บทที่ 2
หลักการและทฤษฎี

ในบทที่ 2 นี้จะเป็นการกล่าวถึงหลักการและทฤษฎีต่าง ๆ ที่นำมาใช้ในการพัฒนาโปรแกรมสำหรับโครงงานตัวต่อประลองปัญญาผ่านระบบเครือข่ายอินเทอร์เน็ต ได้แก่ ทฤษฎีของ DirectX ทฤษฎีของ Window API ทฤษฎีการใช้งานของ Winsocks Control และทฤษฎีของ Microsoft Agent Control

2.1 หลักการและทฤษฎีของ DirectX

ในอดีตนั้น เกมโปรแกรมเมอร์ทั้งหลายต่างพากันพัฒนาเกมในระบบดอสอย่างไม่หยุดยั้ง ซึ่งต่างก็ได้รับความลำบากไปตาม ๆ กันในการที่จะสร้างเกมต่าง ๆ มารองรับกับฮาร์ดแวร์ต่าง ๆ บนเครื่องคอมพิวเตอร์ของคุณ เพื่อที่จะทำงานได้ซึ่งนับวันยิ่งมีจำนวนมากขึ้นเรื่อย ๆ

ปัจจุบันจากสาเหตุข้างต้นและความนิยมของระบบปฏิบัติการแบบวินโดวส์ทำให้โปรแกรมเมอร์เกมทั้งหลายพากันย้ายมาสร้างเกมต่าง ๆ บนวินโดวส์ด้วยเทคโนโลยีที่เรียกว่า DirectX กันเป็นจำนวนมาก เนื่องจากการที่ไม่ต้องมาเขียนไดรเวอร์ของการ์ดต่างๆ ที่มีอยู่ในปัจจุบันให้กับคุณ แล้วการประมวลผลต่าง ๆ ก็ทำได้อย่างรวดเร็ว เนื่องจากขั้นตอนของการประมวลผลต่าง ๆ สามารถทำได้โดยตรง โดยไม่จำเป็นต้องผ่านระบบปฏิบัติการวินโดวส์ ส่วนไดรเวอร์ของการ์ดต่าง ๆ ก็ปล่อยให้เป็นหน้าที่ของผู้ผลิตไปจัดการกันเอง

<u>รู้จักกับ DirectX</u>

Microsoft DirectX SDK (Software Development Kit) นั้นเป็นเครื่องมือที่ใช้ในการพัฒนาซอฟต์แวร์ทางมัลติมิเดียของทางบริษัทไมโครซอฟท์นั่นเอง ซึ่งรวมถึงสามารถพัฒนาเกมคอมพิวเตอร์ได้อีกด้วย โดยเริ่มแรกนั้นเราจะเห็นเฉพาะเฮดเดอร์ไฟล์และไลบรารีไฟล์ของ Ddraw เท่านั้น หลังจากนั้นต่อมา งานด้าน 3 มิติเริ่มเข้ามามีบทบาทมากขึ้น ทางบริษัทไมโครซอฟท์จึงได้ทำการติดต่อกับทางบริษัทจากอังกฤษที่ชื่อว่าเรียลลิตี แล็บ (Reality Lab) เพื่อขอซื้อเรนเดอร์มอร์-ฟิกส์ (Rendermorphics) มาขัดการในงานกราฟิก 3 มิตินั่นเอง ทำให้ทางไมโครซอฟท์มีความคิดที่จะเพิ่มเติมในส่วนอื่น ๆ อีกเพื่อความสมบูรณ์ในตัว SDK จึงได้ออก Microsoft DirectX SDK เวอร์-

ชั่น 2 ออกมาเพื่อทำการแนะนำตลาด ซึ่งก็ได้รับการตอบรับจากโปรแกรมเมอร์ทั้งหลายเป็นอย่างดี ทำให้เกิดการพัฒนาเวอร์ชั่นใหม่ขึ้นเรื่อย ๆ อันได้แก่ Microsoft DirectX SDK เวอร์ชั่น 3, เวอร์ชั่น 5 และเวอร์ชั่น 6 (ล่าสุด) ทำให้มีอินเทอร์เฟสใหม่ ๆ ขึ้นมา รวมถึงการเพิ่มวิธีการบางอย่างเข้าไปอีก เพื่อตอบสนองการใช้งานให้มากที่สุดนั่นเอง

ส่วนประกอบของ DirectX

DirectX SDK เวอร์ชั่น 6 นั้นมีส่วนประกอบสำคัญ ๆ ดังนี้

1. DirectDraw
2. Direct3D
3. DirectSound
4. DirectPlay
5. DirectInput
6. DirectMusic
7. DirectSetup
8. AutoPlay

สำหรับในการพัฒนาโครงงานนี้ ผู้พัฒนาได้ใช้เฉพาะไดเรกซาวด์ (DirectSound) และไดเรกมิวสิก (DirectMusic) เข้ามาช่วยในการพัฒนาโครงงานเท่านั้น ดังนั้นทางผู้พัฒนาจึงขออธิบายวิธีการใช้เพียง 2 ส่วนนี้ ซึ่ง

ไดเรกซาวด์ (DirectSound) มีฟังก์ชันที่ช่วยให้โปรแกรมเมอร์เกมทำงานได้ง่ายขึ้น ในการใส่เสียงเพลงประกอบและเสียงเอฟเฟ็กต์ต่าง ๆ เข้าไปในเกม

ไดเรกมิวสิก (DirectMusic) มีลักษณะคล้าย ๆ กับไดเรกซาวด์นั่นเอง แต่จะมีฟังก์ชันที่คอยสนับสนุนการใช้งานไฟล์ MIDI ต่าง ๆ

ฟังก์ชั่นของไดเรกมิวสิก (DirectMusic)

- **DirectX7.DirectMusicLoaderCreate**

DirectX7.DirectMusicLoaderCreate นี้ ใช้สร้างออบเจ็กต์ DirectMusicLoader ซึ่งมีหน้าที่ในการเรียกไฟล์ที่เกี่ยวข้องขึ้นมาใช้งาน

การประกาศ

object. **DirectMusicLoaderCreate () As DirectMusicLoader**

พารามิเตอร์

object คือ ตัวแปรออบเจ็กต์ DirectX7

ค่าคืนกลับ

วิธีการนี้จะคืนค่าออบเจ็กต์ DirectMusicLoader

- **DirectX7.DirectMusicPerformanceCreate**

DirectX7.DirectMusicPerformanceCreate นี้ ใช้สร้างออบเจ็กต์ DirectMusicPerformance ซึ่งมีหน้าที่ในการจัดการเกี่ยวกับไฟล์นั้น ๆ ไม่ว่าจะเป็นการเล่น หรือหยุดเล่น

การประกาศ

object. **DirectMusicPerformanceCreate () As DirectMusicPerformance**

พารามิเตอร์

object คือ ตัวแปรออบเจ็กต์ DirectX7

ค่าคืนกลับ

วิธีการนี้จะคืนค่าออบเจ็กต์ DirectMusicPerformance

- **DirectMusicPerformance.Init**

DirectMusicPerformance.Init ใช้ในการกำหนดรูปแบบไฟล์ในตอนแรก และทำการเชื่อมออบ-เจ็กต์นี้เข้ากับออบเจ็กต์ DirectSound โดยวิธีการนี้ควรจะทำการเรียกเพียงครั้งเดียวเท่านั้น และเรียกก่อนที่วิธีการอื่น ๆ จะทำการเรียกใช้หรือจัดการกับไฟล์นั้น ๆ

การประกาศ

object.**Init** (DirectSound **As DirectSound**, hwnd **As Long**)

พารามิเตอร์

object	คือ ตัวแปรออบเจ็กต์ DirectMusicPerformance
DirectSound	คือ ตัวแปรที่เป็น DirectSound ที่ DirectMusic จะทำการสร้างขึ้นมา
hwnd	คือ ค่าแฮนเดิล (ซึ่งเป็นค่าตัวเลขที่กำหนดขึ้นโดยโปรแกรม Windows สำหรับใช้อ้างอิงถึงคอนเท็กซ์สำหรับการแสดงผลต่าง ๆ) ของฟอร์มหรือวินโดวส์ที่ต้องการติดตั้ง

- **DirectMusicPerformance.SetPort**

DirectMusicPerformance.SetPort นี้ เป็นการเซตค่าของพอร์ตในการที่จะนำมาใช้งาน

การประกาศ

object.**SetPort** (index **As Long**, numGroups **As Long**)

พารามิเตอร์

object	คือ ตัวแปรออบเจ็กต์ DirectMusicPerformance
index	คือ ค่าอินเด็กซ์ของพอร์ต ต้องอยู่ในช่วง 1 ถึง ค่าส่งกลับของ DirectMusicPerformance แต่ปกติแล้วจะกำหนดค่าไว้ที่ –1
numGroups	คือ จำนวนแชนเนลของพอร์ต

หมายเหตุ

ในแต่ละกลุ่มช่องสัญญาณจะประกอบด้วยช่องสัญญาณ 16 ช่อง

- **DirectMusicLoader.LoadSegment**

DirectMusicLoader.LoadSegment ทำหน้าที่โหลดเซกเมนต์จากไฟล์

การประกาศ

object.**LoadSegment** (filename **As String**) **As DirectMusicSegment**

พารามิเตอร์

object	คือ ตัวแปรออบเจ็กต์ DirectMusicLoader
filename	คือ ชื่อไฟล์ที่เราต้องการเรียกขึ้นมาใช้งาน

ค่าคืนกลับ

ถ้าวิธีการนี้ทำงานสำเร็จ มันจะคืนค่าออบเจ็กต์ DirectMusicSegment

- **DirectMusicSegment.SetStandardMidiFile**

DirectMusicSegment.SetStandardMidiFile นี้เป็นการแจ้งให้ DirectMusic ทราบว่า segment จะอยู่บนพื้นฐานของไฟล์ MIDI

การประกาศ

object.**SetStandardMidiFile ()**

พารามิเตอร์

object คือ ตัวแปรออบเจ็กต์ DirectMusicSegment

หมายเหตุ

วิธีการนี้ควรจะทำการเรียกใช้ ก่อนที่เครื่องมือ (instrument) จะถูกดาวน์โหลดเข้ามาใช้งาน

- **DirectMusicPerformance.SetMasterAutoDownload**

DirectMusicPerformance.SetMasterAutoDownload นี้ ทำหน้าที่ในการเปิดและปิดการ download เครื่องมือต่าง ๆ ที่ถูกนำมาใช้งานโดยอัตโนมัติ

