

Guidelines for Designing Character Interaction

ActiveX™ Technology for Interactive Software Agents



August 1997
Microsoft Corporation

Note: This document is provided for informational purposes only and Microsoft makes no warranties, either expressed or implied, in this document. The entire risk of the use or the results of this document remains with the user.

Information in this document is subject to change without notice. Companies, names, and data used in examples herein are fictitious unless otherwise noted. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents or pending patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. The furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property rights. Microsoft, MS, MS-DOS, Windows, Windows NT, and the Windows logo are either registered trademarks or trademarks of Microsoft Corporation in the U.S. and/or other countries. Other product and company names mentioned herein may be the trademarks of their respective owners.

Contents

Introduction

General Character Interaction Guidelines

Social Interaction

Speech Recognition

Speech Output

Bibliography

Introduction

This document outlines guidelines for designing user interfaces that incorporate interactive characters created with Microsoft Agent. For specific information on designing character animation, see *Designing Characters for Microsoft Agent*.

General Character Interaction Guidelines

Principles of good user interface design also apply to designing an interface using Microsoft Agent. You should understand your users and their goals as well as follow a good user interface design process. *The Windows® Interface Guidelines for Software Design* provides an excellent overview of general design principles and methodology.

Be Non-Exclusive

Interactive characters can be employed in the user interface as assistants, guides, entertainers, storytellers, sales agents, or in a variety of other roles. A character that automatically performs or assists should not be designed contrary to the design principle of keeping the user in control. When adding a character to the interface of a Web site or conventional application, use the character as an enhancement -- rather than a replacement -- of the primary interface. Avoid implementing any feature or operation that exclusively requires a character.

Similarly, let the user choose when they want to interact with your character. A user should be able to dismiss the character and have it return only with the user's permission. Forcing character interaction on users can have a serious negative effect. To support user control of character interaction, Microsoft Agent automatically includes Hide and Show commands. The Microsoft Agent API also supports these methods, so you can include support for these functions in your own interface. In addition, Microsoft Agent's user interface includes an Exit command that enables the user to suspend all character interaction, as well as global properties that enable the user to override certain character output options. To ensure that the user's preferences are maintained, these properties cannot be overridden through the API.

Provide Appropriate Feedback

Quality, appropriateness, and timing are important factors to consider when providing feedback in any interface design. When you incorporate interactive characters, the opportunities for natural forms of feedback increase, as does the user's expectation that the feedback conform to appropriate social interaction. A character can be designed to provide verbal and non-verbal conversational cues in addition to spoken audio output. Use gestures or facial expressions to convey information about its mood or intent. The face is especially important in communication, so always consider the character's facial expression. Keep in mind that no facial expression is a facial expression.

We humans have an orienting reflex that causes us to attend to changes in our environment, especially changes in motion, volume, or contrast. Therefore, character animation and sound effects should be kept at a minimum to avoid distracting users when they aren't directly interacting with the character. This doesn't mean the character must freeze, but natural idling behavior such as breathing or looking around is preferable to greater movement. Idling behavior maintains the illusion of social context and availability of the character without distracting the user. You may also want to consider removing the character if the user hasn't interacted with it for a set time period, but make sure the user understands why the character is going away.

Conversely, large body motion, unusual body motion, or highly active animation is very effective if you want to capture the user's attention, particularly if the animation occurs outside the user's current focus. Note also that motion toward the user can effectively gain the user's attention.

Placement and movement of the character should be appropriate to its participation in the user's current task. If the current task involves the character, the character can be placed at the point of focus. When the user is not interacting with the character, move it to a consistent "standby" location or where it will not interfere with tasks or distract the user. Always provide a rationale for how the character gets from one location to another. Similarly, users feel most comfortable when the character appears in the same screen location from which it departed.

Use Natural Variation

While consistency of presentation in your application's conventional interface, such as menus and dialog boxes, makes the interface more predictable, vary the animation and spoken output in the character's interface. Appropriately varying the character's responses provides a more natural interface. If a character always addresses the user exactly the same way; for example, always saying the same words, the user is likely to consider the character boring, disinterested, or even rude. Human communication rarely involves precise repetition. Even when repeating something in a similar situation, we may change wording, gestures, or facial expression.

Microsoft Agent enables you to build in some variation for a character. When defining a character's animations, you can use branching probabilities on any animation frame to change an animation when it plays. You can also assign multiple animations to each state. Microsoft Agent randomly chooses one of the assigned animations each time it initiates a state. For speech output, you can also include vertical bar characters in your output text to automatically vary the text spoken. For example, Microsoft Agent will randomly select one of the following statements when processing this text as part of the **Speak** method:

"I can say this.|I can say that.|I can say something else."

