



การเรียนรู้และพัฒนาเกม 2 มิติ ด้วย Adobe Flash และ Actionscript 2.0

Game 2D Learning and Development with Adobe Flash and Actionscript 2.0

นางสาวณภัทร แก้วพิจิตร

โครงการสหกิจศึกษานี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร

ปริญญาวิทยาศาสตรบัณฑิต สาขาเทคโนโลยีสารสนเทศ

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีไทย – ญี่ปุ่น

พ.ศ.2554

การเรียนรู้และพัฒนาเกม 2 มิติ ด้วย Adobe Flash และ Actionscript 2.0

Game 2D Learning and Development with Adobe Flash and Actionscript 2.0

นางสาวณภัทร แก้วพิจิตร

โครงการสหกิจศึกษานี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร

ปริญญาวิทยาศาสตรบัณฑิต สาขาเทคโนโลยีสารสนเทศ

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีไทย – ญี่ปุ่น

พ.ศ.2554

คณะกรรมการตรวจสอบ

.....ประธานกรรมการสอบ

(อาจารย์ ดร.สะพรั่งสิทธิ์ มฤตุสาธ)

.....กรรมการสอบและอาจารย์ที่ปรึกษา

(อาจารย์ กานต์ ศักดิ์คำดวง)

..... กรรมการสอบ

(อาจารย์ อมรพันธ์ ชมกลิ่น)