การประกาศ

object.**SetMasterAutoDownload** (b **As Boolean**)

พารามิเตอร์

object คือ ตัวแปรออบเจ็กต์ DirectMusicPerformance

b คือ มีค่าเป็น Ture เมื่อเปิด autodownloading และมีค่าเป็น False เมื่อปิดมัน โดยค่าปกติจะเป็น False

- **DirectMusicSegment.Download**

DirectMusicSegment.Download นี้จะเป็น download เอาเครื่องมือต่าง ๆ ที่เกี่ยวข้องในการใช้งานเข้ามา

การประกาศ

object.**Download** (performance **As DirectMusicPerformance**)

พารามิเตอร์

object คือ ตัวแปรออบเจ็กต์ DirectMusicSegment

performance เป็นออบเจ็กต์ DirectMusicPerformance ที่ซึ่งใช้ในการ download เครื่องมือต่าง ๆ เข้ามา

- **DirectMusicPerformance.PlaySegment**

DirectMusicPerformance.PlaySegment นี้เป็นการเริ่มต้นการเล่นของเซกเมนต์

การประกาศ

object.**PlaySegment** (segment **As DirectMusicSegment,** lflags **As Long,** startTime **As _ Long) As DirectMusicSegmentState**

พารามิเตอร์

object คือ ตัวแปรออบเจ็กต์ DirectMusicPerformance

segment คือ เป็น DirectMusicSegment ที่จะเล่น

lFlags คือ ค่าที่เปลี่ยนไปตามการกระทำของวิธีนี้

startTime คือ เวลาที่เริ่มต้นการเล่นของ segment ถ้าถูกเซตค่าไว้ 0 อาจจะทำให้ segment เริ่มต้นการเล่นในไม่ช้าก็เป็นได้

ค่าคืนกลับ

ถ้าวิธีการทำงานสำเร็จ มันจะคืนค่าออบเจ็กต์ DirectMusicSegment

หมายเหตุ

segment ควรจะมีระยะเวลามากกว่า 250 มิลลิวินาที

- **DirectMusicPerformance.SetMasterTempo**

DirectMusicPerformance.SetMasterTempo นี้ จะตั้งค่าสเกลซึ่งถูกนำไปประยุกต์ใช้ใน tempo

การประกาศ

object.**SetMasterTempo** (tempo **As Single**)

พารามิเตอร์

object คือ ตัวแปรออบเจ็กต์ DirectMusicPerformance

tempo คือ master tempo ซึ่งมีค่าระหว่าง 0.25 ถึง 2.0

หมายเหตุ

โดยค่าปกติ master tempo จะเป็น 1 โดยค่า 0.5 เป็นครึ่งหนึ่งของ tempo และค่า 2.0 จะเป็นสองเท่าของ tempo

- **DirectMusicPerformance.GetMasterTempo**

DirectMusicPerformance.GetMasterTempoจะเป็นการคืนค่า mastertempo ในปัจจุบัน

การประกาศ

object.**GetMasterTempo () As Single**

พารามิเตอร์

object คือ ตัวแปรออบเจ็กต์ DirectMusicPerformance

ค่าคืนกลับ

ถ้าวิธีการนี้ทำงานสำเร็จ มันจะคืนค่าในช่วง 0.25 ถึง 2.0

รหัสข้อผิดผลาด

ถ้าวิธีการนี้ทำงานล้มเหลว มันจะแสดงข้อผิดพลาดนี้และไปเซตค่าของ Err.Number

- **DirectMusicSegment.SetRepeats**

DirectMusicSegment.SetRepeats เป็นการเซตจำนวนเวลาการวนซ้ำของ segment

การประกาศ

object.**SetRepeats** (lRepeats **As Long**)

พารามิเตอร์

object คือ ตัวแปรออบเจ็กต์ DirectMusicSegment

lRepeats คือ จำนวนของการวนซ้ำ

- **DirectMusicSegment.GetRepeats**

DirectMusicSegment.GetRepeatsเป็นการส่งค่าคืนกลับของจำนวนเวลาการวนซ้ำของ segment

การประกาศ

object.**GetRepeats** () **As Long**

พารามิเตอร์

object คือ ตัวแปรออบเจ็กต์ DirectMusicSegment

ค่าคืนกลับ

ถ้าวิธีการนี้ทำงานสำเร็จ มันจะคืนค่าของเวลาของการวนซ้ำ

- **DirectMusicPerformance.Stop**

DirectMusicPerformance.Stop เป็นวิธีหยุดการเล่นของ segment

การประกาศ

object.**Stop** (segment **As DirectMusicSegment,** segmentState **As DirectMusicSegmentState,** mtTime **As Long,** lFlag **As Long**)

พารามิเตอร์

object	คือ ตัวแปรออบเจ็กต์ DirectMusicPerformance
segment	คือ ตัวแปรที่เป็น DirectMusicSegment ที่จะหยุดการเล่น
segmentState	คือ ตัวแปรที่เป็น DirectMusicSegmentState ที่จะใช้แสดงแทนในกรณีที่ segment หยุดการเล่น
mtTime	คือ Music time ที่หยุดการเล่นของ segment หรือ segment state หรือหยุดการเล่นของทั้งคู่

lFlags คือ Flag ที่ใช้แสดงเมื่อมีการหยุดเกิดขึ้น ถ้า lFlags มีค่าเป็น 0 จะหยุดการเล่นทันที

- **DirectMusicSegment.Unload**

DirectMusicSegment นี้ใช้ในการ unload เครื่องมือต่าง ๆ ที่นำมาใช้อยู่ก่อนแล้วด้วยวิธีการ DirectMusicSegment.Download

การประกาศ

object.**Unload** (performance As **DirectMusicPerformance**)

พารามิเตอร์

object คือ ตัวแปรออบเจ็กต์ DirectMusicSegment

performance คือ ตัวแปร DirectMusicPerrformance ที่ซึ่งจัดการ unload เครื่องมือต่างที่นำมาใช้

ฟังก์ชั่นของ DirectSound

- **DirectX7.DirectSoundCreate**

DirectX7.DirectSoundCreate นี้เป็นการสร้างออบเจ็กต์ DirectSound ขึ้นมาใช้งาน

การประกาศ

Object.**DirectSoundCreate** (guid **as string**) **as DirectSound**

พารามิเตอร์

object คือ ตัวแปรออบเจ็กต์ DirectX7

guid คือ เป็นตัวแปรที่ใช้ระบุถึงอุปกรณ์ที่ใช้ในการเล่น Sound ค่านี้ถ้าไม่ระบุก็จะเป็นการใช้อุปกรณ์ที่ได้เซตมาให้เรียบร้อยแล้ว (default device)

ค่าคืนกลับ

ถ้าทำงานเป็นผลสำเร็จ มันจะคืนค่าออบเจ็กต์ DirectSound

- **DirectSound.SetCooperativeLevel**

DirectSound.SetCooperativeLevel นี้ จะใช้เซตระดับค่าความร่วมมือของ Sound Device

การประกาศ

Object.**SetCooperativeLevel** (hwnd **as Long**,level **as As CONST_DSSCLFLAGS**)

พารามิเตอร์

object	คือ ตัวแปรออบเจ็กต์ DirectSound
hwnd	คือ ค่าแฮนเดิล (ค่าตัวเลขที่กำหนดขึ้นโดยโปรแกรม Windows สำหรับใช้อ้างอิงถึงคอนเท็กซ์สำหรับการแสดงผลต่าง ๆ)
level	คือ ระดับความสำคัญของการร้องขอ

หมายเหตุ

โปรแกรมประยุกต์จำเป็นต้องมีการเซตค่าระดับความร่วมมือโดยการเรียกใช้วิธีการนี้ก่อนที่จะทำการเล่น

- **DirectSound.CreateSoundBufferFromFile**

DirectSound.CreateSoundBufferFromFile นี้จะสร้างออบเจ็กต์ DerectSoundBuffer ที่ใช้ยกระดับของตัวอย่างเสียง

การประกาศ

object.**CreateSoundBufferFromFile** (filename **As String, bufferDesc As DSBUFFERDESC,** format **As WAVEFORMATERX) As DirectSoundBuffer**

พารามิเตอร์

object	คือ ตัวแปรออบเจ็กต์ DirectSound
filename	คือ ชื่อไฟล์ของ wave file ที่ถูกโหลดเข้ามาในบัฟเฟอร์
bufferDesc	คือ เป็นประเภท DSBUFFERDESC ที่บรรจุคำพรรณนาของเสียงที่สร้างขึ้น
format	คือ เป็นประเภท WAVEFORATEX ที่อธิบายรูปแบบของข้อมูล waveform-audio

ค่าคืนกลับ

ถ้าวิธีการนี้ทำงานสำเร็จจะมีค่าคืนกลับเป็น DirectSoundBuffer

- **DirectSoundBuffer.SetCurrentPosition**

DirectSoundBuffer.SetCurrentPosition นี้ จะเป็นการเคลื่อนย้ายตำแหน่งในการเล่นปัจจุบันไปเป็น sound buffer ที่สอง

การประกาศ

object.**SetCurrentPosition** (newPosition **As Along**)

พารามิเตอร์

object คือ ตัวแปรออบเจ็กต์ DirectSoundBuffer

newPosition คือ ตำแหน่งใหม่ในไบต์ที่จะถูกใช้เมื่อมีการเล่นใน sound buffer

หมายเหตุ

วิธีการนี้ไม่สามารถเรียกใช้ได้หากอยู่บน sound buffer ที่หนึ่ง

- **DirectSoundBuffer.Play**

DirectSoundBuffer.Play นี้ เป็นการเล่นซึ่งจะเริ่มจากตำแหน่งปัจจุบันใน sound buffer

การประกาศ

object.**Play**(flags **As CONST_DSBPLAYFLAGS**)

พารามิเตอร์

object	คือ ตัวแปรออบเจ็กต์ DirectSoundBuffer
flags	คือ เป็นการระบุว่าจะเล่นไฟล์ในบัฟเฟอร์อย่างไร
DSBPLAY_LOOPING	คือ เมื่อเล่นไปจนถึงจุดสุดท้ายในบัฟเฟอร์แล้ว ก็จะทำการเริ่มเล่นใหม่ที่จุดเริ่มต้นในบัฟเฟอร์และเล่นไปเรื่อย ๆ จนกระทั่งมีการสั่งให้หยุดเล่น

