively if the user does nothing. Otherwise, users will recognize this loophole. In turn, the ways that the program psychologically draws people into the pacing of the experience, through the above principles, will be compromised.

Final Thoughts About
Maintaining Pacing

The requirement to maintain pacing in turn requires jellyvision programs to make two huge breaks with other forms of software--websites and multimedia programs in particular.

The first is that jellyvision programs require the user to act within a given time frame.

The second requires interactive designers to drop the common assumption that the more features and options you give the user, the better the program. To be sure, this is often a good assumption for tools applications, multimedia programs, web pages, and video/PC games.

But jellyvision programs are different. Here is the ironic secret to this new form of interactivity: *Less is More.* Give the user fewer choices. Give them less control. Keep the graphics sparse. Do one thing at a time.

The Jack Principles

Creating the Illusion of Awareness

Maintaining pacing is one of the key factors that makes a jellyvision program television-like. Personalizing the program so that it individually responds to the user or users is what makes a jellyvision program...interactive. Ultimately, the distinguishing power of a jellyvision program is its ability to create the illusion that the character in the program is aware of the person sitting in front of the screen. In order for a program to simulate the way a human would respond—to make it seem like the program is truly aware of its audience—its responses must betray real human intelligence and emotions.

Currently, there is no reliable way to do this with what has been called "artificial intelligence." There are no algorithms that can evaluate the input of a human being and respond intelligently with emotions that would fool anyone into thinking that the software program is "aware" like a human. There is no Hal.

It's hard to predict when and if such technology will ever come into being. Nonetheless, it is safe to say that it will take a mammoth leap of human imagination to even begin to approach the problem of faithfully modeling the non-logical aspects of the human brain, let alone the human spirit.

So, for the time being, to create the illusion of human awareness, we must use actual human beings to do it. Such human beings are called writers and actors. It will be up to writers to apply their craft, answering at every moment of user interaction in a program, "how would my character respond to that?" Then the writer must script it out for an actor to perform. Just like television.

A jellyvision program can *create* the illusion of awareness by utilizing any combination or all of the following principles:

Jack Principles to Create the Illusion of Awareness

Specifically Respond with Human Intelligence and Emotion to:

1. **The user's actions**
2. **The user's inactions**
3. **The user's past actions**
4. **A series of the user's actions**
5. **The actual time and space that the user is in**
6. **The comparison of different users' situations and actions**

The following pages examine each of these principles in detail.

Creating the Illusion of Awareness

Specifically respond with human intelligence and emotion to:

The User's Actions

The most obvious way to indicate awareness is to have the program immediately respond in a human-like way to each instance of a user's input. The response must betray a clear recognition of what the user has done. This is typically accomplished through a character's dialogue.

In YDKJ, when you select the wrong answer, the host will sarcastically let you know. Let's say one of the answers to a question was "Willie Shoemaker" and the user selected it. The host might say something like this:

> "Willie Shoemaker? 1985 power forward for the Lakers? Since he was a jockey, I think Willie would've had a tough time driving the lane for a dunk..."

It is this immediate and human-like response to the user's actions that for most people is the defining characteristic of the Jack experience. It is the first thing people notice about YDKJ. Almost invariably, new users of the program remark: "It's like the guy is *really* talking to you." Indeed, this may prove to be the most blatant differentiating characteristic of jellyvision from all other forms of mass communication.

The telephone enables "one-on-one" communication. Radio, newspapers, television, magazines, and theater enable "one-to-many" communication. Jellyvision enables a new form of communication: "one-to-many, one-on-one." The creator of a jellyvision program can *speak and listen and respond* to many people at the same time. The potential power of this mass medium is great indeed.

Showing Awareness of Freeform Response

There is a deeper level of awareness that a program can imply by responding intelligently to a user's freeform response.

Let's say the above question about the Lakers' power forward had been Fill-in-the-Blank. Well, it is highly unlikely that a user who was actually trying to get the answer right would've buzzed in and typed out "Willie Shoemaker." If he did, he would've heard a generic response simply telling him that he was wrong: "Uh...sorry, that's not gonna do it." However, he might type in "Kareem

Abdul Jabbar" since many people know Jabbar played for the Lakers. The writers of YDKJ could, through their own knowledge and intuition or user testing, anticipate that many players would type in Jabbar. Thus, they might script and record a specific piece of dialogue to accommodate that:

> "Jabbar? Played for the Lakers. Wore funny glasses. But he was the center, not the forward."

This creates the illusion of an even more profound level of awareness when the program can respond intelligently to a user's freeform input.

Clearly, without "artificial intelligence" there is just no way to specifically respond to every possible freeform input from a user. So, success in employing this second level of awareness is fundamentally tied to the anticipation and calculation of what a user might *likely* input. Actual user testing can be of significant value here. Even better, in an on-line situation, a program can track the freeform input of many users over time and flag writers to add additional responses based on that information.

Generic Dialogue vs. Custom Dialogue

Regardless of whether the question is multiple choice or freeform, responses to it that are "generic" (that do not respond with specificity), tend to diminish the illusion of awareness:

> "Umm....sorry that is totally wrong."

"Custom" responses tend to enhance the illusion.

> "Michael Jordan? Michael Jordan playing for the Lakers? Maybe you're new to planet Earth..."

This is not to say that generic dialogue should not be used. YDKJ has a great deal of generic dialogue built into the program. It is used for everything from welcoming the players to the game to responding to wrong answers where we couldn't think of anything funny to say. Part of the art of creating a jellyvision program is balancing when and how you use "generic dialogue" versus "custom dialogue." A program can create the illusion that the character is truly aware of the user's actions by either being very specific or in saying something that isn't so specific but is nonetheless appropriate. In a program that does this well, the user won't much notice the difference.

———

The most typical and obvious way to give the user a sense that the program is "listening," is to respond to a user's input with dialogue that is specific to what the user has done and "feels" human.

Specifically respond with human intelligence and emotion to:

The User's Inactions

The principle for maintaining pacing of telling the user that the program is waiting suggests another important way to create the illusion of awareness: let the user know that program knows that she isn't doing anything.

When you ask someone a question, if she just sits there for ten seconds saying nothing, you're probably going to *do* something. At the very least, you'll repeat the question or ask the person if she heard you. A jellyvision program, modeling the behavior of a human being, also reacts when it receives no response. This reinforces the illusion that the character in the program is aware.