Social Interaction

Human communication is fundamentally social. From the time we are born, we begin reacting to the social cues in our environment and learning the appropriate rules for effective interaction, which include verbal behaviors such as intonation or word ordering, and also non-verbal behaviors such as posture, gestures, and facial expressions. These behaviors convey attitudes, identities, and emotions that color our communication. We often create substitute conventions for communication channels that don't naturally provide bandwidth for non-verbal cues, such as e-mail or online chat sessions.

Unfortunately, the majority of software interface design has focused primarily on the cognitive aspects of communication, overlooking most social aspects. However, recent research has demonstrated that human beings naturally react to social stimuli presented in an interactive context. Further, the reactions often follow the same conventions that people use with each other. Even the smallest cues, such as colors presented or word choice in messages, can trigger this automatic response. The presentation of an animated character with eyes and a mouth heightens the social expectations of and strength of responses to the character. Never assume that users expect a character's behavior to be less socially appropriate because they know it is artificial. Knowing this, it is important to consider the social aspects of interaction when designing character interaction. *The Media Equation: How People Treat Computers,*

Televisions, and New Media as Real People and Places by Byron Reeves and Clifford Nass (New York: Cambridge University Press) is an excellent reference on current research in this area.

Create Personality

We quickly classify the personality of people we meet based on the simplest of social cues, such as posture, gesture, appearance, word choice, and style. So the first impression a character makes is very important. Creating personality doesn't require artificial intelligence or realistic rendering. Great animators have known this for years and have used the simplest social cues to create rich personalities for inanimate objects. Consider, for example, the flying carpet in Disney's *Aladdin*, and Lassiter's *Luxo Jr.*, a humorous animated video of a pair desk lamps. Beginning animators at Disney were often given the challenge of drawing flour sacks that expressed emotion.

A character's name, how it introduces itself, how it speaks, how it moves, and how it responds to user input can all contribute to establishing its basic personality. For example, an authoritative or dominant style of personality can be established by a character making assertions, demonstrating confidence, and issuing commands, whereas a submissive personality may be characterized by phrasing things as questions or making suggestions. Similarly, personality can be conveyed in the sequence of interaction. Dominant personalities always go first. It is important to provide a distinct, well-defined personality type, regardless of which personality type you are creating. Everyone generally dislikes weakly defined or ambiguous personalities.

The kind of personality you choose for a character depends on your objective. If the character's purpose is to direct users toward specific goals, use a dominant, assertive personality. If the character's purpose is to respond to users' requests, use a more submissive personality.

Another approach is to adapt a character's personality to the user's. Studies have shown that users prefer interaction with personalities most like themselves. You might offer the user a choice of characters with different personalities or observe the user's style of interaction with the character and modify the character's interactive style. Research shows that when attempting to match a user's personality you don't always have to be 100% correct. Humans tend to show flexibility in their relationships and, because of the nature of social relationships, are also likely to modify their own behavior somewhat to working with a character.

Observe Appropriate Etiquette

All humans learn rules of social etiquette. Moreover, we judge others based on how well they conform to these rules. The presence of a character in your interface makes rules of etiquette directly applicable to your design.

We expect reciprocity in our interaction and when it doesn't happen we tend to view the behavior as incompetent or offensive. In new or formal situations, politeness is usually expected. Only after a relationship is established does familiarity allow for less formal interaction. So when in doubt, be polite.

For example, consider the appropriate protocol for starting and ending a social interaction. When engaging in a new social context, we generally greet each other and indicate our intent to leave before we depart. You can apply this model directly to character interaction; avoid the character appearing or disappearing without an appropriate indication.

A part of politeness is how we respond. We expect responses that are appropriate. For example, if you ask, "What day is it?" you don't expect the response to be "10 apples." However, accuracy and relevance are not enough; you may also need to determine the context. An accurate response could be "Monday" or "August 25th." Similarly, the wording of a response can be accurate but still be offensive, depending on intonation, facial expression, or other non-verbal behaviors.

In addition, we address each other while talking: we face the person, turning only to direct attention to something. Turning one's back, without some purpose, implies disinterest and is generally considered to be impolite.

Politeness also affects the way we respond to or perceive others. For example, social research indicates that we are more likely to be honest about our evaluation of others when asked by another person than when the people ask about themselves. Similarly, we accept positive evaluations of others more readily than positive self-evaluations. Immodest behavior is generally considered impolite. Therefore, unless intentionally trying to promote an impolite personality style, avoid having the character compliment its own performance.