ค่าความผิดพลาด

ถ้าทำงานผิดพลาด ความผิดพลาดจะถูกแสดงและก็จะเซตค่าความผิดพลาดที่ Err.Number

2.2 หลักการและทฤษฎีของ Windows API

รู้จักกับ Windows API

ฟังก์ชันวินโดวส์ API ของวินโดวส์ ในความเป็นจริงก็หมายถึง ฟังก์ชันทั้งหมดที่อยู่ในไฟล์ไลบรารี .dll ของวินโดวส์ เช่น User32.dll, Gdi32.dll หรือ Kernel32.dll เป็นต้น ซึ่งไฟล์ไลบรารี .dll เหล่านี้ จะเป็นแกนหลักของระบบปฏิบัติการวินโดวส์ เพราะการทำงานของวินโดวส์จะอาศัยความสามารถของฟังก์ชันที่ปรากฏในไฟล์ไลบรารี .dll

ไฟล์ไลบรารี .dll ของวินโดวส์ 32 บิต

ไฟล์ไลบรารี .dll หรือที่เรียกว่า Dynamic Link Library เป็นไฟล์ที่โดยปกติจะถูกกำหนดให้มีนามสกุล .dll และจะถูกจัดเก็บเอาไว้ในไดเรกทรอรีย่อย \System ของวินโดวส์ และในความเป็นจริงการทำงานของวินโดวส์ก็ต้องอาศัยฟังก์ชันวินโดวส์ API ที่มีในแต่ละไฟล์ไลบรารี .dll นั่น

เอง ในที่นี้ เราจะกล่าวเฉพาะไฟล์ Gdi32.dll โดยไฟล์ Gdi32.dll เป็นไลบรารีสำหรับการติดต่อสื่อสารกับดีไวซ์กราฟิก (Graphic Device Interface) ซึ่งประกอบด้วยฟังก์ชันต่าง ๆ สำหรับงานด้านอุปกรณ์แสดงผล การวาดกราฟิก ไฟล์เมต้า (.wmf) ฟอนต์ และการกำหนดตำแหน่งโคออร์ดิเนต

โดยปกติไฟล์ไลบรารีไดนามิกส์ของวินโดวส์จะมีนามสกุล .dll แต่สำหรับ Visual Basic 6.0 ก็สนับสนุนไฟล์ไลบรารีชนิดไดนามิกส์เช่นกัน แต่จะมีรูปแบบของไฟล์ที่แตกต่างไปจากไฟล์ไลบรารีไดนามิกส์มาตรฐาน และจะถูกกำหนดให้มีนามสกุล .ocx

Visual Basic 6.0 สำหรับการเขียนโปรแกรม

Visual Basic 6.0 เป็นภาษาหนึ่งที่มีความใกล้ชิดกับวินโดวส์มากที่เดียว เพราะความสามารถต่าง ๆ และคำสั่งต่าง ๆ ส่วนใหญ่จะเข้ากันได้กับความสามารถของวินโดวส์โดยตรง ซึ่งในทางปฏิบัติก็หมายความว่า Visual Basic 6.0 ได้ดึงเอาความสามารถของวินโดวส์ มาเป็นส่วนหนึ่งของความสามารถของตัวแปลภาษา เพื่อช่วยให้การสร้างโปรแกรมประยุกต์ที่สามารถใช้ประโยชน์จากความสามารถของวินโดวส์ได้อย่างเต็มที่

การประกาศฟังก์ชันวินโดวส์ API

ก่อนที่เราจะสามารถใช้งานฟังก์ชันวินโดวส์ API ในการเขียนโปรแกรม เราจะต้องทำการประกาศฟังก์ชันนั้น ๆ เสียก่อน เพื่อเป็นการกำหนดโปรโตไทป์ (prototype) ในการติดต่อระหว่างฟังก์ชันวินโดวส์ API กับ Visual Basic 6.0 สำหรับการประกาศฟังก์ชันวินโดวส์ API ซึ่งมีอยู่ด้วยกัน 2 รูปแบบ คือ การประกาศโพรซีเยอร์ประเภทรูทีนย่อย (Sub) และ การประกาศโพรซีเยอร์ประเภทฟังก์ชัน (Function) ดังนี้

<u>รูปแบบที่ 1</u> การประกาศโพรซีเยอร์ประเภทรูทีนย่อย

[Public / Private] Declare Sub name **Lib** "libname" [**Alias** "aliasname"][([arglist])]

<u>รูปแบบที่ 2</u> การประกาศโพรซีเยอร์ประเภทฟังก์ชัน

[Public / Private] Declare Function name **Lib** "libname" [**Alias** "aliasname"][([arglist])]
[As type]

Public กำหนดให้โพรซีเยอร์ที่ถูกประกาศสามารถถูกเรียกใช้ได้จากโมดูลหรือฟอร์มอื่น ๆ ในโปรเจ็กต์ปัจจุบัน แต่การประกาศโพรซีเยอร์ด้วย Public จะต้องกระทำในโมดูล

มาตรฐาน (.bas) เท่านั้น

Private กำหนดให้โพรซิเยอร์ที่ถูกประกาศ สามารถถูกเรียกใช้ได้ เฉพาะโมดูลหรือฟอร์มที่ประกาศโพรซิเยอร์นี้เท่านั้น

Sub กำหนดให้โพรซิเยอร์เป็นชนิดรูทีน ซึ่งจะไม่มีการส่งค่ากลับ

Function กำหนดให้โพรซิเยอร์เป็นชนิดฟังก์ชัน ซึ่งจะมีการส่งค่ากลับ ดังนั้น เราจึงสามารถนำค่าที่ได้จากฟังก์ชัน ไปใช้ในลักษณะของตัวแปรหนึ่ง ๆ ในนิพจน์ (expression) เช่น นิพจน์การคำนวณ

name กำหนดชื่อของโพรซิเยอร์ สำหรับ Visual Basic 6.0 ยอมให้ชื่อของโพรซิเยอร์นั้นมีความยาวทั้งหมด 40 ตัวอักษรเท่านั้น

Lib กำหนดให้สตริงที่ตามมา ("libname") จะบ่งบอกถึงชื่อของไฟล์ไลบรารี .dll ซึ่งในการประกาศฟังก์ชันวินโดวส์ API จะต้องมีการกำหนดชื่อของไฟล์ไลบรารี .dll ทุกครั้ง

Alias กำหนดให้โพรซิเยอร์ชื่อ "aliasname" สามารถถูกเรียกใช้ได้ โดยใช้ชื่ออื่น ๆ ตามที่กำหนดโดย name ทั้งนี้เนื่องจาก ในบางกรณีชื่อของฟังก์ชันวินโดวส์ API นั้น อาจจะซ้ำกับประโยคหรือคำสั่งของ Visual Basic 6.0 หรืออาจจะประกอบด้วยตัวอักษรที่ Visual Basic 6.0 ไม่ยอมรับในการกำหนดเป็นชื่อของโพรซิเยอร์ ดังนั้นในกรณีนี้ เราจะใช้ส่วนของ Alias "aliasname" ในการกำหนดให้โพรซิเยอร์ดังกล่าวสามารถถูกเรียกใช้ได้โดยใช้ชื่ออื่น ๆ ตามที่กำหนดโดย name

arglist รายการของพารามิเตอร์ที่จะส่งให้กับโพรซิเยอร์

type การกำหนดชนิดของข้อมูลสำหรับพารามิเตอร์หรือโพรซิเยอร์ที่เป็นฟังก์ชัน ซึ่งสามารถกำหนดได้หลายชนิดด้วยกัน เช่น Byte, Boolean, Integer, Long, Currency, Single,Double, Date, String, Object หรือ Variant เป็นต้น โดยที่รูปแบบของการกำหนดพารามิเตอร์ มีดังต่อไปนี้

[ByVal / ByRef][ParamArray] varname[()]**[As type]**

ByVal กำหนดให้พารามิเตอร์มีการส่งค่าแบบ passed by value แต่ถ้าหากนำไปใช้ประกาศฟังก์ชันวินโดวส์ API ก็จะมีความหมายตามแต่ชนิดข้อมูลของพารามิเตอร์

ByRef กำหนดให้พารามิเตอร์มีการส่งค่าแบบ passed by reference

ParamArray ใช้กำหนดให้พารามิเตอร์ดังกล่าวเป็นอาเรย์ของข้อมูลชนิด variant ซึ่งจะต้องเป็นพารามิเตอร์ตัวสุดท้ายของโพรซิเยอร์เท่านั้น โดยทั่วไปส่วนนี้มักจะไม่สามารถใช้

ได้กับฟังก์ชันวินโดวส์ API มาตรฐาน ยกเว้นฟังก์ชันถูกออกแบบเฉพาะ Visual Basic 6.0 เท่านั้น

การใช้งาน Device Context ของวินโดวส์

ในส่วนนี้จะกล่าวถึงออบเจ็กต์ของวินโดวส์ที่มีความสำคัญในด้านของการแสดงผล เนื่องจากวินโดวส์มีการแสดงผลทุกอย่างในรูปแบบของกราฟิก ดังนั้น วินโดวส์จึงต้องมีการออกแบบส่วนที่จะช่วยในการแสดงผลกราฟิกอย่างมีประสิทธิภาพ และสามารถใช้งานได้กับฮาร์ดแวร์ที่มีความละเอียดแตกต่างกันได้ โดยไม่จำเป็นต้องมีการเขียนโค้ดเฉพาะสำหรับฮาร์ดแวร์ชนิดเดียวกันแต่มีความละเอียดแตกต่างกัน ซึ่งออบเจ็กต์ของวินโดวส์ดังกล่าวถูกเรียกว่า Device Context (DC) ซึ่งการเขียนโปรแกรมด้วย Visual Basic 6.0 เราสามารถอาศัยฟังก์ชันวินโดวส์ API ในกลุ่มของ DC เพื่อช่วยในการเพิ่มความเร็วหรือเพิ่มศักยภาพด้านกราฟิก และสามารถที่จะเข้าถึง DC ทั้งหมดที่วินโดวส์สนับสนุนได้โดยไม่มีขีดจำกัด

การวาดรูปหรือแสดงผลตัวอักษรใด ๆ ก็ตามภายใต้ระบบปฏิบัติการวินโดวส์ จะต้องถูกกระทำผ่านทาง Device Context (DC) ซึ่งเป็นออบเจ็กต์ตัวหนึ่งของวินโดวส์ที่ช่วยในการจัดการในด้านการจัดการกราฟิก เช่น การจัดขนาด การกำหนดขอบเขตพื้นที่การแสดงผล การกำหนดขนาดของเส้นหรือกราฟิก เป็นต้น เพื่อให้การเขียนโค้ดด้านกราฟิกสำหรับวินโดวส์มีความสะดวกมากขึ้น เนื่องจากวินโดวส์สนับสนุนอุปกรณ์แสดงผล (output device) ที่มีความละเอียดแตกต่างกัน ดังนั้น ถ้าหากไม่มีการออกแบบส่วนหรือองค์ประกอบของวินโดวส์ ที่จะช่วยในการแปลงระบบโคออร์ดิเนต (ขนาดความกว้าง หรือ ความยาวของอุปกรณ์แสดงผล) ให้อย่างอัตโนมัติ ก็จะทำให้การเขียนโค้ดมีความยุ่งยากมากขึ้น

