Creating Animation Sequences

Creating animation sequences which the agent character will perform is a key feature of MS Agent you will use. In this chapter, you will learn the basics of creating (choreographing) animation sequences. More advanced techniques are covered in later chapters.

WARNING

The spoken output and text balloon can be disabled by the user in the Advanced Character Options Window. If speech output is vital to your sequence, check the Agent Character **SRStatus** property and the Balloon Object **Enabled** property before running animation sequences.

Basic animation unit.

Figure 3.1 depicts the basic animation unit. Each unit contains an animation that is played, followed by text that is displayed and spoken.
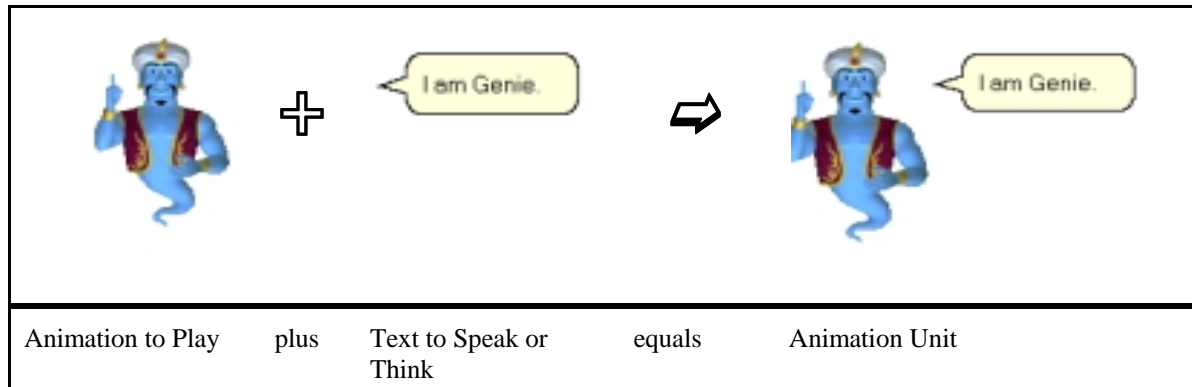
NOTE

There will be no spoken output if a compatible text-to-speech (TTS) engine is not installed.

There will be no balloon text output if the balloon is disabled.

Many of the animations available for the agent characters are speaking animations. A speaking animation will first display all the frames depicting the movement. The last frame is the speaking frame of the animation. The speaking frame contains the mouth overlays used during spoken output.

Animations cannot be performed while the agent character is speaking. If a speaking animation is following by a phrase to speak, then MS Agent plays the animation up to the speaking frame.

Figure 3.1 Basic Animation Unit

Animation to Play	plus	Text to Speak or Think	equals	Animation Unit

The text to be spoken is next displayed and spoken, while the mouth overlays corresponding to the speech are played.

NOTE

You cannot provide speech output with the looping animations (such as Processing, Reading, Searching, Writing). You must stop the looping animation before the agent character can speak.

When speak follows a non-speaking animation, the agent character returns to the RestPose position before the spoken output begins.

When choreographing your animation performance, be sure to think in animation sequence units: the agent character will do something, then say something You will string together the basic animation units in a Toolbook script handler to form an animation sequence. The MS Agent control will place the lines of code in a queue for the agent character, and execute each line after the preceding line is completed.

Choreographing an animation sequence.

Choreograph means to arrange or direct performance, process, and details. It is a term used in 3D animations to describe bringing together all the elements in the 3D design. It is also an appropriate term to describe the

methods you will use to create an animation sequence for the agent character to perform. Although not an official MS Agent term, it is used in this manual to describe your design process when creating your animation sequences for your agent characters.

How well you choreograph your animation sequences will determine how social your agent character is. It is beyond the scope of this manual to provide an in depth discussion of techniques to use to improve social interaction, but basic hints will be provided.

NOTE

Animation sequences consist of lines of code placed within a handler in the script of an object in your book. For clarity, the coding convention used by this manual is to place all animation sequences within their own handler. The name of the handlers used in this manual usually correspond to the purpose of the animation sequence. Other coding conventions are possible.