Also consider the cultural impact on social etiquette. This may apply to the character's attire, gestures, or word choice. While polite behaviors may vary by culture, the concept of politeness is cross-cultural.

Use Praise

Humans respond better to praise than criticism, even when the praise may be unwarranted. Although most software interfaces strive to avoid evaluative feedback, praise is demonstrably more effective than providing no evaluation at all. Further, many interfaces designed as neutral are often perceived as critical by users, because they rarely get positive feedback when things are operating normally, and error messages when things go wrong. Similarly, wording that tells the user that there is a better way to perform a task implicitly criticizes the user.

Because characters create a social context, careful use of praise and criticism is even more important than in a traditional user interface. While few people enjoy sycophantic behavior, the limits of the liberal use of praise have yet to be demonstrated. Praise is especially effective in situations where users are less confident about their performance of a task. On the other hand, criticism should be used sparingly. Even when you think criticism is appropriate, be aware that humans have a tendency to dismiss it or redirect it back to its source.

However, avoid self-praise unless it is a humorous part of the personality the character projects. We tend to judge self-conceit with skepticism. If you want to direct approbation to a character, have it directed from another source, such as another character or descriptive explanation.

Create a Team Player

When a team is created, group dynamics have a powerful effect on the members in the group. First, people in a group or team context have a tendency to identify more with the other people on the team than they typically would in a non-team setting. As a result, they can also identify more with their teammates than those outside the team. But equally important, members of a team are more willing to cooperate and modify their attitudes and behavior. Because the social dynamics of a team affect its members' interaction, it can be useful to consider when designing interaction with characters.

Creating a sense of team involves two factors: identity and interdependence. You can create identity by creating a team name, color, symbol, or other identifier that the user and character share. For example, you could provide a sticker the user could affix to their computer or enable the user to pick a team name or icon that would appear with the character. Team identity may also be established by what the character says. For example, you could have the character refer to itself as a partner or to the user and itself as a team.

Interdependence may be harder to implement or take longer to establish, though it is important to consider because interdependence seems to have a stronger social impact than team identity. This is illustrated by the product brand loyalty that marketing organizations endeavor to establish. Creating a sense of interdependence involves demonstrating continuing usefulness and reliability for the user. Important questions to answer are “Does the character provide value?” and “Does the character operate predictably and reliably?” An important factor here is how the character’s relationship is established with the user. To engender a sense of team interdependence, the character needs to be presented as a peer to the user. Although it may be useful in some scenarios to present the character as an expert or a servant, to leverage the collaborative benefits of team dynamics, there must be a sense of equality where the user can be dependent on the character without a sense of inferiority. This may be as simple as having the character refer to itself as a teammate or companion rather than as a wizard. It can also be influenced by how the character requests information from the user. For example, a character might say, “Let’s work together to answer this question...”

Consider Gender Effects

Social responses can be affected by gender. For example, Nass and Reeves’ work indicates that in the U.S., male personalities are stereotypically considered to understand more about technical subjects, while females are attributed to be better experts in interpersonal subjects like love and relationships. This doesn’t suggest that you should perpetuate gender stereotypes, only that you be aware that they may exist and understand how they might affect interaction with your character.

Remember that a character’s perceived gender isn’t solely established by its name or appearance. Even a gender-neutral character may convey gender based on its voice characteristics, speech patterns, and movement.

Speech Recognition

Speech recognition provides a very natural and familiar interface for interacting with characters. However, speech input also presents many challenges. Speech engines currently operate without substantial parts of the human speech communication repertoire, such as gestures, intonation, and facial expressions. Further, natural speech is typically unbounded. It is easy for the speaker to exceed the current vocabulary, or *grammar*, of the engine. Similarly, wording or word order can vary for any given request or response. In addition, speech recognition engines must often deal with large variations in the speaker’s environment. For example, background noise, microphone quality, and location can affect input quality. Similarly, different speaker pronunciations or even same-speaker variations, such as when the speaker has a cold, make it a challenge to convert the acoustic data into representational understanding. Finally, speech engines must also deal with similar sounding words or phrases in a language, such as “new,” “knew,” and “gnu,” or “wreck a nice beach” and “recognize speech.”

Speech isn’t always the best form of input for a task. Because of the turn-taking nature of speech, it can often be slower than other forms of input. Like the keyboard, speech input is a

poor interface for pointing unless some type of mnemonic representation is provided. Therefore, always consider whether speech is the most appropriate input for a task. It is best to avoid using speech as the exclusive interface to any task. Provide other ways to access any basic functionality using methods such as the mouse or keyboard. In addition, take advantage of the multi-modal nature of using speech in the visual interface by combining speech input with visual information that helps specify the context and options.