การสร้างและใช้งาน Device Context

วินโดวส์ได้เตรียม DC มาตรฐานอยู่ภายใน (build-in) หน่วยความจำที่เรียกว่า pool เพื่อให้สามารถใช้ในการวาดกราฟิกลงในพื้นที่แสดงผลของหน้าต่างหรือดีไวซ์ต่าง ๆ ได้ โดยถ้าหากเราต้องการวาดกราฟิกลงในพื้นที่ของหน้าต่างหรือดีไวซ์ เราก็ต้องอ้างอิงถึง DC ตัวใดตัวหนึ่งจาก pool และกำหนดค่าแอตทริบิวส์ตามที่ต้องการ เช่น ขนาดหรือสีของเส้น เป็นต้น ให้กับ DC จากนั้นจึงเรียกใช้ฟังก์ชันวินโดวส์ API ด้านกราฟิกเพื่อวาดกราฟิก และเมื่อเราไม่ต้องการใช้ DC อีกต่อไป ก็ต้องทำการยกเลิกการถือครอง DC ดังกล่าว เพื่อให้ DC ที่ถูกยกเลิกการถือครอง ได้ถูกส่งกลับไปยัง pool และเป็น DC ที่ว่างสำหรับถูกเรียกใช้งานโดยโปรแกรมประยุกต์ตัวอื่น ๆ ต่อไป และนอกจากนี้ถ้าหากเราไม่พอใจกับ DC มาตรฐานที่วินโดวส์ได้เตรียมไว้ให้ เราก็สามารถที่จะสร้าง DC ขึ้นมา

ใหม่ ให้เข้ากันได้กับ DC มาตรฐาน เพื่อนำมาใช้ส่วนตัวสำหรับแต่ละโปรแกรมประยุกต์หนึ่ง ๆ ก็ได้เช่นกัน

ออบเจ็กต์ DC ไม่ว่าจะเป็น DC มาตรฐานหรือที่ถูกสร้างขึ้นมาโดยฟังก์ชันวินโดวส์ CreateDC ก็ตาม จะต้องมีหมายเลข handle ประจำตัวเสมอ เพื่อให้สามารถใช้กำหนดการเรียกใช้ DC โดยฟังก์ชันวินโดวส์ API ได้อย่างถูกต้อง สำหรับตัวของออบเจ็กต์ DC ก็เป็นข้อมูลโครงสร้างภายในของวินโดวส์ที่มีขนาดประมาณ 800 ไบต์ โดยที่วินโดวส์ได้เตรียมวิธีการที่หลากหลายในการสร้าง และ เข้าถึง DC โดยขึ้นกับลักษณะของ DC ดังนี้

- Private Device Context
- Cached Device Context
- Created Device Context

ในที่นี้ เราจะกล่าวเพียง Created Device Context เท่านั้น Created Device Context เป็น DC ที่ถูกสร้างขึ้นมาใหม่โดยโค้ดของโปรแกรม ซึ่งเราสามารถสร้าง DC สำหรับโปรแกรมประยุกต์หนึ่งๆ ได้โดยใช้ฟังก์ชันวินโดวส์ CreatDC หรือ CreatCompatibleDC โดยในทางทฤษฎีเราสามารถสร้าง DC ประเภทนี้ได้มากเท่าที่ต้องการหรือไม่มีข้อจำกัด แต่ในทางปฏิบัตินั้นจำนวนของ Created Device Context จะถูกจำกัดโดยทรัพยากรของระบบ โดยเฉพาะภายใต้วินโดวส์ 95 และ 98 ที่ยังคงมีข้อจำกัดในด้านของการจัดการทรัพยากรของระบบ สำหรับหมายเลข handle ของ Created Device Context จะถูกรายงานโดยฟังก์ชันวินโดวส์ CreatDC หรือ CreatCompatibleDC เมื่อการสร้าง Created Device Context ประสบผลสำเร็จ

การสร้าง DC คอมแพททิเบิลกับฟอร์มหรือคอนโทรล PictureBox

การสร้าง DC ที่คอมแพททิเบิลกับฟอร์มหรือคอนโทรล PictureBox ของ Visual Basic 6.0 โดยฟังก์ชันวินโดวส์ CreateCompatibleDC เพื่อทำการแสดงผลบิตแมปและไอคอนลงในคอนโทรล PictureBox หรือฟอร์ม ซึ่งจะมีฟังก์ชันวินโดวส์ API ที่เกี่ยวข้องดังนี้

BitBlt

การประกาศ

```
Declare Function BitBlt Lib "gdi32" Alias "BitBlt" (Byval
hDestDC As Long, ByVal x As Long, ByVal y As Long, ByVal
nWidth As Long, ByVal nHeight As Long, ByVal hSrcDC As
Long, ByVal xSrc As Long, ByVal ySrc As Long, ByVal dwRop As
Long)As Long
```

ลักษณะการทำงาน ก๊อบปี้บิตแมปจากดีไวซ์คอนเท็กซ์ (DC) หนึ่งไปยังดีไวซ์คอนเท็กซ์ (DC) อื่น ๆ โดยดีไวซ์คอนเท็กซ์ทั้งคู่จะต้องคอมแพทิเบิลกัน

อาร์กิวเมนต์

hDestDC	ข้อมูลชนิด Long – ค่าแฮนเดิล (ค่าตัวเลขที่กำหนดขึ้นโดยโปรแกรม Windows สำหรับใช้อ้างอิงถึงคอนเท็กซ์สำหรับการแสดงผลต่าง ๆ) ของดีไวซ์คอนเท็กซ์เป้าหมาย
X,Y	ข้อมูลชนิด Long – ตำแหน่งมุมบนซ้ายที่ต้องการจะวาดภาพบนดีไวซ์คอนเท็กซ์เป้าหมาย
Nwidth,nHeight	ข้อมูลชนิด Long – ความกว้างและความสูงของพื้นที่ในการวาดที่จะทำลงบนดีไวซ์คอนเท็กซ์เป้าหมาย
HSrcDC	ข้อมูลชนิด Long – ค่าแฮนเดิล (ค่าตัวเลขที่กำหนดขึ้นโดยโปรแกรม Windows สำหรับใช้อ้างอิงถึงคอนเท็กซ์ สำหรับการแสดงผลต่าง ๆ) ของดีไวซ์คอนเท็กซ์แหล่งกำเนิด
XSrc,ySrc	ข้อมูลชนิด Long – ตำแหน่งมุมบนซ้ายที่ต้องการจะวาดภาพบนดีไวซ์คอนเท็กซ์แหล่งกำเนิด
DwROP	ข้อมูลชนิด Long - ค่า Raster Operations ในการถ่ายเทข้อมูล

ค่าคืนกลับ ข้อมูลชนิด Long - หากฟังก์ชันทำงานสำเร็จจะคืนค่าที่ไม่ใช่ 0 แต่หากเกิดข้อผิดพลาดจะคืนค่า 0

CreateCompatibleDC

การประกาศ Declare Function CreateCompatibleDC Lib "gdi32" Alias "CreateCompatibleDC" (ByVal hdc As Long) As Long

ลักษณะการทำงาน สร้างดีไวซ์คอนเท็กซ์ในหน่วยความจำ โดยดีไวซ์คอนเท็กซ์ที่สร้างขึ้นจะคอมแพทิเบิลกับดีไวซ์ที่กำหนด

อาร์กิวเมนต์

hDC	ข้อมูลชนิด Long – ค่าแฮนเดิล (ค่าตัวเลขที่ถูกกำหนดขึ้นโดยโปรแกรม Windows สำหรับใช้อ้างอิงถึงคอนเท็กซ์สำหรับการแสดงผลต่าง ๆ) ของดีไวซ์คอนเท็กซ์ที่ต้องการให้ดีไวซ์คอนเท็กซ์ที่สร้างขึ้นมามีความ

คอมแพททิเบิลด้วยหากส่งค่า 0 จะหมายถึงดีไวซ์คอนเท็กซ์ของหน้าจอ

ค่าคืนกลับ ข้อมูลชนิด Long – หากฟังก์ชันทำงานสำเร็จ จะคืนค่าแฮนเดิล (ค่าตัวเลขที่กำหนดขึ้นโดยโปรแกรม Windows สำหรับใช้อ้างอิงถึงคอนเท็กซ์สำหรับการแสดงผลต่าง ๆ) ของดีไวซ์คอนเท็กซ์ในหน่วยความจำที่สร้างขึ้นใหม่และจะคืนค่า 0 ถ้าเกิดความผิดพลาด

หมายเหตุ ก่อนที่จะทำการวาดภาพใด ๆ ลงบนดีไวซ์คอนเท็กซ์ จะต้องทำการเลือกข้อมูลเข้าสู่ดีไวซ์คอนเท็กซ์เสียก่อน และควรจะลบดีไวซ์ที่สร้างขึ้นทิ้งเมื่อไม่จำเป็นต้องใช้งานแล้ว

DeleteDC

การประกาศ Declare Function DeleteDC Lib "gdi32" Alias "DeleteDC" (ByVal hdc AS Long) As Long

ลักษณะการทำงาน ขัดการลบดีไวซ์คอนเท็กซ์ตามค่าแฮนเดิล (ค่าตัวเลขที่กำหนดขึ้นโดยโปรแกรม Windows สำหรับใช้อ้างอิงถึงคอนเท็กซ์สำหรับการแสดงผลต่าง ๆ) ที่ส่งเป็นอาร์กิวเมนต์ ห้ามใช้กับ ดีไวซ์คอนเท็กซ์ที่ได้จากฟังก์ชัน GetDC()

อาร์กิวเมนต์ hDC ข้อมูลชนิด Long –ค่าแฮนเดิล (ค่าตัวเลขที่กำหนดขึ้นโดยโปรแกรม Windows สำหรับใช้อ้างอิงถึงคอนเท็กซ์สำหรับการแสดงผลต่าง ๆ) ของดีไวซ์คอนเท็กซ์ที่ต้องการจะลบออก