When choreographing your animation sequences, try to make the animation played match what the agent character is saying. Also try to have the agent character move after each short phrase. If you observe people talking, their bodies tend to be in motion while they speak. Your agent character will appear more life-like if you add motion.
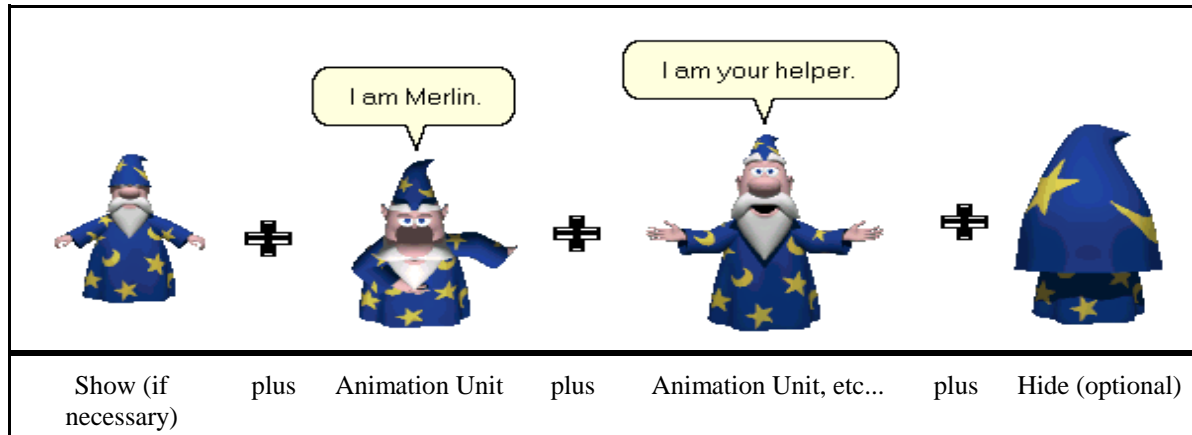
Figure 3.2 Animation Sequence

| Show (if necessary) | plus | Animation Unit | plus | Animation Unit, etc... | plus | Hide (optional) |

Figure 3.2 shows the components of an animation sequence. No animations can play unless the character is shown, so you must show the character first if you haven't previously shown the agent character during other animation sequences. You construct the animation sequence by choreographing basic animation units containing an animation to **Play**, coupled with text to **Speak** or **Think**. At the completion of your animation sequence, you will hide the agent character, if necessary.

✔

ASSUMPTION

You have already initialized the Agent control, and loaded the agent character(s) for which you will be choreographing animation sequences. Remember, you can only load the same agent character once. See Chapters 2 and 11 for how to load the agent characters.

Procedure 3.1: How to choreograph the first animation sequence.

An agent character must be shown before any animations will play or speech output will occur.

```
to handle Intro_Genie – user-defined handler
  system AgentGenie

  get extShow() of AgentGenie – shows the agent character
    -- "Genie"
  get extPlay("Greet") of AgentGenie – plays the animation
  -- "Greeting".
  --Note that the name of the animation is placed within ""
  get extSpeak("Salutations") of AgentGenie – Genie will say
  -- "Salutations."
  --Note that the text to be spoken is placed within ""
  get extPlay("Announce") of AgentGenie
  get extSpeak("I am Genie, your humble servant.") of AgentGenie

end Intro_Genie
```

The handler would be placed in the script of an object, such as a page, and called:

```
 to handle enterPage
   forward
   send Intro_Genie
 end enterPage
```

☞

NOTE

OpenScript statements used:
get extShow()
            the agent character appears.
get extPlay("Name of Animation")
            the agent character performs the
            named animation.
get extSpeak("Text to Speak")
            the agent character says the text to
            speak, and the text appears in a
            cartoon balloon near the agent
            character's head.

✎

TIP

When your user first encounters the agent character, you should provide a brief animation sequence introducing the user to the character and telling the user the purpose of the agent character.

You can make your agent character think words, too, not just speak them. You will use the **Think** method. It is similar to the **Speak** method, except that there is no audio output, the agent character's mouth does not move, and the text balloon is different. Figure 3.3 shows the difference in the text balloon. If a **extPlay** statement precedes the **extThink** statement, the animation freezes on the

speaking frame, then the text is displayed in the text balloon but there is no mouth movements by the agent character.
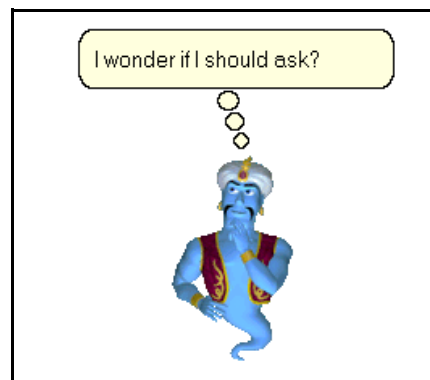
Figure 3.3. Think animation.