For first time players of YDKJ, the moment when they first recognize that the program is "talking to them" often occurs if they fail to type in their name. Cookie, the stage manager, reacts to the player's lack of action:

> "Hey Player One! Wake up! I said type in your name! C'mon!"

Reacting with human intelligence and emotion to the user's inactions does not necessarily mean the character has to be a jerk:

> "Forgive me, but I was asking if you could type in your name for me..."

The character can have any type of personality the writers and the actor invent, but every character should react somehow if the user fails to interact. That's what a human being would do—and that is the illusion the program should be trying to create.

———

When a user fails to act, a jellyvision program can create the illusion that its characters are aware by having them specifically respond to the user's lack of action in a way that "feels" human.

Specifically respond with human intelligence and emotion to:

The User's Past Actions

A program can create an even higher level of perceived awareness by creating the illusion that the characters in a program "remember." It is one thing to respond intelligently to what a user does. It is even more human to "remember" what a user did and allude to it intelligently in the context of what is happening in the present.

In YDKJ, for example, imagine a player incorrectly chooses the multiple-choice answer "Twinkies™" when asked which Hostess snack product looks like a hockey puck. (The correct answer is: "Ding Dongs™"). Later in the same game, there might be another food question about the shape of pasta. If the *same player* gets that question wrong, the host might say:

> "No, if your covered wagon's wheels were shaped like linguine, your horses would be pretty damned tired."
>
> —and then the host adds:
>
> "...of course, they'd be even more tired if the wheels were shaped like Twinkies instead of Ding Dongs. Yeah, that'd be really stupid."

Having the character make a sarcastic comment specifically alluding to an event triggered by the user ten minutes earlier powerfully creates the illusion of awareness. The character, like a human being, appears to remember.

This effect can be more powerful still if the program "remembers" events that happened during interactions with previous programs that took place days, weeks or perhaps years ago.

Remembering Freeform Input

Taking it one step even further would involve "remembering" a user's *freeform* input. Imagine in January, the host of the book review program had asked the user who her favorite character had been from the books she had purchased to date. This program has voice-recognition. The user simply says into a microphone: "Holden Caulfield." The program has a database of all the main characters from all the books the user has read and plays a media file of the host responding intelligently:

> "Holden Caulfield from *The Catcher in the Rye.* Very cool character indeed."

The program notes the user's response about Holden Caulfield in a history file. Four months later, in April, the host is reviewing a new book:

> "...And there's another reason why I think you'll like *The Giver*...
> "...The lead character, Jonas—the dilemmas he deals with kind of reminds me of Holden Caulfield."

The host appears to be remembering that the user liked Holden Caulfield, something the user had just spoken aloud out of her head four months earlier.

The organization, writing and production that are involved with seamlessly executing upon this principle can be quite complex, but depending on the circumstances, the illusion of awareness it creates could make the program infinitely more effective and be worth the effort.

Remembering Too Much

In the fourth volume of You Don't Know Jack: The Ride, we implemented awareness of past actions. As described above, the host would allude to answers the player got wrong or right in the past. We came upon a funny problem in testing, however. Sometimes the players didn't get the allusions, because they didn't remember what they had done – even though the program did. In some cases, we rewrote the dialogue so that the host's response to the first action was bigger and more memorable and the latter allusion to it more blatent. In other cases, where the being subtle was what made the joke work, we left it in (and hoped for the best) or just dropped it. The lesson: even though a computer program can remember everything, doesn't mean there's always value in it.

A human being can remember what another person said in the past and allude to it intelligently in the course of conversation. A character in a jellyvision program can appear to do the same.

Specifically respond with human intelligence and emotion to:

A Series of the User's Actions

One of the core principles for maintaining pacing is to give the user only one task to do at a time. Nonetheless, there are circumstances where it will not be appropriate for a character to respond immediately after every single interaction. In such situations, it makes more sense for the character to react after a series of the user's responses.

In YDKJ, in the final question, the JackAttack, the host says something like "Good luck..." just before the actual question begins. He is seemingly on the sidelines watching. Up to three players buzz in whenever they see a match on the screen. If one of the players is getting practically everything wrong and then gets the very last match correct, the host will come back on and yell:

> "Player 2 you were horrible until that last one! Let's see if it ruined your score!"

If Player 3 simply never buzzed in at all during the whole JackAttack, the host will yell:

> "Player 3, what the hell happened to you?! Did your fingers atrophy? Let's see the final scores!"

If a player is playing alone and he gets about as many right as he gets wrong (making the whole question a wash), the host will say:

> "What a total waste of time! You were completely mediocre! Let's see your final score!"

In order to create the illusion of awareness, the program tracks the players' performance and then feeds that information into an algorithm, which in turn calculates which of the host's audio files should play. Again, a designer needs to account for every possible combination of user inputs and make sure there is at least one moment of dialogue that intelligently responds to each circumstance.

Sequencing a Series of Files to Respond to a Series of Actions

In our book review program, imagine you are a first time user. The program wants to get a "taste" of what subjects you find interesting , so it can make better recommendations. The host reads off a whole series of various subjects:

> "War...Parenting...Sports...Archeology...Religion...Space Flight...Modern Art ...Cooking...Love Stories..."

After each subject, you hit one of two keys to indicate if you are "Very interested" or "Less interested." The host gives no specific (custom) response after each answer; he only says things like "O.K...Uh-huh...Got it..." and goes to the next subject. By the end of the sequence, however, the series of your choices is run through an algorithm. The algorithm then selects and plays a sequence of audio files that creates the illusion that the host is aware of your choices:

> "All right, so you've got a pretty diverse set of interests...
> "...but you're seriously into the science stuff...
> "...and you're a sucker for a good love story....
> "...That's great. I've got a lot of ideas for books you might like."

All four of these phrases would be in separate audio files. They would, however, be strung one after another into what should appear to the user as one seamless response.

The host's first line about having a diverse set of interests might have come from the program's algorithm identifying that (a) you indicated "very interested" more than, say, ten times and (b) that the ten specific subject areas were in both the sciences and the arts.

The host's second line about being particularly interested in science might have come from the algorithm identifying that, for example, over 70% of the subject areas in which you were interested were science-related.

The host's third line about being a sucker for a love story came from the algorithm identifying one particular subject area ("Love Stories") that was in the other 30%.