Finally, the successful use of speech input is due only in part to the quality of the technology. Even human recognition, which exceeds any current recognition technology, sometimes fails. However, in human communication we use strategies that improve the probability of success and that provide error recovery when something goes wrong. Therefore, the effectiveness of speech input also depends on the quality of the user interface that presents it.

Studying human models of speech interaction can be useful when designing more natural speech interfaces. Recording actual human speech dialogues for particular scenarios may help you better understand the constructs and patterns used as well as effective forms of feedback and error recovery. It can help determine the appropriate vocabulary to use (for input and output). It is better to design a speech interface based on how people actually speak than to simply derive it from the graphical interface in which it operates.

Note that Microsoft Agent uses the Microsoft Speech API (SAPI) to support speech recognition. This enables Microsoft Agent to be used with a variety of compatible engines. Although Microsoft Agent specifies certain basic interfaces, the performance requirements and quality of an engine may vary.

Speech is not the only means of supporting conversational interfaces. You can also use natural-language processing of keyboard input in place of or in addition to speech. In those situations, you can still generally apply guidelines for speech input.

Listen, Don't Just Recognize

Successful communication involves more than recognition of words. The process of dialogue implies exchanging cues to signal turn-taking and understanding. Characters can improve conversational interfaces by providing cues like head tilts, nods, or shakes to indicate when the speech engine is in the listening state and when something is recognized. For example, Microsoft Agent plays animations assigned to the **Listening** state when a user presses the push-to-talk listening key and animations assigned to the **Hearing** state when an utterance is detected. When defining your own character, make sure you create and assign appropriate animations to these states. For more information on designing characters, see *Designing Characters for Microsoft Agent*.

In addition to non-verbal cues, a conversation involves a common context between the conversing parties. Similarly, speech input scenarios with characters are more likely to succeed when the context is well established. Establishing the context enables you to better interpret similar-sounding phrases like "check's in the mail" and "check my mail." You may also want to enable the user to query the context by providing a command, such as "Help" or "Where am I," to which you respond by restating the current context, such as the last action your application performed.

Microsoft Agent provides interfaces that enable you access the best match and the two next best alternatives returned by the speech recognition engine. In addition, you can access confidence scores for all matches. You can use this information to better determine what was spoken. For example, if the confidence scores of the best match and first alternative are close, it may indicate that the speech engine had difficulty discerning the difference between them. In

such a case, you may want to ask the user to repeat or rephrase the request in an effort to improve performance. However, if the best match and first or second alternatives return the same command, it strengthens the indication of the correct recognition.

The nature of a conversation or dialogue implies that there should be a response to spoken input. Therefore, a user's input should always be responded to with verbal or visual feedback that indicates an action was performed or a problem was encountered, or provides an appropriate reply.

Clarify and Limit Choices

Speech recognition becomes more successful when the user learns the range of appropriate grammar. It also works better when the range of choices is limited. The less open-ended the input, the better the speech engine can analyze the acoustic information input.

Microsoft Agent includes several built-in provisions that increase the success of speech input. The first is the Commands Window displayed when the user says, "Open Commands Window," or "What can I say?" (or when the user chooses Open Commands Window from the character's pop-up menu). The Command Window serves as a visual guide to the active grammar of the speech engine. It also reduces recognition errors by activating only the speech grammar of the input-active application and Microsoft Agent's global commands. Therefore, the active grammar of the speech engine applies to the immediate context. For more information on the Commands Window, see *Microsoft Agent Programming Interface Overview*.

When you create Microsoft Agent voice-enabled commands, you can author the caption text that appears in Commands Window as well as its voice text (grammar), the words that the engine should use for matching this command. Always try to make your commands as distinctive as possible. The greater the difference between the wording of commands, especially for the voice text, the more likely the speech engine will be able to discriminate between spoken commands and provide an accurate match. Also avoid single-word or very short commands. Generally, more acoustic information in a spoken utterance gives the engine a better chance to make an accurate match.

When defining the voice text for a command, provide a reasonable variety of wording. Requests that mean the same thing can be phrased very differently, as illustrated in the following example:

Add some pepperoni.

I'd like some pepperoni.

Could you add some pepperoni?

Pepperoni, please.

Microsoft Agent enables you to easily specify alternatives or optional words for the voice grammar for your application. You enclose alternative words or phrases between parentheses, separated by a vertical bar character. You can define optional words by enclosing them between square bracket characters. You can also nest alternatives or optional words. In addition, you can also use an ellipsis (...) in voice text as a placeholder for any word. However,

using ellipses too frequently may make it more difficult for the engine to distinguish between different voice commands. In any case, always make sure that your voice text includes at least one distinctive word for each command that is not optional. Typically, this should match a word or words in the caption text you define that appears in the Commands Window.