ค่าคืนกลับ ข้อมูลชนิด Long - หากฟังก์ชันทำงานสำเร็จจะคืนค่า 1 หากเกิดความผิดพลาดจะคืนค่า 0

SelectObject

การประกาศ Declare Function SelectObject Lib "gdi32" Alias "SelectObject" (ByVal hdc As Long, ByVal hObject As Long) As Long

ลักษณะการทำงาน โดยปกติแล้วจะมี GDI Object อย่างเช่น pen, brush, font, regions, palletes หรือ bitmap อย่างใดอย่างหนึ่งถูกเลือกเข้าสู่ดีไวซ์คอนเท็กซ์

เสมอ ซึ่งออบเจ็กต์แต่ละชนิดจะสามารถถูกเลือกเข้าสู่ดีไวซ์คอนเท็กซ์ได้เพียงชนิดละ 1 ออบเจ็กต์ เพื่อใช้สำหรับการวาดภาพบนดีไวซ์คอนเท็กซ์

อาร์กิวเมนต์	hDC	ข้อมูลชนิด Long –ค่าแฮนเดิล (ซึ่งเป็นค่าตัวเลขที่กำหนดขึ้นโดยโปรแกรม Windows สำหรับใช้อ้างอิงถึงคอนเท็กซ์สำหรับการแสดงผลต่าง ๆ) ของดีไวซ์คอนเท็กซ์ ที่ต้องการจะเลือกออบเจ็กต์เข้าสู่ดีไวซ์คอนเท็กซ์
	hObject	ข้อมูลชนิด Long –ค่าแฮนเดิล (ซึ่งเป็นค่าตัวเลขที่กำหนดขึ้นโดยโปรแกรม Windows สำหรับใช้อ้างอิงถึงคอนเท็กซ์สำหรับการแสดงผลต่าง ๆ) ของ GDI Object ที่จะเลือก
ค่าคืนกลับ	ข้อมูลชนิด Long – หากฟังก์ชันทำงานสำเร็จจะคืนค่าแฮนเดิล (ซึ่งเป็นค่าตัวเลขที่กำหนดขึ้นโดยโปรแกรม Windows สำหรับใช้อ้างอิงถึงคอนเท็กซ์สำหรับการแสดงผลต่าง ๆ) ของออบเจ็กต์ที่เป็นชนิดเดียวกันกับออบเจ็กต์ที่ถูกเลือกเข้าสู่ดีไวซ์คอนเท็กซ์เมื่อก่อนหน้านี้ หากเกิดข้อผิดพลาดจะคืนค่า 0	

2.3 หลักการและทฤษฎีการใช้งานของ Winsocks Control

ประวัติของวินโดวส์ซ็อกเก็ต

ในระบบปฏิบัติการยูนิกซ์ (Unix) มีความสามารถอย่างหนึ่งที่เรียกว่า ซ็อกเก็ต ซึ่งก็คือการที่โปรแกรมต่าง ๆ สามารถสื่อสารข้อมูลระหว่างกันได้โดยไม่จำเป็นว่าจะต้องทำงานอยู่บนเครื่องเดียวกัน ซึ่งได้กลายเป็นมาตรฐานของการสื่อสารข้อมูลทางเครือข่ายคอมพิวเตอร์ไปโดยปริยาย

เมื่อระบบปฏิบัติการ Windows ได้รับความนิยมสูงขึ้นในเวลาต่อมา จึงได้มีการพัฒนาความสามารถในแบบซ็อกเก็ตซึ่งเรียกว่า วินโดวส์ซ็อกเก็ตหรือวินซ็อก โดยการพัฒนาวินซ็อกเกิดจากความร่วมมือระหว่างหลายบริษัทด้วยกันและวินซ็อกได้ถูกทำให้เป็นมาตรฐานเปิด (OpenStandard) ซึ่งทางทีมงานได้ปล่อยวินซ็อก 1.0 ออกมาในเดือนมิถุนายน ค.ศ. 1992 และอีก 6 เดือนถัดมาคือมกราคม ค.ศ. 1993 ก็ได้ปล่อยวินซ็อก 1.1 ออกมาและได้กลายเป็นมาตรฐานที่ใช้ในวินโดวส์ 95

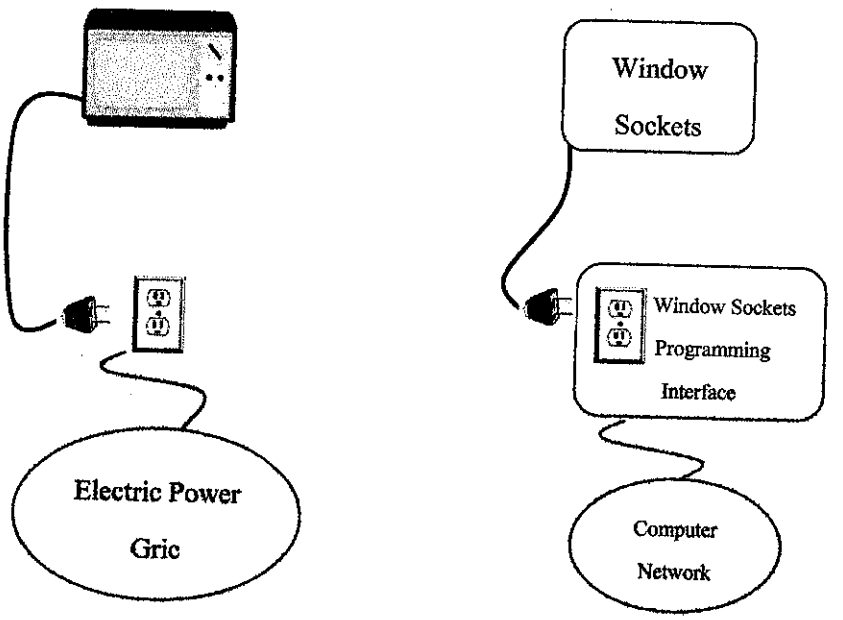
ในมุมมองของนักพัฒนาโปรแกรมบนระบบเครือข่าย วินซ็อก คือ ชุดคำสั่งมาตรฐานให้เรียกใช้ซึ่งเรียกชุดคำสั่งมาตรฐานนี้ว่า เอพีไอ (API : Application Program Interface) ดังนั้นวินซ็อกจึงมีอีกชื่อหนึ่งเรียกว่า วินซ็อกเอพีไอ (Winsock API)

ในระบบปฏิบัติการวินโดวส์ 95/98 จะมีไลบรารีไฟล์สำหรับเรียกใช้วินซ็อกเอพีไออยู่แล้ว โดยไฟล์นี้มีชื่อว่า wsock32.dll ซึ่งจะถูกเก็บอยู่ในโฟลเดอร์ \windows\system

วินซ็อกคืออะไร

วินซ็อก คือ ระบบเปิดที่เชื่อมต่อการเขียนโปรแกรมระบบเครือข่ายภายใต้ระบบปฏิบัติการวินโดวส์ คำว่าระบบเปิดหมายถึง เป็นระบบที่ผู้พัฒนาโปรแกรมสามารถจะนำวินซ็อกมาใช้และดัดแปลงโดยที่ไม่ต้องจ่ายเงินค่าลิขสิทธิ์

วินซ็อก ได้รวบรวมฟังก์ชั่นและโครงสร้างข้อมูลที่จำเป็นต่อการพัฒนาโปรแกรมในระบบเครือข่ายให้ผู้พัฒนาได้นำไปใช้ภายใต้มาตรฐานเดียวกัน ถ้าจะให้เปรียบเทียบแล้ว วินซ็อกเปรียบเสมือนปลั๊กเสียบสำหรับการเขียนโปรแกรม(Programming Plug) ที่จะเชื่อมต่อระหว่างโปรแกรมกับระบบเครือข่ายเข้าด้วยกัน เช่นเดียวกันกับ ปลั๊กไฟที่เชื่อมไฟฟ้ากับเครื่องใช้ไฟฟ้าเข้าด้วยกัน หรือ เปรียบเหมือนกับแจ๊คโทรศัพท์ที่เชื่อมต่อระหว่างเครื่องโทรศัพท์เข้ากับสายสัญญาณโทรศัพท์



รูปที่ 2.1 หลักการของวินซ็อก

(ที่มา : Windows Socket Network Programming, Bob Quinn and Dave Shute)

การใช้งาน Winsock Control

Visual Basic ได้เตรียม ActiveX Control ที่มีชื่อว่า Winsock Control ไว้ให้เราใช้งาน ซึ่งพร้อมให้เราใช้งานเพื่อสร้างการเชื่อมต่อระหว่างคอมพิวเตอร์สองเครื่องในเครือข่าย ทำให้เราสามารถแลกเปลี่ยนข้อมูลระหว่างกันได้

โหมดการทำงานของ Winsock Control

การทำงานของ Winsock Control กับรูปแบบการสื่อสารข้อมูลของโพรโตคอลทีซีพี/ไอพี (TCP/IP) นี้แบ่งการทำงานออกเป็น 2 รูปแบบ คือ ทีซีพี (Transmission Control Protocol : TCP) และ ยูดีพี (User Datagram Protocol : UDP)

1. ทีซีพี (TCP: Transmission Control Protocol)

ทีซีพี คือการสื่อสารข้อมูลแบบที่ต้องมีการเชื่อมต่อก่อน (Connection Oriented) คือ จะมีการเชื่อมต่อกันในครั้งแรกแล้วระหว่างการรับหรือส่งข้อมูลจะต้องมีการส่งสัญญาณโต้ตอบกันตลอด เช่น ถ้า A ส่งข้อมูลไปหา B ทาง A ก็จะรอจนกว่า B จะตอบกลับมาว่าได้รับข้อมูลอย่างถูกต้อง แล้ว A จึงจะส่งข้อมูลต่อไป และทีซีพีเป็นโพรโตคอลที่มีการรับส่งข้อมูลแบบ stream oriented protocol หมายความว่า การรับส่งข้อมูลจะไม่คำนึงถึงปริมาณข้อมูลที่จะส่งไป แต่จะแบ่งข้อมูลเป็นส่วนย่อย ๆ ก่อน แล้วจึงจะส่งไปยังปลายทางอย่างต่อเนื่องเป็นลำดับข้อมูล ในกรณีที่ข้อมูลส่วนใดส่วนหนึ่งสูญหายไป ก็จะส่งข้อมูลส่วนนั้นใหม่อีกครั้ง สำหรับปลายทางก็จะทำหน้าที่จัดเรียงส่วนของข้อมูลดาต้าแกรม (datagram) ใหม่ให้ต่อเนื่องกัน และประกอบกลับเป็นข้อมูลทั้งหมดได้ ซึ่งจะแยกข้อมูลที่ไม่ถูกต้องออก ดังนั้นโปรแกรมประยุกต์หรือโปรเซสใดที่อาศัยการส่งผ่านข้อมูลด้วยโปรโตคอลทีซีพี จะต้องใช้หน่วยความจำหรือขนาดของช่องสัญญาณ (bandwidth) มากกว่ายูดีพี