If you had chosen "Parenting" (which would have been a subject in the 30% that was not science-related) the program's algorithm could have selected a piece of dialogue related to that instead of "Love Stories." In algorithms, there will be times when several pieces of dialogue *could* be used. In this case, the rules should prioritize the character's dialogue by selecting the response to the user's input that is most noteworthy. The program selected to allude to "being a sucker for a good love story" because the designers of the program felt that it was more noteworthy than having the host say "...and you're interested in what it takes to raise kids."

Note how any one or more of these four phrases could be swapped out and replaced with another phrase while maintaining what can appear to be a seamless flow of dialogue.

> "All right, so you've got a pretty diverse set of interests...

"...but you're seriously into the science stuff...
"*...and you're interested in what it takes to raise kids....*
"...That's great. I've got a lot of ideas for books you might like."

The combination of these phrases into one response powerfully creates the illusion that the host is aware of the user. Part of the writer's and the actor's art for jellyvision is to write, perform and record many individual phrases in such a way that they all flow together seamlessly when "mixed and matched" with one another.

———

A human being not only has the ability to respond intelligently upon witnessing a single action, but can analyze and synthesize a series of actions and sum them up with a single response. A character in a jellyvision program can appear to do the same.

Specifically respond with human intelligence and emotion to:

The Actual Time and Space that the User is in

By being aware of the date, time and location of the user, jellyvision programs can use that information in characters' dialogue and thereby create the illusion of awareness.

In YDKJ *Volume 2*, the program looks at the internal clock of the computer before each game. Depending on the time and date, Cookie might say:

> "Hey there, Happy Easter. I'll tell ya, you know what puzzles me about this holiday—what the heck does the resurrection of the son of God have to do with little painted eggs? Total mystery to me. Anyway, how many players we got?"
>
> Helen: "Cookie, I need names. We've got contestants."
> Cookie: "Are you kidding me?! What kind of losers are up at this hour on Saturday night playing computer games? Give me a break."

Utilizing the Ability to Update Information Online

In an online situation, a jellyvision program that knows the city in which the user lives can also weave in awareness of the weather and events specific to the user's area.

> "All right let's hurry and find you a book to curl up with, because with this lousy weather, you're gonna be spending a lot of time inside."

These examples above are just fun, tangential uses of this principle. If, on the other hand, you are creating an online program to help someone select a social activity for the day, the ability of a character to show awareness of the time, the day of the week, whether or not it is a holiday and the weather could be quite meaningful:

> "How about going to the zoo?...
> "...They've got a new exhibit in the Lion House...
> "... It's the middle of the week so it won't be too crowded...
> "...and it's beautiful outside, so this would be a perfect day to go."

If the program were recommending the zoo on a day when it was pouring rain, the host would seem pretty stupid—and unaware. Instead, the program can check the weather in the user's area (in this case uncovering that it is a sunny day) and play a piece of dialogue like the one above to convey that. Note that this line of dialogue could be utilized in suggestions for hundreds of activities—anything that takes place outside.

The dialogue above also utilizes its awareness that it is the middle of the week in a way that can actually help the user make a decision. Note that this line of dialogue could also be swapped in for hundreds of activities—anything that tends to get crowded on the weekend.

Another way to show awareness of the time and space the user is in, is to be aware of actual events that are currently taking place. The reference above about the new exhibit at the Lion House is one example of this. The YDKJ net show (the online version of the game that gets updated twice a week), includes trivia questions based on current events. It regularly includes questions alluding to major sports events, movies that just came out, the results of elections, TV commercials that are currently running, holidays, etc.

A human being can make intelligent decisions and observations based on her awareness of the time as well as circumstances and events taking place in the physical environment. A character in a jellyvision program can appear to do the same.

Specifically respond with human intelligence and emotion to:

The Comparison of Different Users' Situations and Actions

Evaluating information about two users and comparing them is a strong way to intimate awareness.

In YDKJ, the host reviews the scores after every game. If one player beats the other two players by a landslide, a decent margin, or a thin margin, the host will allude to that specifically:

> "Well, that was a squeaker. But Player One, you edged out Player Two for the win. Nice job."

This principle can be extended well beyond the simple comparison of numbers.

Comparing the Past Actions of Two Users

Going back to our book review program, you remember that the user indicated back in January, that she was a fan of the character "Holden Caulfield." Imagine that in June, the host is about to show the user a book review typed up by another user:

> "Well, here's what one of our other readers had to say about *Shiloh...*"
>
> —and then add:
>
> "...and if it gives her any more credibility, you should know that she *also* loves Holden Caulfield. O.K., here's what she said..."

The program uses its history files of both users, compares the list of their favorite characters, notes that there is a match and plays the appropriate media file of the host. This forcefully creates the illusion of awareness.

Synthesizing Multiple Comparisons & Memory of Past Actions

It's possible to deepen the level of awareness even further. Imagine that the host has asked you to rate a book you previously purchased. From a multiple choice list, you choose to give it three out of five stars. The host responds:

> "You thought the book was just so-so?"
>
> —and then adds:
>
> "...interesting. You and your sister usually have similar tastes and she just loved it."

This simple piece of dialogue implies an awareness of:

(1) The user’s current input (i.e. you gave the book 3 stars out of 5)

(2) “Memory” of another user’s family relationship to the current user (i.e. information which you previously gave to the program. When you first signed up the host asked you for the names of any family or friends who also use the book review program)

(3) Another user’s past actions in comparison to the current user’s past actions (i.e. you and your sister usually give the same ratings for books)

(4) Another user’s past action in comparison to the current user’s current action (i.e. your sister gave the book 5 stars but you gave it 3).

The Cost and Complexity of Creating Deep Awareness

The application of this principle can create an illusion of an incredible depth of awareness. However, the complexity of actually doing the design, production and organization can be equally deep. The algorithm for selecting this piece of dialogue and the sheer volume of writing and recording necessary to achieve this effect may be daunting, unnecessary or cost-prohibitive. It is, however, theoretically possible if it makes sense for a particular program. To do this, a creative team would need the right set of production tools to assist in the design, scripting and organization of the media. Unfortunately, that set of tools does not yet exist.

Whenever you are doing jellyvision design, you need to consider the value of achieving a deep level of awareness in the context of the difficulty and cost of execution.