Procedure 3.2: Using think in an animation sequence.

```
to handle Explain2_Peedy– user-defined handler
 system AgentPeedy

 get extShow() of AgentPeedy – shows the agent character
 -- "Peedy", if necessary
 get extPlay("Explain") of AgentPeedy – plays the animation
 -- "Explain".
 -- Note that the name of the animation is placed within ""
 get extSpeak("You can type in your answer in any order.") of\
    AgentPeedy
 get extPlay("Confused") of AgentPeedy
 get extThink("What else was I supposed to tell you?") of\
AgentPeedy
 get extPlay("Announce") of AgentPeedy
 get extSpeak("Oh, don't forget to click on the score quiz button.")\
    of AgentPeedy
 end Explain2_Peedy
```

NOTE

OpenScript statement used:

get extThink("Text to Think")
>   the text appears in a cartoon balloon
>   near the agent character's head, but
>   the agent character does not say
>   anything.

TIP

If you are not going to have an animation sequence for awhile, it is a good idea to hide the agent character. The agent character goes into idle after a short period, and the noise and movement associated with the idle agent character can be quite distracting. Merlin snores!

TIP

When you have widely scattered animation sequences in your book, it is a good idea to start each animation sequence with the **Show** method. If you have left your agent character visible on the screen, the user might have hidden the agent character since the last time an animation played. Although you can use the agent character **Visible** property to see if the agent character is being shown, and then create a conditional statement to **Show** the agent character, it is not necessary. Playing **Show** if the agent character is currently visible will not raise an error - the Agent control will simply continue on to the next statement in the agent character's animation queue.

Procedure 3.3: How to hide an agent character once an animation sequence is completed.

```
to handle Explain_Purpose – user-defined handler
 system AgentMerlin

 get extShow() of AgentMerlin – shows the agent character "Merlin",
 -- if necessary
 get extPlay("GestureDown") of AgentMerlin – plays the animation
 -- "GestureDown".  Note that the name of the animation
 -- is placed within ""
 get extSpeak("This is the Summon Help button.") of AgentMerlin
 – Merlin will say "This is the Summon Help button".  Note that
 --the text to be spoken is placed within ""
 get extPlay("Explain") of AgentMerlin
 get extSpeak("I will appear and help you if you click the button.")\
    of AgentMerlin
 get extPlay("Pleased") of AgentMerlin
 get extSpeak("I will be most pleased to be of service to you.") of\
    AgentMerlin
 get extPlay("Wave") of AgentMerlin
 get extSpeak("Goodbye, for now.") of AgentMerlin
 get extHide() of AgentMerlin
end Explain_Purpose
```

The handler would be placed in the script of an object, such as a page, and called:

```
 to handle enterPage
   forward
   send Explain_Purpose
end enterPage
```

NOTE

If an agent character has been shown, then hidden, the agent character will reappear in the same location as when hidden unless you have moved it using the **MoveTo** method.

Trapping errors when an animation is not available for an agent character (Advanced Information).

If you are designing a book that will allow the user to select the agent character to be used, you must make provisions for animations that might not be in the user-selected agent character's list. If you are specifying each agent character to be used, you do not have to provide error trapping for missing animations, as long as you have tested the animation sequence before finalizing the book.

WARNING

Microsoft provides four MS Agent characters (Peedy, Merlin, Genie, and Robby). These agent characters have the same standard animation set of 59 animations, but also have additional animations. (See Appendix A for a list of the animations). They do not have all the same additional animations.

NOTE

If you don't know ahead of time which agent character will be selected by the user, it is a good idea to include an error trap to prevent an execution suspended error from being raised if you try to run an animation not in the agent character's animation list.

Procedure 3.4: Trapping errors raised when animation not available for agent character.

This code snipette replaces the missing animation with the RestPose animation (which all agent characters have)

```
sysError = NULL
sysSuspend = false
--the following will play if Merlin has the animation
-- "LookUpLeftBlink" in his list.  If not, an error will occur and be
-- handled 3 lines down.
get extPlay("LookUpLeftBlink") of AgentMerlin
sysSuspend = true
 if sysError<>NULL – an error has occurred
   get extPlay ("RestPose") of AgentMerlin
 end if
```

Since it would become tedious to use the code lines from Procedure 3.4 for each and every extPlay statement, you can create a function that will pass the extPlay where it is checked and executed.

Procedure 3.5: Function for trapping errors raised when animation not available for agent character.