ในระหว่างการรับส่งข้อมูลนี้ โพรโตคอลทีซีพี จะเพิ่มขบวนการสอบทานข้อมูลเพื่อให้ข้อมูลมีความถูกต้องไม่ผิดผลาดไปจากเดิม โดยการส่งสัญญาณสอบทานข้อมูล และส่งข้อมูลให้ใหม่อีกครั้งถ้าปลายทางไม่ได้รับข้อมูลหรือเกิดความผิดผลาดของข้อมูลขึ้น

ความน่าเชื่อถือของการส่งผ่านข้อมูลโดยโพรโตคอลทีซีพีจะมีมากกว่า แต่ก็ต้องอาศัยทรัพยากรของระบบมากเช่นกัน

2. ยูดีพี (UDP: User Datagram Protocol)

โพรโตคอลยูดีพี เป็นการทำงานในลักษณะตรงกันข้ามกับโพรโตคอลทีซีพี นั่นคือ ในการรับส่งข้อมูลผ่านโพรโตคอลยูดีพี จะเป็นแบบที่ทั้งสองด้านไม่จำเป็นต้องอาศัยการสร้างช่องทางเชื่อมต่อกัน (connectionless) ระหว่างเครื่องเซิร์ฟเวอร์ให้บริการกับเครื่องที่ขอใช้บริการ โดยไม่ต้องแจ้งให้ฝ่ายรับข้อมูลเตรียมรับข้อมูลเหมือนโพรโตคอลทีซีพี และไม่มีการตรวจสอบความถูกต้องของข้อมูลในการรับส่งข้อมูลนั้น ๆ ด้วย เนื่องจากโพรโตคอลยูดีพีนี้ไม่มีสัญญาณสอบทานข้อมูล (acknowledgement) ในการส่งข้อมูลแต่ละครั้ง และไม่มีการส่งข้อมูลใหม่อีกในกรณีที่เกิดความ

ผิดผลาดของข้อมูลในการส่งข้อมูล เมื่อเป็นเช่นนี้โปรแกรมประยุกต์หรือโปรเซสใดที่ต้องอาศัยโพรโตคอลยูดีพี ในการส่งผ่านข้อมูลก็อาจจะต้องสร้างขบวนการตรวจสอบข้อมูลขึ้นมาเอง

ดังนั้นในการรับส่งข้อมูลนั้น แบบทีซีพีจะมีความถูกต้องของข้อมูลมากกว่าแบบยูดีพี แต่จะช้าและยุ่งยากกว่า

พรอพเพอร์ตี้สำคัญของ Winsock Control

Protocol	เป็นการเลือกโพรโตคอลสำหรับการทำงาน
LocalPort	เป็นการกำหนดหมายเลขพอร์ตของคอมพิวเตอร์ที่จะใช้งานกับ Winsock
RemoteHost	เป็นการกำหนดชื่อของคอมพิวเตอร์ที่เราจะติดต่อด้วย อาจจะเป็นหมายเลขIP Address หรือ เป็นชื่อคอมพิวเตอร์ที่เป็นชื่อที่ง่ายต่อการจดจำ
RemotePort	เป็นการกำหนดหมายเลขพอร์ตของคอมพิวเตอร์ที่เราจะติดต่อด้วย
ByteReceive	เป็นจำนวนข้อมูลที่รับเข้ามาเก็บไว้ในบัฟเฟอร์ (จากวิธีการ GetData) มีหน่วยเป็นไบต์

วิธีการสำคัญของ Winsock Control

Listen	เป็นวิธีการที่ใช้สร้าง Socket ทำให้คอมพิวเตอร์เครื่องอื่น ๆ สามารถติดต่อเข้ามาได้
Connect	เป็นวิธีการที่ใช้สร้างการติดต่อแบบ Socket ไปยังคอมพิวเตอร์เครื่องอื่น โดยจะต้องระบุ Socket Address (IP Address กับหมายเลขพอร์ตที่กำหนดให้ใช้กับ Socket)
Accept	เป็นวิธีการที่ใช้รับ Request จากคอมพิวเตอร์ที่ติดต่อเข้ามา
SendData	เป็นวิธีการที่ใช้ส่งข้อมูลไปยังคอมพิวเตอร์เครื่องอื่น ๆ ที่เราติดต่อแบบ Socket
GetData	เป็นวิธีการที่ใช้รับข้อมูลจากบัฟเฟอร์เข้ามาเก็บในตัวแปรที่เรากำหนดให้ โดยสามารถกำหนดชนิดตัวแปร และความยาวของข้อมูลที่จะนำมาเก็บได้
Close	เป็นวิธีการที่ใช้ยกเลิกการติดต่อแบบ Socket

เหตุการณ์สำคัญของ Winsock Control

ConnectionRequest	เป็นเหตุการณ์ที่เกิดเมื่อ คอมพิวเตอร์เครื่องอื่นมีการ Request เข้ามา ซึ่งจะมีการกำหนด ID ให้กับแต่ละ Request ที่เข้ามา
DataArrival	เป็นเหตุการณ์ที่เกิดเมื่อ มีข้อมูลชุดใหม่เข้ามาเก็บในบัฟเฟอร์ ซึ่งเราสามารถตรวจสอบขนาดของข้อมูลนั้นได้ จากพรอพเพอร์ตี้ ByteReceive
SendProgress	เป็นเหตุการณ์ที่เกิดขณะที่กำลังมีการส่งข้อมูลระหว่างกัน ซึ่งจะมีพารามิเตอร์แสดงจำนวนข้อมูลที่ส่งมาแล้ว และข้อมูลที่ยังคงเหลือ
SendComplete	เป็นเหตุการณ์ที่เกิดเมื่อการส่งข้อมูลเสร็จสิ้นสมบูรณ์
Error	เป็นเหตุการณ์ที่เกิดเมื่อมีความผิดพลาดเกิดขึ้น ซึ่งจะแสดงหมายเลขของความผิดพลาด, คำอธิบาย และรายละเอียดอื่น ๆ สำหรับการจัดการกับข้อผิดพลาดที่เกิดขึ้น

กลไกการทำงานของโปรแกรมในระบบเครือข่าย

- **แบบจำลองไคลเอนท์-เซิร์ฟเวอร์ (Client – Server Model)**

ทุก ๆ โปรแกรมในระบบเครือข่ายนั้น จะต้องมีปลายทางของการสื่อสาร ซึ่งก็คือไคลเอนท์ และ เซิร์ฟเวอร์ โดยในการสื่อสารไคลเอนท์จะส่งแพ็กเก็ตแรก และ เซิร์ฟเวอร์จะรับไปเพื่อสร้างการเชื่อมต่อ

ในการเชื่อมต่อระหว่างไคลเอนท์กับเซิร์ฟเวอร์นั้น ไคลเอนท์จะต้องรู้ตำแหน่งและรู้จักซ็อกเก็ตของเซิร์ฟเวอร์ และเซิร์ฟเวอร์จะต้องตั้งชื่อซ็อกเก็ตของตัวเองเพื่อให้ไคลเอนท์สามารถใช้อ้างอิงได้ ชื่อของซ็อกเก็ตจะสอดคล้องกับไอพีแอดเดรส (IP Address) และหมายเลขพอร์ต (Port - Number)

เมื่อไคลเอนท์ทำการเชื่อมต่อกับเซิร์ฟเวอร์ได้สำเร็จ ซ็อกเก็ตของทั้งสองจะถูกรวมกันอยู่ในรูปแบบของการเชื่อมต่อ ซึ่งประกอบด้วยองค์ประกอบ 4 สิ่งด้วยกันคือ

1. โพรโตคอล (ต้องเป็นโพรโตคอลเดียวกันทั้งไคลเอนท์และเซิร์ฟเวอร์)
2. ไอพีแอดเดรสของไคลเอนท์
3. หมายเลขพอร์ตของไคลเอนท์
4. ไอพีแอดเดรสของเซิร์ฟเวอร์
5. หมายเลขพอร์ตของเซิร์ฟเวอร์

- **ขั้นตอนการทำงานของโปรแกรมในระบบเครือข่าย**

โปรแกรมในระบบเครือข่ายทั้งหมดไม่ว่าจะเป็นไคลเอนท์หรือเซิร์ฟเวอร์ จะมีขั้นตอนการทำงานในการสื่อสารข้อมูล 5 ขั้นตอนดังนี้

1. เปิดซ็อกเก็ต (Open a socket)

ทั้งไคลเอนท์และเซิร์ฟเวอร์ต้องการซ็อกเก็ตในการสื่อสารข้อมูลในระบบเครือข่ายการเปิดซ็อกเก็ตทำได้โดยใช้ฟังก์ชัน socket()

2. ให้ชื่อซ็อกเก็ต (Name the socket)

ไคลเอนท์ต้องสามารถรู้ตำแหน่งและรู้จักซ็อกเก็ตของเซิร์ฟเวอร์และเซิร์ฟเวอร์ต้องให้ชื่อกับซ็อกเก็ตเพื่อให้ไคลเอนท์สามารถใช้อ้างอิงได้ ซึ่งชื่อของซ็อกเก็ตจะประกอบด้วย 3 สิ่งคือ โพรโตคอล หมายเลขพอร์ต และแอดเดรส

3. สื่อสารกับซ็อกเก็ตอื่น (Associate with another socket)

การสื่อสารระหว่าง 2 ซ็อกเก็ต มีขั้นตอนดังนี้คือ

- เซิร์ฟเวอร์เตรียมการสำหรับการสื่อสาร
- ไคลเอนท์เริ่มการสื่อสาร
- เซิร์ฟเวอร์ตอบสนองการสื่อสาร

หลังจากการสื่อสารสำเร็จ ทั้งไคลเอนท์และเซิร์ฟเวอร์จะรู้จักซ็อกเก็ตของอีกฝ่าย

<u>การเตรียมการสื่อสารของเซิร์ฟเวอร์</u>

สำหรับสตรีมเซิร์ฟเวอร์ (Stream Server) ซึ่งทำงานแบบ ทีซีพี คือ ต้องมีการเชื่อมต่อก่อนจึงจะส่งข้อมูลได้ ทางฝั่งเซิร์ฟเวอร์จะรอรับการเชื่อมต่อด้วยฟังก์ชัน listen()