———

A human being can dynamically and simultaneously draw comparisons of actions from both the past and the present and sum up such comparisons in a single response. A character in a jellyvision program can appear to do the same.

Final Thoughts about Creating the Illusion of Awareness

Massive Media Creation

It is no doubt apparent that to create the illusion of awareness, writers and actors have to produce many, many separate pieces of media. In the first YDKJ compilation product, *You Don't Know Jack XL*, there are nearly 10,000 individual audio files, ranging in length from less than a second to more than a minute. This is for an entire CD-ROM product. A single 21 question game of the YDKJ net show involves approximately 300 - 350 individual audio files. This includes both custom audio files (specific to that show) and generic audio files (reusable for any show). The fictional book review program, expanded over time as it adds more books to its offerings, could easily involve hundreds of thousands of files. This sounds complicated, and it certainly is, but a professional development group *can* successfully establish a production and editorial process that allows for the efficient creation of this much media. Consider the amount of media created daily by a single news organization like CNN.

Awareness and the "Wow" Factor

There are many opportunities within a jellyvision program to create the illusion of awareness. Pick the moments to do this that are actually meaningful in the context of the program. In YDKJ *Volume 2*, the first time users hear Cookie wish them "Merry Christmas" on Christmas Day, they freak out. There is a big "wow!" factor when a computer program shows off how aware it can be—the experience is still novel. This phenomenon won't last forever. Ultimately, a program's awareness will not be impressive, it will be expected. At that point, the challenge becomes not simply creating the illusion of awareness, but how seamlessly and meaningfully a program does so. In fact, eventually, it may be a detriment to a program if the *way* that it is aware is too obvious or too solicitous. The art will be to create the illusion of awareness without the illusion drawing attention to itself.

"Big Brother" Awareness

There is yet one additional form of awareness, but it is *not* one of *The Jack Principles*: responding with awareness to outside information about the user which the program legally or ethically should not access.

Here was one idea that was suggested for the end of YDKJ.

> "Well, player one, you won it big! And given the fact that you have less than $200 bucks in your Quicken account, I'd say you could use the cash."

We probably would've had difficulty cracking the encryption on Quicken's database, but we never pursued the idea anyway. No question, this would have been very funny, but it would've upset a lot of people and could potentially have been embarrassing to them in front of their friends. Moreover, even if such a comment at the end of YDKJ was harmless, it is a slippery slope toward seeking and utilizing information in a way that can be truly damaging.

The reason this "unprincipled" form of awareness is even being noted, is because a designer of jellyvision programs will inevitably recognize this opportunity. So we wanted to warn you upfront: it's a very bad idea.

If jellyvision, as defined here, does turn out to be a form of mass communication, it is going to be dealing with privacy issues in unprecedented ways. There are no laws that can substitute for responsible behavior of those who create and control information. The potential power of this medium is intense. If you are taking advantage of this power, please take these issues seriously.

The Jack Principles

Maintaining the Illusion of Awareness

As a jellyvision program creates the illusion of human awareness, there are many moments in a program where that illusion can be inadvertently shattered.

In order to *maintain* the illusion of awareness, a designer should adhere to each of the following principles:

Jack Principles to Maintain the Illusion of Awareness

1. **Use dialogue that conveys a sense of intimacy**
2. **Make sure characters act appropriately while the user is interacting**
3. **Make sure dialogue never seems to repeat**
4. **Be aware of the number of simultaneous users**
5. **Be aware of the gender of the users**
6. **Make sure the performance of dialogue is seamless**
7. **Avoid the presence of characters when user input cannot be evaluated**

Maintaining the Illusion of Awareness

Use Dialogue that Conveys a Sense of Intimacy

Writers and actors must always remember they are speaking to only one or a handful of people at a time, not a crowd.

Here's an example of dialogue that does not follow this principle:

> "Hey you folks out there, welcome to our show. You know, if you're like most people, you probably like a good adventure flick now and again..."

Could you ever imagine speaking this way to three people standing right in front of you? Words like "folks out there" suggest a sense of distance, as if the performer were talking to a TV audience that he can't see and with whom he can't interact. This quote compounds that distance by assuming something about the user based on what "most people" like, instead of asking the user herself. To maintain the illusion of awareness, the word choice must be more intimate. In YDKJ in a game of two players, the host will greet them by saying something like:

> "Hey you two! Welcome to the show. It's good to have you here."

The language indicates the host's awareness of the two people who are actually in front of the screen. "It's good to have you *here*." The words suggest that the host and the users are actually in the same space.

Characters in programs designed by our company are typically ultra-casual in their conversational style. This creates a sense of intimacy that helps users forget that they are actually using a computer. This is an artistic choice. If the goal of a jellyvision program was to help you find the best doctor for a serious ailment, the host might be much less casual in style. Still, he should speak *to you*, that is, perform his lines and use language, that suggests that you are right in front of him.

Jellyvision is a Second Person Point-of-View Experience

Jellyvision programs are primarily a second person point-of-view experience. With a film, the viewer does nothing other than observe the action at a distance. This is known as a third person point-of-view experience. With a video game or a website, the user is primarily driving the action, whether she is literally driving a virtual car or telling the web browser where to go next. This is known as a first person point-of-view experience.

With jellyvision programs, the user is primarily the *object* of the action. The action of the program is happening *to her*. A jellyvision designer, writer or actor should always remember: almost all of the time, action is directed *at* the user. The most important word in a jellyvision program is “YOU.”

Shift into a Third Person Point-of-View with Caution

When a jellyvision program temporarily shifts into being a third person point of view experience, like television, it can break the user’s engagement with the program. For example, if two characters suddenly begin talking between themselves instead of involving the user, the program is no longer showing the user that it is aware of her. It is no longer personalized. During those moments, depending on how they are handled, the program can easily frustrate users or lose their attention; jellyvision users are *expecting* to be doing something. It isn’t like television where viewers aren’t expecting to do anything but watch.

There is a sequence in several of the original YDKJ programs called the “Fiber Optic Field Trip.” For up to 90 seconds the program shifts into a third-person point of view while the host “calls up” someone at random and asks them to come up with a trivia question. We knew that this was dangerous, but fortunately user testing revealed that the sequences managed to hold users’ attention and some actually said they appreciated the short break in the action. There were also very few Fiber Optic Field Trips in the game (six out of 800 questions). It was perceived almost like a surprise treat. In *You Don’t Know Jack, Volume 2* we did a variation on this known as the “Celebrity Collect Call.” We produced more than a dozen funny sequences with celebrities ranging from Phyllis Diller to Dana Carvey.