Although you can include symbols, punctuation, or abbreviations in your caption text, avoid them in your voice text. Many speech recognition engines cannot handle symbols and abbreviations or may use them to set special input parameters. In addition, spell out numbers. This also ensures more reliable recognition support.

You can also use directive prompts to avoid open-ended input. Directive prompts implicitly reference the choices or explicitly state them, as shown in the following examples:

What do you want?	Too general, an open-ended request
Choose a pizza style or ingredient.	Good, if choices are visible, but still general
Say "Hawaiian," "Chicago," or "The Works."	Better, an explicit directive with specific options

This guides the user toward issuing a valid command. By suggesting the words or phrase, you are more likely to elicit expected wording in return. To avoid unnatural repetitiveness, change the wording or shorten the original for subsequent presentation as the user becomes more experienced with the input style. Directive prompts can also be used in situations where the user fails to issue a command within a prescribed time or fails to provide an expected command. Directive prompts can be provided using speech output, your application interfaces, or both. The key is helping the user know the appropriate choices.

Wording influences the success of a prompt. For example, the prompt, "Would you like to order your pizza?" could generate either a "Yes" or "No" response, but it might also generate an order request. Define prompts to be non-ambiguous or be prepared to accept a larger variety of possible responses. In addition, note the tendency for people to mimic words and constructs they hear. This can often be used to help evoke an appropriate response as in the following example:

User:	Show me all messages from Paul.
Character:	Do you mean Paul Allen or Paul Maritz?

This is more likely to elicit the full name of one of the parties with the possible prefix of "I mean" or "I meant."

Because Microsoft Agent characters operate within the visual interface of Windows, you can use visual elements to provide directive prompts for speech input. For example, you can have the character gesture at a list of choices and request that the user select one, or display choices in a dialog box or message window. This has two benefits: it explicitly suggests the words you want the user to speak and it provides an alternate way for the user to reply.

You can also use other modes of interaction to subtly suggest to users the appropriate speech grammar, as shown in the following example:

User: (Clicks Hawaiian-style pizza option with the mouse)

Character: Hawaiian-style pizza.

User: (Clicks Extra Cheese option with the mouse)

Character: Add "Extra Cheese."

Another important factor in successful speech input is cueing the user when the engine is ready for input, because many speech engines allow only a single utterance at a time. Microsoft Agent provides support for this in two ways. First, if the sound card supports MIDI, Microsoft Agent generates a brief tone to signal when the speech-input channel is available. Second, the Listening Tip window displays an appropriate text prompt when the character (speech engine) is listening for input. In addition, this tip displays what the engine heard.

Provide Good Error Recovery

As with any well-designed interface, the interactive process should minimize the circumstances that lead to errors. However, it is rarely possible to eliminate all errors, so supporting good error recovery is essential to maintain the confidence and interest of the user. In general, error recovery involves detecting an error, determining the cause, and defining a way to resolve the error. Users respond better to interfaces that are cooperative, that work with the user to accomplish a task.

The first step in speech error recovery is detecting the failure condition. Speech recognition can fail due to a variety of errors. Error conditions can usually be detected as the result of invalid input, explicit user correction or cancellation, or user repetition.

A *rejection error* occurs when the recognition engine has no match for what the user has said. Background noise or early starts are also common causes of recognition failure, so asking the user to repeat a command is often a good initial solution. However, if the phrase is outside of the current active grammar, asking the user to rephrase the request may solve the problem. The difference in wording may result in a match with something in the current grammar. Listing or suggesting appropriate expected input options is another alternative.

A good strategy for rejection error recovery is to combine these techniques to get the user back on track, offering increasingly more assistance if the failure persists. For example, you can begin by responding to the initial failure with an interrogative like "Huh?" or "What?" or a hand-to-the-ear gesture. A short response increases the likelihood that the user's repeated statement will not fail because the user spoke too soon. Upon a repeated failure, the subsequent request to rephrase improves the chance of matching something within the given grammar. From here, providing explicit prompts of accepted commands further increases the chance of a match. This technique is illustrated in the following example:

User: I'd like a Chicago-style pizza with anchovies.

Character: (Hand to ear) Huh?

User: I want a Chicago pizza with anchovies.

Character: (Head shake) Please rephrase your request.

User: I said Chicago pizza, with anchovies.

Character: (Shrug) I'm sorry. Tell me the style of pizza you want.

User: Chicago, with anchovies.