<u>การเริ่มการสื่อสารของไคลเอนท์</u>

สำหรับการเชื่อมต่อแบบทีซีพี ทางฝั่งไคลเอนท์จะต้องเรียกฟังก์ชัน connect() เพื่อที่จะเริ่มการเชื่อมต่อกับเซิร์ฟเวอร์

<u>เซิร์ฟเวอร์ตอบรับการเชื่อมต่อ</u>

สำหรับการเชื่อมต่อแบบทีซีพีทางฝั่งเซิร์ฟเวอร์นั้นจะทำการตอบสนองต่อฟังก์ชัน connect ของไคลเอนท์ด้วยฟังก์ชัน accept() ซึ่งจะคอยตรวจสอบการ connect() ของไคลเอนท์

4. ส่งและรับข้อมูลระหว่างซ็อกเก็ต (Send and Receive between Socket)

การส่งข้อมูลของซ็อกเก็ตที่ได้ทำการเชื่อมต่อเรียบร้อยแล้ว เราจะใช้ฟังก์ชัน send() นั่นหมายความว่าก่อนที่จะใช้ฟังก์ชัน send() ได้ จะต้องมีการเรียกฟังก์ชัน connect() สำเร็จก่อน

5. ปิดซ็อกเก็ต (Close Socket)

ทั้งไคลเอนท์และเซิร์ฟเวอร์ เมื่อต้องการยกเลิกการสื่อสารข้อมูลในระบบเครือข่าย จะต้องทำการปิดซ็อกเก็ต โดยใช้ฟังก์ชัน close()

กลไกการทำงานของไคลเอนท์-เซิร์ฟเวอร์

ในการพัฒนาโปรแกรมที่ทำงานในระบบเครือข่ายโดยการใช้วินซ็อก เราจะสามารถสร้างโปรแกรมที่เป็นไคลเอนท์ และ โปรแกรมที่เป็นเซิร์ฟเวอร์ ขึ้นมาคู่กัน โดยที่เราเพียงปรับปรุงโปรแกรมเล็กน้อย เพราะขั้นตอนการทำงานของไคลเอนท์และเซิร์ฟเวอร์มีลักษณะการเรียกใช้งานฟังก์ชันของวินซ็อกเอพีไอที่คล้าย ๆ กัน จะต่างกันเพียงบางขั้นตอนเท่านั้น

นอกจากฟังก์ชันในกลุ่มที่ใช้ในการเชื่อมต่อกันระหว่างเซิร์ฟเวอร์แล้ว วินซ็อกยังมีฟังก์ชันอื่น ๆ ที่ใช้เพื่อเตรียมการเชื่อมต่อ หรือประโยชน์อื่น ๆ อีก เช่น ฟังก์ชันที่ใช้ในการรับค่าแอดเดรสของเครื่อง ฟังก์ชันที่ใช้ในการกำหนดรูปแบบการทำงานของซ็อกเก็ตและอื่น ๆ อีกมาก

2.4 ทฤษฎีและหลักการที่นำมาใช้ในการจัดการฐานข้อมูล

ในการพัฒนาโปรแกรมนี้ จะมีการเข้าถึงฐานข้อมูลอยู่ตลอดเวลา ซึ่งผู้พัฒนาจะกำหนดประเภทของ Recordset อยู่ 2 ลักษณะ คือ

- Table เป็นชนิดของ Recordset ที่กระทำกับข้อมูลใน Table โดยตรงซึ่ง Recordset ชนิดนี้จะมีความเร็วในการค้นหาข้อมูลดีที่สุด เนื่องจากใช้ Index เป็นฟิลด์พิเศษช่วยในการค้นหาข้อมูล ซึ่งผู้พัฒนาจะเปิดฐานข้อมูลในลักษณะนี้ในเรื่องของ Dictionary of CrossWordd

- Dynaset เป็นชนิดของ Recordset ที่สามารถสร้างจากตารางเดียวหรือหลายตารางก็ได้และเราก็สามารถแก้ไขค่าของข้อมูลในตารางได้ซึ่งผู้พัฒนาจะเปิดฐานข้อมูลในลักษณะนี้ในขณะที่มีการเล่นเกมตัวต่อประลองปัญญา

<u>วิธีการ (Method) ที่ใช้จัดการกับข้อมูลในฐานข้อมูล</u> มีดังนี้

- AddNew เพิ่มข้อมูลเข้าไปใน Recordset
- Delete ลบข้อมูลใน Recordset

- Edit ใช้แก้ไขข้อมูลใน Recordset
- Update ใช้บันทึกข้อมูลของ Recordset

วิธีการ (Method) ที่ใช้ในการเลื่อน Pointer

- MoveFirst สั่งให้ Pointer เลื่อนไปยัง Record แรกใน Recordset
- MoveLast สั่งให้ Pointer เลื่อนไปยัง Record สุดท้ายใน Recordset
- MoveNext สั่งให้ Pointer เลื่อนไปยัง Record ถัดไปใน Recordset
- MovePrevious สั่งให้ Pointer เลื่อนไปยัง Record ก่อนหน้าใน Recordset

สำหรับการค้นหาข้อมูลในฐานข้อมูลนั้น ผู้พัฒนาจะใช้ Method ดังต่อไปนี้

- Seek ใช้ค้นหาข้อมูลใน Recordset ซึ่งจะต้องมีการกำหนด Index ในฐานข้อมูล และ Seek นี้ก็เหมาะสำหรับการค้นหาข้อมูลในฐานข้อมูลที่มีขนาดใหญ่
- FindFirst ใช้สำหรับค้นหา Recordset โดยทิศทางในการหาจะเริ่มจาก Record แรก ไปยัง Record สุดท้ายใน Recordset จนกระทั่งพบ Record ซึ่งตรงกับเงื่อนไขในการค้นหา

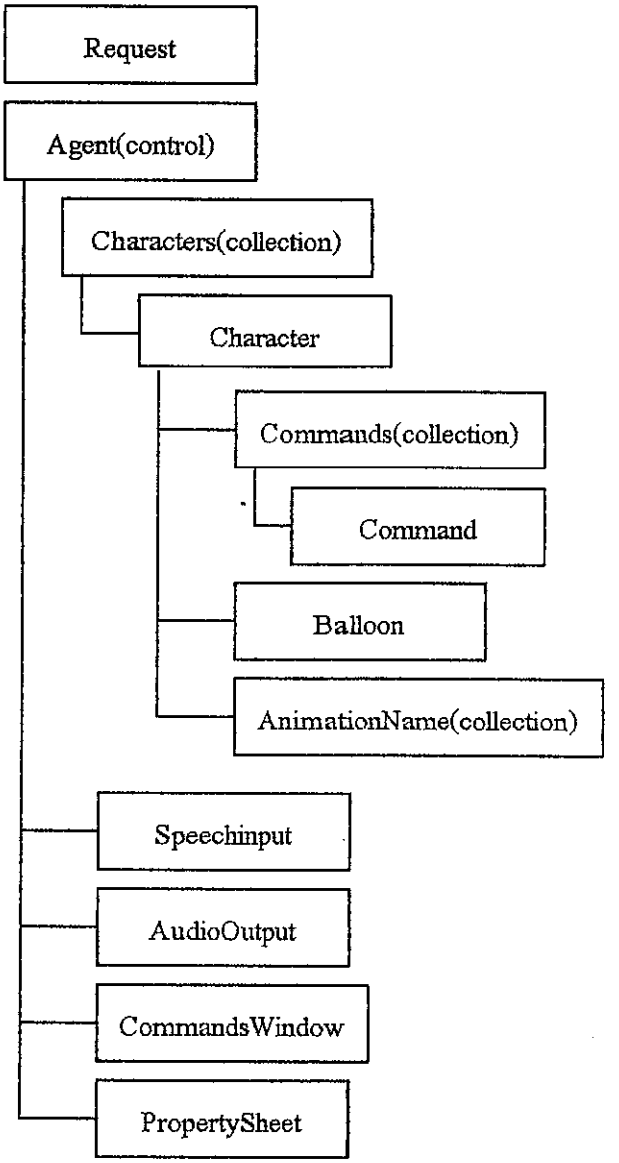
2.5 การเขียนโปรแกรมเกี่ยวกับ Microsoft Agent Control

Microsoft Agent เป็น component หนึ่ง ที่ทางไมโครซอฟต์ได้พัฒนาและปรับปรุงขึ้นมา เพื่อให้นักพัฒนาโปรแกรมทั้งหลายสามารถนำไปใช้ในการพัฒนาโปรแกรมเพื่ออำนวยประโยชน์ให้แก่แอพลิเคชันของพวกเขาเหล่านั้น ขณะนี้ Microsoft Agent ก็ได้รับการพัฒนามาจนถึง version 2.0 แล้ว

เราสามารถใช้คอนโทรล (control) ของ Microsoft Agent ได้จากแอพลิเคชัน Microsoft Visual Basic หรือภาษาโปรแกรมอื่น ๆ อีก เช่น C, C++ หรือ Java แต่เราต้องแน่ใจก่อนว่าภาษานั้นสนับสนุนและอินเตอร์เฟสกับ ActiveX Control นั้นจริง ๆ การที่จะเข้าถึงหน่วยควบคุมของ Agent ได้นั้น Agent จะต้องติดตั้งอยู่บน system เรียบร้อยเสียก่อน ถ้าเราต้องการที่จะติดตั้ง Microsoft Agent เราจะต้องได้รับการอนุญาตจากทางไมโครซอฟต์เสียก่อน

เราสามารถที่จะได้รับการอนุญาตนั้น ถ้าเราเข้าไปทำความตกลงการได้รับอนุญาตกับเว็บไซต์ (http://www.microsoft.com/sitebuilder/workshop/imedia/agent/licensing.asp) และที่นี่เราสามารถทำการดาวน์โหลดไฟล์ที่จะใช้ทำการติดตั้งได้ สำหรับระบบปฏิบัติการ windows 2000 และ windows NT 5.0 นั้น ไม่ต้องทำการติดตั้ง Microsoft Agent เพราะว่าทางไมโครซอฟต์ได้ทำการติดตั้งมาให้เรียบร้อยแล้ว

โครงสร้างของออบเจ็กต์ Microsoft Agent ประกอบด้วยออบเจ็กต์ดังต่อไปนี้



รูปที่ 2.2 โครงสร้างของออบเจ็กต์ Microsoft Agent

(ที่มา : Windows Socket Network Programming, Bob Quinn and Dave Shute)