At some point, unfortunately, the novelty wore off. The feedback we sought out indicated that people no longer wanted to stare at their screen and listen to these conversations (even with big stars); they wanted to play the game. They wanted the program to be about them and not somebody else.

You can shift out of second person point of view occasionally, but it should be done with great diligence paid to the user’s reaction.

Calling Users by Name

The ultimate intimacy is to call the user by name. Theoretically, this is possible. Practically, it is very difficult to physically record thousands of names multiple times in the context of various sentences, so it sounds natural. There are circumstances, however, where if the task can reasonably be limited, it might make sense.

Even if a program is faithfully following all the principles for creating the illusion of awareness, the words the actors use and the feel of their performances must convey to the user that they are speaking directly to her. Otherwise the user will recognize that the dialogue is staged. She will be reminded that she is only sitting in front of a machine. The illusion, the necessary suspension of disbelief, will be lost.

Maintaining the Illusion of Awareness

Make Sure Characters Act Appropriately while the User is Interacting

A character in a jellyvision program must appear as if she is listening to the user.

Have you ever been at a party, midway through a sentence, when the person to whom you are talking suddenly starts talking to someone else who passes by? You're talking to him and he is talking to someone else. It's rude because it's quite clear the person isn't listening to you.

When a user is interacting with a jellyvision program, like typing or selecting something, she is essentially "talking" to the host character.

Let's say the host of our fictitious book review program asks you to type in the name of your favorite book. You start typing, but the host never stops talking. Whether he is talking to you or is talking to another character, the effect is the same: It will destroy the illusion of awareness. It will appear as if the character isn't aware that you've started responding. The character isn't "listening."

When you consider the same situation with voice input, the problem is even more blatant: you and the character are literally talking at the same time.

Normally, after a character asks a question, he should be quiet and wait for a response from the user. If the response doesn't come fast enough, then, based upon the principles from the section on maintaining pacing, the character should begin "prodding" the user.

Talking While the User is Able to Interact

There are occasionally times when it makes sense for a character to ask a question, but have the character continue to talk to provide further instruction in case it is needed. In our book review program, the host might ask you to type in a brief review of a book you've just read:

> "...four out of five stars, huh? O.K. so tell me why you liked it and what about it kept it from being a five star book...?"
>
> — at this point, a text field is available in which you can start typing if you so choose, but the host continues giving instructions...

> "...Just type your review into the box. It doesn't have to be long. Whatever you'd like to say. If it's just one or two sentences, that's fine..."

As long as you don't start typing, there's nothing wrong with this. If you immediately understand what to do, you can begin typing right away without having to hear all the instructions. On the other hand, the longer instructions are there for those who need them.

The Character Interrupts Himself if the User Begins Inputting

The only problem is if the user starts typing while the character is talking. In this case, the character should interrupt himself and stop talking.

> "...have to be long. Whatever you'd like to say. It it's just..."
>
> — user starts typing. A second audio file cuts off the first.
>
> "Oh. You've already got a quote. Great. "

In YDKJ, while Cookie the stage manager, is prodding you because you didn't type your name in fast enough, you can interrupt him by starting to type. Cookie will respond appropriately:

> "HELLO! Player Two, what's your deal, pal!? I just asked you to..."
>
> —and then Player Two starts typing his name. Cookie responds intelligently and with emotion by cutting himself off:
>
> "AH! Well, thank you...finally."

The "AH!" needs to be dramatic enough to sound human as it cuts off the previous audio file. In other cases, it may make more sense to just cut off the host audio, say nothing and just "listen." Human beings regularly do the same thing when they are interrupted.

If the characters in the program do not appear to be listening while the user is interacting, it can destroy the illusion of awareness.

Maintaining the Illusion of Awareness

Make Sure Dialogue Never Seems to Repeat

Users should never recognize that the same piece of dialogue is being repeated.

When human beings are in a situation where they have to say the same thing over and over again, they tend to say it in slightly different ways with somewhat different intonations. The characters in jellyvision programs should do the same thing.

In YDKJ, when the host announces the value of the question ("and it's worth $2000") there are multiple versions of generic host audio for this moment:

"...$2000 bucks for a right answer"

"...Two grand for this one"

"...2000 big ones for this question"

"...and you'll pocket 2K for getting it right"

Sometimes it makes sense to use the exact same words, but have the actor perform it differently.

Determining How Many Variations of Generic Dialogue are Needed

Regardless, there must be enough variations so that the user does not notice them repeating. For some generic moments within a program, that might be very few variations. In YDKJ, after Player Three buzzes in, there are only two files of generic dialogue for that moment:

"Player Three!"

"Player Three, go for it!"

The players might hear one of these files fifteen times in the same game, but they don't notice it. Here's why: (1) the dialogue is short and unmemorable, (2) the host performs the dialogue without any unusual inflection that would draw attention to it, (3) this moment in the game is a high point of excitement so the players are distracted and (4) there's really little more anyone *could* say in that situation. It more or less makes sense to repeat the same thing.

In our book review program, when the host reacts to the user rating a book with five stars, the program might use custom dialogue:

> "Five stars! Yes! Totally! Was the ending intense or what? Lois Lowry is just a fantastic author."

With custom dialogue like this, it typically doesn't repeat. The same moment, however, could also use generic dialogue:

> "Five stars! Wow. You're totally ga-ga about this one, huh? That's fantastic."

This audio file could be used any time the user rates a book with five stars. Note, however, that this dialogue includes the memorable words "ga-ga." If a user were to hear this twice within a week, she would certainly notice that the host had repeated this exact same phrase. This would ruin the illusion of awareness. On the other hand, it is highly unlikely that the user will both read several books in the same week and rate more than one of them with five stars. If the designer of this program added two additional generic variations and rotated their use, it is unlikely that the user would ever notice that these generic files were repeated:

> "Five stars! Wow. You're totally ga-ga about this one, huh? That's fantastic."
>
> "Whoa! You totally loved it. Cool."
>
> "Five stars? Outstanding."

There are other times when a program might require many variations for a particular recurring situation (generic moment). In YDKJ, when a player gets an answer wrong, sometimes there is custom dialogue that is specific to the exact wrong answer the person picked:

> "Dino didn't belong to George Jetson. You are in a serious time-warp."