Character: Still no luck. Here's what you can say: "Chicago," "Hawaiian," or "Combo."

To make the error handling feel more natural, make sure you provide a degree of random variation when responding to errors. In addition, a natural user reaction to any request to repeat a response is to exaggerate or increase the volume when repeating the statement. It may be useful to occasionally remind the user to speak normally and clearly, as the exaggeration or increased volume may make it harder for the speech engine to recognize the words.

Progressive assistance should do more than bring the error to the user's attention; it should guide the user toward speaking in the current grammar by successively providing more informative messages. Interfaces that appear to be trying to understand encourage a high degree of satisfaction and tolerance from the user.

Substitution errors, where the speech engine recognizes the input but matches the wrong command, are harder to resolve because the speech engine detects a matching utterance. A mismatch can also occur when the speech engine interprets extraneous sounds as valid input (also known as an *insertion error*). In these situations, the user's assistance is needed to identify the error condition. To do this, you can repeat what the speech engine returned and ask the user to confirm it before proceeding:

User: I'd like a Chicago-style pizza.

Character: Did you say you'd like a "Chicago-style pizza"?

User: Yes.

Character: What additional ingredients would you like on it?

User: Anchovies.

Character: Did you say "anchovies"?

User: Yes.

However, using this technique for every utterance becomes inefficient and tiresome. To handle this, restrict confirmation to situations that have significant negative consequences or increase

the complexity of the immediate task. If it is easy for the user to make or reverse changes, you may be able to avoid requesting confirmation of their choices. Similarly, if you make choices visible you may not need to provide explicit correction. For example, choosing an item from a list may not require verification because the user can see the results and easily change them. You can also use confidence and alternative scores to provide a threshold for confirmation. You might adjust the threshold by keeping a history of the user's actions in a given situation and eliminating verification based on consistent user confirmation. Finally, consider the multi-modal nature of the interface. Confirmation from the mouse or keyboard may also be appropriate.

Carefully choose the wording of confirmations. For example, "Did you say...?" or "I think you said..." are better than "Do you really want to...?" because the former phrases imply that the accuracy of the character's listening (recognition) is being queried, not that the user may have misspoken.

Also consider the grammar for a response. For example, a negative response is likely to generate a rejection error, requiring an additional prompt as shown in the following example:

User: I'd like some pepperoni.
Character: Did you say "no ham"?
User: No, I said pepperoni.
Character: Huh?
User: Pepperoni.

Modifying your grammar to include prefixes to handle natural response variations increases the efficiency of the recovery process, especially when the user doesn't confirm the verification prompt. In this example, the confirmation could have been handled in a single step by modifying the grammar for the "pepperoni" by also including "no I said pepperoni", "I said pepperoni", and "no pepperoni".

You can also handle substitution errors using the alternative matches returned by the speech engine as the corrective confirmation:

User: I'd like some pepperoni.
Character: (Hears "no ham" as best match, "pepperoni" as first alternative) Did you say "no ham"?
User: No, pepperoni.
Character: (Still hears "no ham" as best match, but now offers first alternative) "Pepperoni"?

Similarly, you can keep a history of common substitution errors and if a particular error is frequent, offer the alternative the first time.

In any recognition error situation, avoid blaming the user. If the character suggests or even implies that the user is to blame, or the character seems indifferent to the error, the user may become offended. Here also, carefully choose wording that explicitly accepts responsibility, is appropriate to the situation, and uses variety to create a more natural response. When expressing an apology, avoid ambiguous words like “oops” or “uh-oh” that could be interpreted as blaming the user. Instead, use phrases like “I’m sorry” or “My mistake.” Repeated or more serious errors might use a more elaborate apology like “I am really sorry about that.” Also consider the personality of the character when determining the type of response. Another option is to blame an external situation. Comments such as, “Boy, it’s noisy out there,” take the blame away from the user and the character. Reminding the user of the cooperative nature of the interaction may be helpful as well: consider phrases such as, “Let’s see what we can do to make this work.”

Microsoft Agent also supports some automatic feedback for recognition. When an utterance is detected, the Listening Tip displays the voice text of the best match heard. You can set your own text to display based on the confidence setting for a command you define.

Because of error potential, always require confirmation for any choices that have serious negative consequences and are irreversible. Naturally, you’ll want to require confirmation when the results of an action could be destructive. However, consider also requiring confirmation for situations that abort any lengthy process or operation.