ในแต่ละออบเจ็กต์ของโครงสร้าง Microsoft Agent นั้นก็จะมี Method, Event ต่าง ๆ อยู่มากมายแต่ในการพัฒนาโครงงานนี้ไม่ได้นำ Method และ Event เหล่านั้นมาใช้ทั้งหมด ดังนั้นผู้พัฒนาจึงได้อธิบายเฉพาะออบเจ็กต์ที่นำมาใช้งานเท่านั้น

The Request Object

เราสามารถใช้ออบเจ็กต์ Request นี้ในการตรวจสอบสถานภาพของวิธีการต่าง ๆ ได้เช่น Load, Play, Speak, Show และ Hide โดยการกำหนดตัวแปรให้แก่วิธีการนั้น

ตัวอย่าง

```
Dim MyRequest As Object                                    ' เป็นการประกาศตัวแปร
Set MyRequest = Agent1.Characters.Load ("Genie")           ' เป็นการเซตค่าให้กับตัวแปร
และ "http://agent.microsoft.com/characters/v2/genie/genie.acf")  ยังเป็นการโหลด Character
                                                           ให้กับตัวแปรด้วย

If (MyRequest.Status = 2 ) then
   'do something
Else If (MyRequest.Status = 0 ) then
   'do something right away
End If
```

จากตัวอย่าง Status เป็น property ที่สามารถบอกสถานภาพปัจจุบันของการ Request ว่าเป็นอย่างไร ค่าคืนกลับของ Status จะเป็นค่า long integer

Status	คำอธิบาย
0	การ Request สำเร็จ
1	การ Request ล้มเหลว
2	การ Request อยู่ในระหว่างการรอคอย
3	การ Request ถูกขัดจังหวะ
4	การ Request กำลังดำเนินการอยู่

The Agent Control

การอ้างอิงถึง Agent control จะเป็นการเตรียมเพื่อที่จะเข้าถึง events และ properties ต่าง ๆ ดังนี้

#Show Event

คำอธิบาย

เกิดขึ้นเมื่อ Character ถูกแสดงขึ้นมา

รูปแบบไวยากรณ์

Character.**Show**

The Character Object

ในการเข้าถึงออบเจ็กต์ Character เราจะต้องทำการโหลดข้อมูลของ Character เข้ามาใน Character Collection ให้เรียบร้อยเสียก่อนซึ่งในคอลเลกชั่นของออบเจ็กต์ Character นี้จะประกอบด้วยหลายวิธีการ ดังนี้

#Load Method

คำอธิบาย

เป็นการโหลด Character เข้ามาไว้ในคอลเลกชั่น Characters

รูปแบบไวยากรณ์

agent.**Characters.Load** "*CharacterID*",*Provider*

Part	คำอธิบาย
CharacterID	จำเป็นจะต้องมีเนื่องจากจะต้องใช้ในการอ้างอิงถึงข้อมูลของ Character ที่จะต้องถูกโหลดเข้ามา
Provider	จำเป็นจะต้องมี เป็นข้อมูลชนิด Variant ซึ่งอาจจะเป็นได้ดังนี้
	Fileopen ตำแหน่งของ local file ที่ใช้ระบุถึง definition file ของ Character
	URL ที่อยู่แบบ HTTP ของ definition file ของ Character

ตัวอย่าง

```
Agent.Character.Load"genie","MyCharacters\genie.acs"
'เราสามารถระบุถึงไดเรกทอรีของ Character ได้
```

#Unload Method

คำอธิบาย

เป็นการ unload ข้อมูลของ Character ที่เรานำมาเก็บไว้ที่ Character collection

รูปแบบไวยากรณ์

*agent.***Character.Unload** *"CharacterID"*

#Speak Method

คำอธิบาย

เป็นการแสดงการพูดหรือการอ่านออกเสียง

รูปแบบไวยากรณ์

*Character.***Speak** *"Word of Speech"*

#Play Method

คำอธิบาย

เป็นการแสดงภาพเคลื่อนไหวต่าง ๆ

รูปแบบไวยากรณ์

*Character.***Play** *" Name of the Animation"*

ซึ่ง Name of the Animation หรือชื่อของอากัปกิริยาที่ต้องการให้ Genie แสดงออกมามีดังนี้

ตารางที่ 2.1 รายชื่อของอากัปกิริยาที่ต้องการให้ Ginie แสดงออกมา

Name of the Animation	Return Animation	Supports Speaking	Sound Effects	Assigned to State	Description
Acknowledge	None	No	No	None	Nods head
Alert	Yes, using Exit branches	Yes	No	**Listening**	Straightens and raises eyebrows
Announce	Yes, using Exit	Yes	No	None	Raises hand

Announce	Yes, using Exit branches	Yes	No	None	Raises hand
Blink	None	No	No	**IdlingLevel1** **IdlingLevel2**	Blinks eyes
Confused	Yes, using Exit branches	Yes	No	None	Scratches head
Congratulate	Yes, using Exit branches	Yes	Yes	None	Applauds
Congratulate_2	Yes, using Exit branches	Yes	No	None	Gives thumbs-up gesture
Decline	Yes, using Exit branches	Yes	No	None	Raises hands and shakes head
DoMagic1	None	Yes	No	None	Turns to side and raises hands
DoMagic2	Yes, using Exit branches	No	Yes	None	Lowers hands, clouds appear
DontRecognize	Yes, using Exit branches	Yes	No	None	Holds hand to ear
Explain	Yes, using Exit	Yes	No	None	Extends

	branches				hand to ear
Explain	Yes, using Exit branches	Yes	No	None	Extends arms to side
GestureDown	Yes, using Exit branches	Yes	No	GesturingDown	Gestures down
GestureLeft	Yes, using Exit branches	Yes	No	GesturingLeft	Gestures left
GestureRight	Yes, using Exit branches	Yes	No	GesturingRight	Gestures right
GestureUp	Yes, using Exit branches	Yes	No	GesturingUp	Gestures up
GetAttention	GetAttentionReturn	Yes	No	None	Waves arms
GetAttentionContinued	GetAttentionReturn	Yes	No	None	Waves arms again
GetAttentionReturn	None	No	No	None	Returns to neutral position
Greet	Yes, using Exit branches	Yes	No	None	Bows
Hearing_1	None	No	No	**Hearing**	Ears extend (*looping animatio

Hearing_3	None	No	No	**Hearing**	Turns head left (*looping animation)
Hearing_4	None	No	No	**Hearing**	Turns head right (*looping animation)
Hide	None	No	Yes	**Hiding**	Disappears into smoke
Idle1_1	None	No	No	**IdlingLevel1** **IdlingLevel2**	Takes breath
Idle1_2	None	No	No	**IdlingLevel1** **IdlingLevel2**	Glances right and blinks
Idle1_3	Yes, using Exit branches	No	No	**IdlingLevel1** **IdlingLevel2**	Glances left and blinks
Idle1_4	None	No	No	**IdlingLevel1** **IdlingLevel2**	Glances up to the right and blinks
Idle1_5	Yes, using Exit branches	No	No	**IdlingLevel1** **IdlingLevel2**	Glances down and blinks
Idle1_6	None	No	No	**IdlingLevel1**	Glances

				IdlingLevel2	up and blinks
Idle2_1	None	No	No	**IdlingLevel2**	Wisp snakes
Idle2_2	Yes, using Exit branches	No	No	**IdlingLevel2**	Reveals scroll and reads
Idle2_3	Yes, using Exit branches	No	No	**IdlingLevel2**	Reveals scroll and writes
Idle3_1	None	No	Yes	**IdlingLevel3**	Yawns
Idle3_2	Yes, using Exit branches	No	Yes	**IdlingLevel3**	Falls asleep (*looping animation)
LookDown	**LookDownReturn**	No	No	None	Looks down
LookDownBlink	**LookDownReturn**	No	No	None	Blinks looking down
LookDownReturn	None	No	No	None	Returns to neutral position
LookLeft	**LookLeftReturn**	No	No	None	Looks left
LookLeftBlink	**LookLeftReturn**	No	No	None	Blinks looking left

LookLeftReturn	None	No	No	None	Returns to neutral position
LookRight	**LookRightReturn**	No	No	None	Looks right
LookRightBlink	**LookRightReturn**	No	No	None	Blinks looking right
LookRightReturn	None	No	No	None	Returns to neutral position
LookUp	**LookUpReturn**	No	No	None	Looks up
LookUpBlink	**LookUpReturn**	No	No	None	Blinks looking up
LookUpReturn	None	No	No	None	Returns to neutral position
MoveDown	Yes, using Exit branches	No	Yes	**MovingDown**	Flies down
MoveLeft	Yes, using Exit branches	No	Yes	**MovingLeft**	Flies left
MoveRight	Yes, using Exit branches	No	Yes	**MovingRight**	Flies right
MoveUp	Yes, using Exit branches	No	Yes	**MovingUp**	Flies up
Pleased	Yes, using Exit	Yes	No	None	Smiles

					neutral position
Reading	Yes, using Exit branches	No	Yes	None	Reveal scroll and reads (*looping animation)
RestPose	None	Yes	No	**Speaking**	Neutral position
Sad	Yes, using Exit branches	Yes	No	None	Sad expression
Search	No	No	No	None	Reveals binoculars and turns
Searching	Yes, using Exit branches	No	No	None	Reveals binoculars and turns (*looping animation)
Show	None	No	Yes	**Showing**	Appears out of smoke
StartListening	Yes, using Exit branches	Yes	No	None	Puts hand to ear
StopListening	Yes, using Exit branches	Yes	No	None	Puts hands over ears
Suggest	Yes, using Exit branches	Yes	No	None	Displays lightbulb
Surprised	Yes, using Exit branches	Yes	No	None	Looks surprised
Think	Yes, using Exit branches	Yes	No	None	Looks up with hand on chin

Thinking	No	No	No	None	Looks up with hand on chin (*looping animation)
Uncertain	Yes, using Exit branches	Yes	No	None	Moves one hand to chin, other to hip, and raises right eyebrow
Wave	Yes, using Exit branches	Yes	No	None	Waves
Write	**WriteReturn**	Yes	Yes	None	Reveals scroll, writes and looks up
WriteContinued	**WriteReturn**	Yes	Yes	None	Writes and looks up
WriteReturn	None	No	No	None	Returns to neutral position
Writing	Yes, using Exit branches	No	Yes	None	Reveals scroll, writes (*looping animation)