If the writer hadn't written custom dialogue for this particular wrong answer, then one of the variations of "wrong answer generic dialogue" would get played:

> "Nice try. But trying doesn't count for much when you fail."

This piece of dialogue can be played every time someone gets a wrong answer. But if you heard this piece of dialogue twice in the same week, let alone twice in the same game, you'd notice it. Thus, YDKJ has dozens of variations of wrong answer generic dialogue. We determined the number of variations by estimating (roughly) that if you play the game regularly, you'd be unlikely to hear the same variation more than once every couple of weeks.

———

It often makes sense in a program to use generic dialogue, that is, audio or video files that can be used in recurring situations. However, a jellyvision program must have enough variations of generic dialogue for any one recurring situation so that the user doesn’t recognize that generic dialogue is getting replayed. The user will otherwise be reminded that she is simply in front of a machine playing back media files, thus shattering the illusion that she is interacting with a human being who is “alive” and “aware.”

Maintaining the Illusion of Awareness

Be Aware of the Number of Simultaneous Users

A jellyvision program that allows for multiple users at the same time, either in the same room in front of the same screen, or across a network, must know how many users there are and adjust its language to appropriately speak in the plural or the singular.

Dialogue that Differentiates the Second Person Singular and Plural

A feature of the English language is that the second person singular and plural are the same word: "you." Practically, however, when addressing small groups of two or more people, English-speakers regularly augment the plural: "you all," "you folks," "you both," "you guys," etc.

In YDKJ, the convention has been to be very specific:

> "Hey you two, how you doing? Welcome to the show."
>
> "We've got a trio, a triad. Nice to have all of you here."
>
> "Playing all alone today? That's all right, I'm all alone too. It's just you and me, kid. Ready to rock and roll?"

This, indeed, *creates* the illusion of awareness, but in truth, this is not a principle a designer can apply optionally; if you don't follow this principle, users will notice the lack of awareness and their suspension of disbelief will be destroyed.

Moreover, in a program where the host calls on all the users as a group and alternately on an individual in the group, people count on these sort of idiomatic differentiations between the singular and the plural to know who is being addressed. This is especially true when the speaker can't make actual eye contact to select out individuals in the group.

To follow this principle, somehow you need to find out how many people are actually participating. Over a network, the program can simply count the number of computers that are hooked into the program. In the YDKJ CD-ROM series, where the users are all in front of the same screen, we simply ask:

> "So, how many people are going to be playing the game today?"

Calling Users "by Name"

When the host has to speak with individuals or subgroups within a larger group, in some manner the host needs to refer to them "by name." In YDKJ, we get around this by calling users by their player number:

> "Player 2 and Player 3 move aside. Player 1 step on up to the plate, because it is time to play the Dis-or-Dat."

> "You know what, Player 2, you're horrible at this game. But you had the good fortune of playing against Player 1 and Player 3 who, as impossible as it seems, are even worse."

This seems a bit awkward calling people by player numbers, but it appears that they quickly accept the convention so it doesn't draw attention to itself. This is, of course, a place where it would be ideal to use peoples' actual names. Even looking at these two examples above, however, you can begin to appreciate the complexity of the task of "slotting" in the names of the users into the sentences in a way that would sound natural. In YDKJ, this problem is compounded by the fact that people regularly type in names like "rbb" or "XXX" or "j" or "Mr. Twister."

Differentiating between One User Alone and in a Group

Programs that allow for one *or* multiple users may need additional dialogue files to differentiate between these two situations when calling on one user. In YDKJ, when there are multiple players, and Player 1 buzzes in, the host will say:

> "Player 1, go for it!"

If there is only one player, the host will simply say:

> "Go for it!"

In this situation, the program *could* use the dialogue calling the individual "Player 1." However, a real human being dealing with someone one-on-one wouldn't do that. A human would know that there are no other players around and naturally adjust his language accordingly. Thus, in YDKJ, in any situation where the host calls on an individual player, there are always four groups of files: for Player 1, for Player 2, for Player 3 and for one player playing alone.

Human beings use different language when addressing an individual alone, a group or an individual in a group. Jellyvision programs need to be aware of who and how many people a character is talking to at any one time and account for that in the character's dialogue. Otherwise, users will be confused about whether they are being addressed and recognize this as the character’s lack of awareness.

Maintaining the Illusion of Awareness

Be Aware of the Gender of the Users

There will be circumstances in certain programs in which a character will need to mention another user in the third person. For example:

> "Well, here's what one of our other readers had to say about *Shiloh...*"
>
> —and then add:
>
> "...and if it gives her any more credibility, you should know that she *also* loves Holden Caulfield. O.K., here's what she said..."

Note that the host recognizes that the other user is female: "...*she* also loves Holden Caulfield." The actor who plays the host of this program would need to have also recorded the same sentence using the pronoun "he."

Dialogue that "Works-Around" Awareness of Gender

There may be moments within a program where a writer may be able to "work-around" being clear about gender:

> "I spoke to someone who really liked your review of *The Giver*. "
>
> —and then add:
>
> "This reader asked me to send you a message. Here it is..."

"This reader asked..." can reasonably substitute for having two files: "He asked..." and "She asked...." This substitution can be made because the flow of the dialogue above still sounds fairly natural. However, such a work-around should be the exception to the rule. Indeed, a jellyvision program that is being diligent about creating the illusion of awareness will typically record two variations using "he" and "she" whenever a pronoun is appropriate.

It is very awkward to sustain speech talking about someone in the third person without eventually resorting to pronouns:

> "This person thought your review was right on the mark, which doesn't surprise me since this particular reader has a long history of buying books that you like. I think you and this person would make good pen pals. They said they would be interested. What do you think?"

As you can see, eventually the host has to break down and use some kind of pronoun. If you don't know someone's gender, then you'll probably wind up using the plural pronoun "they." A character might be able to get away with this occasionally. More often than not, though, it will sound bizarre and draw attention to itself, breaking the illusion of awareness.

Asking and "Remembering" the User's Gender

The ability to apply this principle means, of course, that early on, a jellyvision program must ask the user if the user is male or female. That information must be maintained in a history file or a user's profile for future use. Moreover, before a user begins a jellyvision program, the user must identify him or herself, so the program can reference their profile and history. (This is important for the application of this principle and many others).