Speech Output

Like speech input, speech output is a familiar and natural form of communication, so it is also an appropriate complement in a character-based interface. However, speech output also has its liabilities. In some environments, speech output may not be preferred or audible. In addition, by itself, speech is invisible and has no persistent output, relying heavily on short-term memory. These factors limit its capacity and speed for processing large amounts of information. Similarly, speech output can also disrupt user input, particularly when speech is the input method. Speech engines generally have little support that enables the user to interrupt when speech or other audio has the output channel.

As a result, avoid using speech as the exclusive form of output. However, because Microsoft Agent presents characters as a part of the Windows interface, it provides several advantages over speech-only environments. Characters can be combined with other forms of input and output to make options and actions visible, enabling a more effective interface than one that is exclusively visual or speech-based.

In addition, to make speech output more visible, Microsoft Agent includes the option of authoring a character with a cartoon-style word balloon. Other settings enable you to determine how text appears in the balloon and when the balloon is removed. You can also determine what font to use. Although you can set a character’s word balloon attributes, be aware that the user can override these settings.

Be Efficient and Natural

When accomplishing tasks, effective human conversations are typically exchanges of brief information. Often, elements in the discussion are established between the parties and then referred to indirectly using abbreviated responses. These forms of abbreviation are beneficial

because they are efficient, and they also imply that the speaker and listener have a common context; that is, that they are communicating. Using appropriate forms of abbreviation also makes a dialogue more natural.

One form of conversational abbreviation is the use of contractions. When they are not used, they make a speaker seem more formal and rigid, and sometimes less human. Most human conversations demonstrate more freedom in the linguistic rules than written text.

Another common form of abbreviation in conversations is *anaphora*, the use of pronouns. For example, when someone asks, "Have you seen Bill today?" responses that substitute "him" for "Bill" are more natural than repeating the name again. The substitution is a cue that the parties in the dialogue share a common context of *who* "him" is. Keep in mind that the word "I" refers to the character when he or she says it.

Shared context is also communicated by the use of linguistic *ellipsis*, the truncation of many of the words in the original query. For example, the listener could respond, "Yes, I saw him," demonstrating the shared context of *when* or even respond with a simple "Yes" that demonstrates the shared context of *who* and *when*.

Implicit understanding can also be conveyed through other forms of abbreviated conversational style, where content is inferred without repetition, as shown in the following example:

User: I'd like a Chicago-style pizza.

Character: With "Extra Cheese"?

Similarly, if someone says, "It is hot in here," the phrase is understandable and requires no further detail if you know where the speaker is. However, if the context is not well established or is ambiguous, eliminating all contextual references may leave the user confused.

When using abbreviated communication, always consider the user's context and the type of content. It is appropriate to use longer descriptions for new and unfamiliar information. However, even with long descriptive information, try to break it up into smaller chunks. This gives you the ability to change the animation as the character speaks. It also provides greater opportunity for the user to interrupt the character, especially when using speech input.

Consistency is important in speech output. Strange speech patterns or prosody may be interpreted as downgrading the intelligence of the character. Similarly, switching between TTS and recorded speech may cause users to interpret the character as strange or possessing more than one personality. Lip-synced mouth movements can improve intelligibility of speech. Microsoft Agent automatically supports lip-syncing for TTS engines that comply with its required SAPI interfaces. However, lip-syncing is also supported for recorded speech. Sound files can also be enhanced with the Microsoft Linguistic Sound Editing Tool.

Use the Active Voice

When using speech output to provide directive information or to elicit a user response, use the active voice and clearly specify the user's expected action. The following example illustrates the differences:

Let me repeat your number.	No user action
The number will be repeated.	Passive voice, no user action
Listen while the number is repeated.	Passive voice
Listen to the repetition.	Best choice

In addition, construct your output to unfold the key information at the end of the phrase as shown in the following examples:

Instead of: "Is three the next digit?"	Use: "Is the next digit three?"
Instead of: "Click OK to begin."	Use: "To begin, click OK."
Instead of: "Say 'Done' to complete your order." 'Done.'"	Use: "To complete your order, say 'Done.'"

Use Appropriate Timing and Emphasis

Like all feedback, the effectiveness of speech output depends on timing and emphasis. A good deal of information can be communicated in the pace, volume, and pitch used when something is spoken. If you use a text-to-speech engine as your character's voice, most engines let you set the speed, pauses, pitch, and emphasis of words or phrases. You can use these attributes to indicate interest and understanding of the character as well as direct the user's attention or indicate your character's intent. For further information on how to set speech attributes, see *Speech Output Tags*. If you are using sound files as your character's output, consider these factors as well in your recorded audio.