The function is called by passing the agent character and animation to be played:

```
 send playAnimation "AgentMerlin", "LookUpLeftBlink" - you are
  --passing AgentMerlin and LookUpLeftBlink to the playAnimation
handler
```

The playAnimation handler:

```
 to handle playAnimation pChar, pAnimation
   --pChar is the name of the agent character passed
   --pAnimation is the name of the animation passed
```

```
system AgentGenie, AgentMerlin, AgentPeedy, AgentRobby

sysError = NULL –-clear system error so it is ready to accept
--any errors which might be raised below
sysSuspend = false –used so an error statement doesn't
--appear on screen
--the following will play if Merlin has the animation
-- "LookUpLeftBlink" in his list.  If not, an error will occur and be
--  handled 3 lines down.
 execute "get extPlay (" & QUOTE & pAnimation & QUOTE &")
     of "&&pChar
 sysSuspend = true –turn back on error showing now that you've
 --tried to run the animation
 if sysError<>NULL -- an error has occurred, so run the
 --animation RestPose (which every agent character has)
```

```
     execute "get extPlay ("& QUOTE & RestPose & QUOTE &")
        of"&& pChar
     end if
  end playAnimation
```

For Advanced Users:
To reduce the amount of typing you must do to
create animation sequences, you can create your
own functions to act as a wrapper for the various
agent character methods.

TIP

Exercise

For this exercise you will need the file "ToolbookAgentAssemblyPlatform.tbk".   Open the book.

Exercise 3.1 - Choreographing a basic animation sequence.

Press Esc key several times to stop Genie's introduction animation sequence. Although you are welcome to look at the code
for the opening animation sequence at this time, the handler contains advanced additional animation techniques covered in
other chapters of this manual. You may wish to wait until you have completed all the chapters before dissecting the code.

To make the agent character hide, so you can start the entire animation sequence over, right click the agent graphic object. Click on
the hide. Or, right click the agent character icon located on the task bar and click on hide.

NOTE

Procedure	Step-by-Step
1.　Show agent character Genie.　Add to script.	1a.　Click on the Action Combo Box. 1b.　Select Show. 1c.　Click on the Character Combo Box (the one on the left).

		1d. Select Genie.
		1e. Click on the Preview Animation button.
		1f. Add the code to the script:
		Click on Assemble Code button.
		Click on Add preview code to Script button.
2.	Play animation "Greet". Add to Script.	2a. Click on the Action Combo Box.
		2b. Select Play.
		2c. Make sure that Genie still shows in the Character Como Box.
		If not, click on the Combo Box and select Genie.
		2d. Click on the Animation Como Box.
		2e. Select the animation Greet.
		2f. Click on the Preview Animation button.
		2g. Add the code to the script:
		Click on Assemble Code button.
		Click on Add preview code to Script button.
3.	Have Genie speak "Salutations." Add to Script.	3a. Click on the Action Combo Box
		3b. Select Speak
		3c. Make sure that Genie still shows in the Character Como Box.
		If not, click on the Combo Box and select Genie.
		3d. Click in the text box and type `Salutations.` Do not type any "".
		3e. Click on the Preview Animation button.
		3f. Add the code to the script:
		Click on Assemble Code button.
		Click on Add preview code to Script button.
4.	Have Genie think "What else should I say?"	4a. Click on the Action Combo Box.
		4b. Select Think.
		4c. Make sure that Genie still shows in the Character Como Box.
		If not, click on the Combo Box and select Genie.
		4d. Click in the text box and type `What else should I say?` Do not type any "".
		4e. Click on the Preview Animation button.
		4f. Add the code to the script:
		Click on Assemble Code button.
		Click on Add preview code to Script button.

5. Copy your animation sequence script and Test it.	5a. Click on the Copy Script button. 5b. Click on the Test Button. 5c. Enjoy your animation sequence.
6. Now add to your animation sequence. Use steps 2, 3, 4, and 5 to add animation units to your script. Try playing different animations from the animation Combo Box. Be sure to preview each selection first. Do not click on the "Assemble code" button or the "Add preview code to Script" button until you are satisfied with your selections.	6. Follow steps, 2, 3 4, and 5.
7. End your animation sequence by hiding your agent character.	7a. Click on the Action Combo Box. 7b. Select Hide. 7c. Make sure that Genie still shows in the Character Combo Box. If not, click on the Combo Box and select Genie. 7d. Click on the Preview Animation button. 7e. Add the code to the script: Click on Assemble Code button. Click on Add preview code to Script button. 7f. Repeat step 5 to add this code to the Test Button script.
8. Save the script of your animation sequence.	8a. Right click on the "Test Script" button. Select the script. 8b. Click on file, then Export Script. 8c. Select the subdirectory you want, and type in the name of the file. Click OK. 8d. You can now use the import script option to bring this script into the script of any object in your book. If it is not a button, change the "to handle buttonClick" to the handler you will be using.

NOTE: It is a good idea to always preview the animation to be played before adding it to your script. That way you make sure that the selected animation is available for that agent character. If the animation is not available, an execution suspended error will occur.