Clearly, it would be much easier if core demographic information (such as gender and zip code) were collected from the user at one initial sign-up on her computer or jellyvision set. That information could then be made available to every jellyvision program that a user decided to experience.

This is where important issues of one's right to privacy begin to surface. Who collects that information, how it stored, who has access to it and what control the user has over it need to be carefully considered.

Awareness of Gender is Usually, but Not Always Applicable

In YDKJ, we don't need to apply this principle because it rarely talks about users in the third person. When it does, it uses the convention of identifying people by their player number. "Games" with "players" will hopefully only be one genre of jellyvision within a much larger body of programs that cover other areas of entertainment, news and information, services, education, training, advertising and transactions. Programs in which a character needs to speak to a user about someone else will usually need to be cognizant of gender. Our book review program that tries to bring like-minded readers together is an example of this.

———

When a character is speaking to a user about another user, the ability of the character to "know" and accurately reference the gender of that other user is critical to maintaining the illusion of awareness.

Maintaining the Illusion of Awareness

Make Sure the Performance of Dialogue is Seamless

In order to respond dynamically to user input, jellyvision programs will often have to "mix & match" several pieces of audio or video of an actor's performance. Here's a typical sequence of audio in YDKJ:

"The category is...
"Astronomy & Laundry Detergent...
"Two grand for this one...
"Put your fingers on your buzzers, here's the question...
"If planets were housewives, which planet would need to use Wisk™?...
"Mercury, Saturn, Pluto, or Mars?"

This sequence of dialogue is made up of *six* separate sound files that all butt up next to one another. *Each sound file could have been recorded several months before or after the files adjacent to it.* It is usually critical that from one media file to the next the actor is consistent in his mood, pacing, inflections, volume, pitch, and accent. Significant shifts in any of these characteristics from one file to the next will betray the underlying technology (that is, reveal the fact that the program is stringing media files together). This will destroy the illusion of awareness, because the user will be reminded that "it's just a computer."

There are exceptions to this. For example, a shift of mood between two audio files, might make sense in the context of the writing. The bottom line is: if the actor were performing the dialogue straight through, would one long reading be similar to shorter multiple readings strung together? If not, there's a problem.

Stringing Audio Clips Together with Video Clips

It is important to note that audio *and video* files, with the right technology can also be strung together and maintain the illusion. *That's a Fact Jack!* does a decent job of this. The video host stands behind a podium and reads the category name on screen. The screen then cuts to a graphic with the name of the category. As the point value of the question dissolves on, voice-over audio of the host announces it, which is butted right up after the video:

Host on screen with video: "This category is 'Determination'..."

— cut to graphic that reads: "Determination"
— the words "25 points" begin to dissolve on

Voice-over audio of the host: "...it's worth 25 points..."

With the right technology, this combination of audio and video can cut together quite smoothly.

Stringing Video Clips Together

Theoretically, separate clips of video could also be "mixed and matched." For programs that rely completely on audio, the actor only has to deal with his vocal delivery. If the program includes visuals of an actor that butt up next to one another, the actor must also maintain continuity with his physical presence. Actors working in film have dealt with this challenge for years. Films, however, have the advantage of moving from shot-to-shot in a single sequence that is established (at least roughly) ahead of time. Jellyvision programs may combine a single shot with one of a dozen or more shots in various sequences.

Jellyvision has not experimented with this yet, but based on established film / video editing conventions, individual video clips can be strung together seamlessly if they are of:

(1) completely different images or

(2) adequately different angles of the same actor who has a relatively similar expression and posture at the end of the first clip and the beginning of the next.

If the clips are of the same actor from the same angle, the actor's facial expression and physical posture at the end of the first clip and the beginning of the next would have to be perfectly identical. Otherwise, the viewer would see a "jump cut" or unnatural jerkiness where the files butted together. If these sequential clips are shot in two separate takes (as opposed to cutting one long take in half), it is essentially impossible to perfectly recreate the actor's expression and posture for the second take.

Improvements in morphing technology could possibly provide some solutions here. Potentially, two video clips of the same actor shot from the same angle can be butted together seamlessly by doing a short morph to hide the cut. If the actor was careful about the positioning of his head and body, technology might be able to pull off this illusion without the user noticing. Still, the number of times audio files butt up next to one another in a single game of YDKJ is enormous. Given such requirements, the morphing technology would have to be highly automated or performed on the fly by the program.

Whether a jellyvision program presents its characters using video, audio or both, it will be constantly mixing and matching media files to build paragraphs and even sentences in order to respond to the user in a completely personalized way. The actor needs to understand how each piece of dialogue will butt up to the others that could play around it and delivery his performance so that they will all flow together seamlessly.

Avoid the Presence of Characters when User Input Cannot be Evaluated

If a user's response is freeform (that is, the user is inputting a response that the program cannot always evaluate—like Fill-in-the-Blank) it may make sense to create a situation at that moment within the program such that the user is not expecting the character to make any response at all.

Limitations to Responding Intelligently to Undirected Freeform Input

In our fictional book buying program, when the character asks the user to type out her review of the book, the user might type out:

> First of all, reading the opening chapters was like going to a block party where the beer is free and the kids aren't obnoxious. It drew me in. But by the time it got to the part where the Gorman family moved into the neighborhood, it just dragged like an old mule.

At this stage in history, a computer program cannot respond with human-like intelligence and emotion to a freeform response like this.

For example, the first sentence about the block party where "the beer is free and the kids aren't obnoxious" might be an allusion to either a scene in this book, one of the author's earlier works or something current in the news that week. A computer program today could never figure that out. It might be able to pick out words like "obnoxious." However, the program might not recognize that "obnoxious" was being used in the context of a *positive* comparison. In the last sentence, the user mentions that much of the book "dragged on like an old mule." There is certainly no technology today that could form text to respond with human-like intelligence to such a random simile.

However, the host of the book review program could simply recognize that the user had completed the task: "O.K. thanks." There will be many situations where this is acceptable. Although it's true that with this type of generic response, the program will necessarily betray its limitations and soften the strength of the illusion of awareness.

"Removing" the Character to Maintain the Illusion

The other approach is to completely remove the character from the situation all together.

> "So, do you mind typing up a few sentences as to why you didn't like the book? Just hit SUBMIT when you're done and your review will shoot off to our Member's Book Review Compendium. I'll come back when you're done and if you want, we'll look for another book."