Bibliography

- Ball, G. et al. "Life-Like Computer Characters: The Persona Project at Microsoft Research." *Software Agents*. Ed. Jeff Bradshaw. Cambridge, MA: MIT Press, 1996.
- Bates, J. "The Nature of Character in Interactive Worlds and the Oz." *Technical Report CMU-CS-92-200*. School of Computer Science, Carnegie Mellon University. Pittsburgh, PA. October 1992.
- Bates, J., Loyall, A., and Reilly, W. "An Architecture for Action, Emotion, and Social Behavior." In *Proceedings of the Fourth European Workshop on Modeling Autonomous Agents in a Multi-Agent World* (S. Martino al Cimino, Italy, 1992).
- Bates, J., Loyall, A., and Reilly, W. "Integrating Reactivity, Goals and Emotions in a Broad Agent." In *Proceedings of the 14th Annual Conference of the Cognitive Science Society* (Indiana, July 1992).

- Cassell, J. "Believable Communicating Agents: The Relationship Between Verbal and Nonverbal Behavior in Autonomous Communicating Agents." In *Siggraph '96 Course Notes*. Reading, MA: ACM Press, 1996.
- Foner, L. "What's an Agent, Anyway? A Sociological Case Study." *Agents Memo 93-01*. Agents Group. Cambridge, MA: MIT Media Lab, 1993.
- Grice, H. "Logic and Conversation." In *Syntax and Semantics 3: Speech Acts*. Ed. P. Cole and J. Morgan. New York: Academic Press, 1975.
- Herman, J. *Newstalk: A Speech Interface to a Personalized Information Agent*. MIT Master's Thesis, Program in Media Arts and Sciences. Cambridge, MA. June 1995.
- Koda, T. and Maes, P. "Agents with Faces: The Effects of Personification of Agents." In *Proceedings of HCI'96* (London, 1996).
- Lassiter, J. "Principles of Traditional Animation Applied to 3-D Computer Animation." In *Proceedings of SIGGRAPH '87*. 35-44. Reading, MA: ACM Press, 1987.
- Laurel, B. "Interface Agents: Metaphors with Character." *The Art of Human Computer Interface Design*. Reading, MA: Addison-Wesley, 1990.
- Maes, P. "Agents that Reduce Work Overload and Information Overload." In *Communications of the ACM*. 31-40. July 1994.
- Marx, M. *Toward Effective Conversational Messaging*. MIT Master's Thesis, Program in Media Arts and Sciences. Cambridge, MA.. June 1995.
- Nass, C., Steuer, J., and Tauber, E. "Computers are Social Actors." In *Proceedings of the CHI '94 Conference* (Boston, MA, April 1994). 72-77. Reading, MA: ACM Press, 1994.
- Nass, C. and Reeves, B. *The Media Equation: How People Treat Computers, Televisions, and New Media as Real People and Places*. Cambridge University Press, 1996.
- Negroponte, N. "Hospital Corners." *The Art of Human-Computer Interface Design*. Ed. Brenda Laurel. Reading, MA: Addison Wesley, 1990.
- Oren, T., et al. "Guides: Characterizing the Interface." *The Art of Human-Computer Interface Design*. Ed. Brenda Laurel. Reading, MA: Addison Wesley, 1990.
- Parke, F. and Waters, K. *Computer Facial Animation*. Wellesley: AK Peters, 1996.
- Perlin, K. and Goldberg, A. "Improv: A System for Scripting Interactive Actors in Virtual Worlds." In *Proceeding of SIGGRAPH 1995* (New Orleans, 1995).
- Schmandt, C. *Voice Communication with Computers: Conversational Systems*. New York: Van Nostrand Reinhold, 1994.
- Syrdal, A., Bennett, R., and Greenspan, S. *Applied Speech Technology*. Boca Raton: CRC Press, 1995.
- Thomas, F., and Johnson, O. *The Illusion of Life*. New York: Abbeville Press, 1981.
- Thorisson, K. "Dialogue Control in Social Interface Agents." *InterCHI Adjunct Proceedings* (Amsterdam, Holland, 1993). Reading, MA: ACM Press, 1993.

Creating Personalities for Synthetic Actors: Toward Autonomous Personality Agents. Ed. R. Trappl and Paolo Petta. New York: Springer-Verlag, 1997.

Yankelovich, N., Levow, G., and Marx, M. "Designing Speech Acts: Issues in Speech User Interfaces." In *Proceedings of CHI '95* (Denver, CO, May 1995). 369-376. Reading, MA: ACM Press, 1995.

Yankelovich, N. "How Do Users Know What to Say?" *Interactions*, Vol. III.6. 32-43. November/December, 1996.

The Windows Interface Guidelines for Software Design. Redmond, WA: Microsoft Press, 1995.