The user is "alone" temporarily. The character is not present and thus, cannot "see" the user's words. If the user's words have been "submitted" and are no longer on the screen by the time the character "gets back," the user probably will not be expecting the character to be aware of what she has written. Perfect it isn't, but it is one technique to maintain the illusion when you bump into the limits of technology.

Characters Can be Present when Freeform Input Can be Evaluated

There are exceptions. For Fill-in-the-Blank questions, in YDKJ, the host is "present." As mentioned earlier, these are freeform response situations in which to interact *successfully*, there is, in fact, one proper response. So quite often, the program can and will be aware of a user's freeform input and maintain the illusion. This approach can work quite well, if a generic response ("I don't think so...") is acceptable when the user *does not* give the proper response.

There are certain types of input from a user which a program cannot intelligently evaluate. These include any typed in or voice responses that cannot be mapped to a database of possible answers. Another example would be open-ended drawings. In these cases, the character can simply acknowledge that the user has completed the task without commenting on it or the character can appear to "leave the user alone" until she is finished with the task and "come back" after her response is removed off the screen. This can potentially finesse the problem of the character's inability to respond with awareness; the character "doesn't see" the user's input, so he doesn't have to respond to it.

Final Thoughts about Maintaining the Illusion of Awareness

Never Let the User Smell the Technology

When YDKJ was first being created, the question came up: what kind of "busy cursor" should we display at those moments when the user has to wait for the program to retrieve information off the CD-ROM? The answer was: the user should never feel like she is waiting for anything. This would not only destroy the pacing of the program, but users would then "smell" the technology, shattering their suspension of disbelief. The illusion would be ruined.

Of course, YDKJ did have to go load information off the CD-ROM. At that time, if a computer was loading, any animation on the screen had to freeze otherwise it would be jumpy. PC's had a hard time getting information off a CD-ROM while simultaneously smoothly displaying animation. However, PC's could play audio smoothly and load off the CD-ROM at the same time.

Given this, there were two courses of action we took: one technological and one creative. First, the engineers looked for moments in the game when there was no animation, only sound, and loaded as much media as they could during those temporary visual lulls. In a number of places, however, we still needed more loading time than the program was naturally offering. So, we got creative.

The entire opening of the JackAttack was conceived because we needed to cover up a large load of data off the CD-ROM. All of the music from the JackAttack had to be loaded at one time, because there could be no loading during the question itself since it featured wall-to-wall animation. Berkeley System's lead engineer originally told us that he needed five seconds of load time on the minimum configuration machine. This is an eternity for a jellyvision program. Through clever engineering, he managed to shave it down to 3-3/4 seconds.

We didn't cover up those 3-3/4 seconds with a busy cursor or a status bar. We set out to cover it up by creating an additional audio/visual sequence (that was almost all audio and no visual) which would made sense creatively in the context of the game.

As you select the JackAttack category, the screen completely freezes and you hear the sound of a foreboding church bell and thunder rattling off in the distance. This lasts for about 2-1/4 seconds.

Meanwhile, underneath this audio distraction, the technology is frantically loading media off the CD-ROM.

Then, you hear two large ripping sounds as the screen appears to quickly tear away in two parts. The ripping sound effect and the tearing animation have a large audio/visual impact on the user, but they are very tiny files. The screen is now without animation, in fact, it is without any graphics at all...it's totally black. You hear the whizzing sound of an object flying against the wind moving toward you from a distance.

Underneath this audio distraction, the technology is madly hitting the CD-ROM trying to complete the last 1-1/2 seconds of load.

The moment the 1-1/2 seconds are up, the load is finished and the whizzing sound crescendos as the YDKJ ball, like a Molotov Cocktail, flies on to the screen and explodes into the JackAttack logo. Animation resumes, the program is back on its way, and hopefully, you had no conception that you ever waited for anything.

When a designer is covering up a load creatively, there is one goal: make it look like it was planned that way all along.

Covering up today's technological limitations takes a combination of technical problem solving—and when that wall is hit—creative problem solving. The strongest designers know that when they run into roadblocks, the solutions they create to get around them will often be better than their original ideas. The technical limitations, as frustrating as they are, regularly make you better. Embrace them.

The reason that none of the ideas above are wrapped up into a Jack Principle is because we believe that these technical limitations are temporary problems. Berkeley Systems, which did the incredible engineering for *You Don't Know Jack: The Net Show*, released in 1996, ran into similar loading problems while pulling media through the Internet on a 28.8K modem. Now, in 2000, using Flash-based vector technology, audio is the bandwidth hog. So, you have cover up audio loads with art and animation, instead of vice-versa. These limitations, however, will diminish over time. At some point (and who knows when), you will be able to sit in your house and have an interactive program mix and match ten channels of video, twenty channels of audio, and forty channels of animation on the fly. Loading will not be a problem that designers have to work around, the technology will handle it automatically.

In the meantime, however, it is important to remember: *Never Let the User Smell the Technology*. To do this, both engineers and designers have to work together closely and be very clever.

Don't Forget who is the Star of the Show

Although this document begins by legitimating our approach to interactive design by comparing it to television, jellyvision is *not* television. It feels more like television than other forms of interactivity, but it has its own set of defining characteristics.

In particular, the second person point-of-view nature of jellyvision represents a major break with the psychological dynamics of television:

The star of the show is not on the screen;
the star of the show is sitting in front of it.

This axiom should permeate every decision a jellyvision designer makes. Never, never forget, in a jellyvision program, your audience is the star.

Concluding Thoughts

The question was once raised: "So what is so good about this form of communication? How does this help the world?"

No form of mass communication is intrinsically good or bad. Like literature, radio, television, theater, film and magazines, jellyvision is just a way of communicating. There are great books and horrible books. There are delightful films and others that are derogatory and insipid. There are fabulously informative websites and others that encourage violence. People will use the medium of jellyvision no differently.

Whether this medium will ultimately be seen as helping the world or hurting it depends on the virtue of the people who master and practice the craft. We are incredibly fortunate to be alive at a moment in history when science has handed us a completely new and powerful way of entertaining, informing, serving and educating millions of people at once.

The possibilities are nothing short of glorious.

If the ideas in this document have made sense to you, by all means, please set out and take advantage of them.

And use them in a way in which you can be proud.