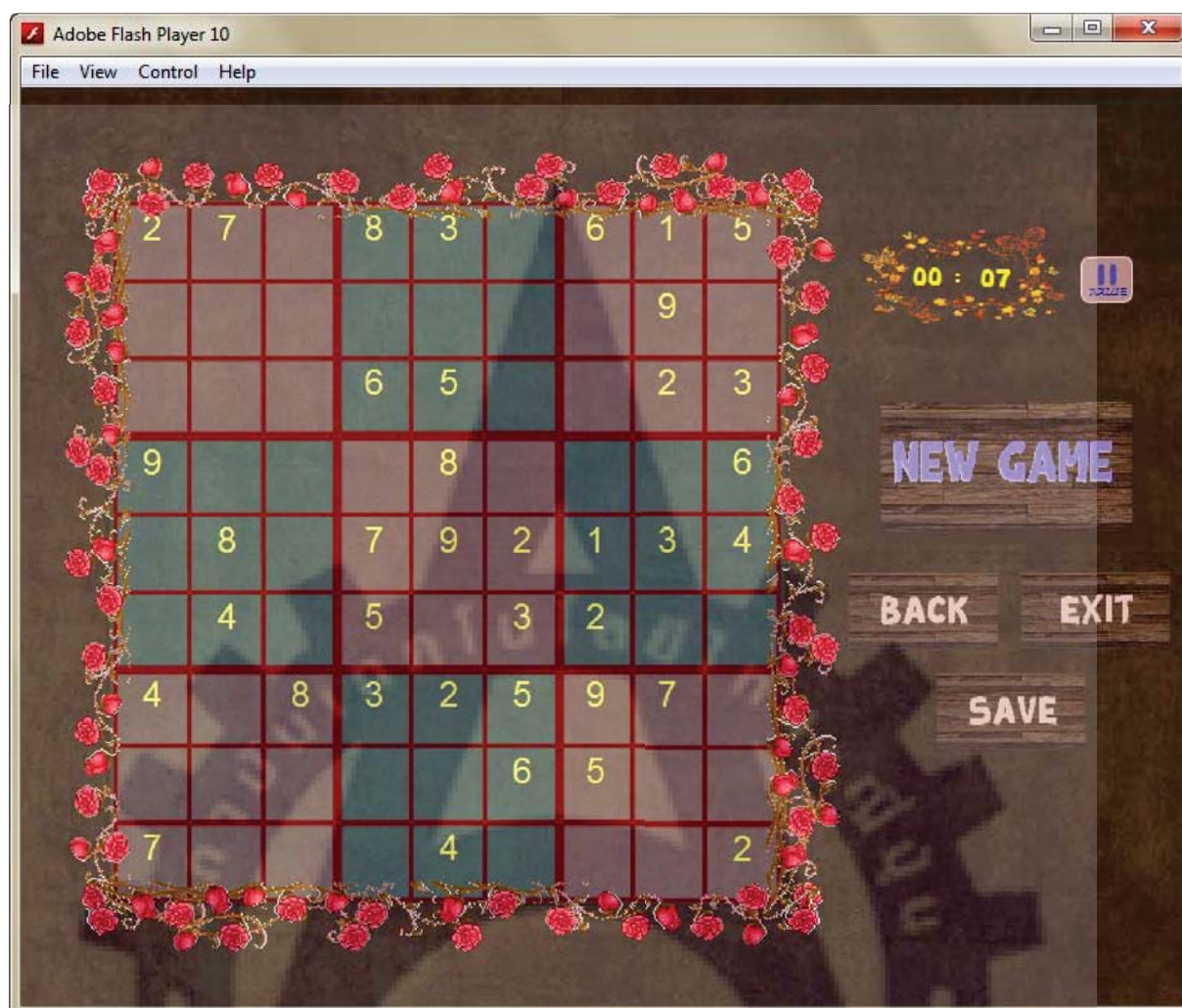
บทสรุป

ชื่อโครงการ	การเรียนรู้และพัฒนาเกม 2 มิติ ด้วย Adobe Flash และ Actionscript 2.0		
	Game 2D Learning and Development with Adobe Flash and Actionscript 2.0		
ผู้เขียน	นางสาว ฌภัทร แก้วพิจิตร		
คณะวิชา	เทคโนโลยีสารสนเทศ	สาขาวิชา	เทคโนโลยีสารสนเทศ
อาจารย์ที่ปรึกษา	อาจารย์กานต์ ศักดิ์คำดวง		
พนักงานที่ปรึกษา	นายศิริพงษ์ ยกย่อ่ง		
ชื่อบริษัท	Digicraft Co.,Ltd		
ประเภทธุรกิจ	Digital Content		
งานที่ปฏิบัติ			

เนื่องจากบริษัทได้ทำงานเกี่ยวกับการออกแบบ พัฒนาสื่อออนไลน์ต่างๆ และเกม ดังนั้นงานที่ได้รับมอบหมายในการฝึกสหกิจครั้งนี้คือการ ออกแบบและพัฒนาเกม 2 มิติ ด้วยโปรแกรม Adobe Flash และ Action script 2.0 โดยการดำเนินงานจะเริ่มตั้งแต่การเรียนรู้ ทั้งตัวโปรแกรมและภาษาที่ใช้ในการทำงาน การออกแบบพัฒนา จัดทำเกมขึ้นมาจริงๆ อีกทั้งยังมีการเรียนรู้เรื่องเกมขั้นต้นต่างๆซึ่งเป็นพื้นฐานความรู้ที่จะนำมาออกแบบและพัฒนาเกมนั่นเอง

ผลที่ได้รับจากการดำเนินงานและประโยชน์ที่ได้รับ

จากการปฏิบัติงานสหกิจในครั้งนี้ ผลที่ได้รับคือความรู้ ความเข้าใจ ตลอดจนการสามารถนำความรู้เกี่ยวกับเรื่องเกม Adobe Flash และ Action script 2.0 ไปใช้ในการเรียนและการทำงานจริงๆในอนาคตได้ อีกทั้งยังได้รับความรู้เกี่ยวกับเรื่องการทำงานจริง ความสัมพันธ์ต่างๆในองค์กร และการเรียนรู้เรื่องความรับผิดชอบต่องานต่างๆมากยิ่งขึ้น



ตัวอย่างงานที่ได้มีการจัดทำ

กิตติกรรมประกาศ

การที่ข้าพเจ้าได้มาปฏิบัติงานสหกิจศึกษา ณ บริษัท Digicraft .Co.,Ltd นับตั้งแต่วันที่ 1 มิถุนายน พ.ศ. 2554 ถึงวันที่ 30 กันยายน พ.ศ. 2554 ส่งผลให้ข้าพเจ้าได้รับความรู้และประสบการณ์ต่างๆ ที่มีค่ามากมาย ได้รับความเอาใจใส่ และยังได้รับการอุปการะอย่างดีจากบุคคลหลายๆ ท่าน เรื่อยมา

สำหรับรายงานวิชาสหกิจศึกษานี้ สำเร็จลงได้ด้วยดีและสมบูรณ์จากความร่วมมือและสนับสนุนจากหลายฝ่าย ดังนั้นข้าพเจ้าขอลงนามขอบคุณไว้ ณ โอกาสนี้

1. บิดามารดา บุคคลในครอบครัวและมิตรสหายของข้าพเจ้า
2. อาจารย์ในสาขาวิชาเทคโนโลยีสารสนเทศทุกท่าน
3. นาย เสวิต จิตตุนนท์ Department Manager
4. นาย ศิริพงศ์ ยกย่อง IT Assistant Manager
5. นายธเนศ พรศิริรัตน์ Web Programmer

และบุคคลท่านอื่นๆที่ไม่ได้กล่าวนามทุกท่านที่ได้ให้คำแนะนำช่วยเหลือในการจัดทำรายงาน

ข้าพเจ้าใคร่ขอขอบพระคุณผู้ที่มีส่วนเกี่ยวข้องทุกท่าน ที่มีส่วนร่วมในการให้ข้อมูล เป็นที่ปรึกษาในการทำรายงานฉบับนี้จนเสร็จสมบูรณ์ ตลอดจนให้การดูแลและให้ความเข้าใจเกี่ยวกับชีวิตของการทำงานจริง ข้าพเจ้าขอขอบคุณ ไว้ ณ ที่นี้

นางสาวณภัทร แก้วพิจิตร
ผู้จัดทำรายงาน

สารบัญ

	หน้า
บทสรุป	จ
กิตติกรรมประกาศ	ง
สารบัญ	จ
รายการตาราง	ช
รายการรูปประกอบ	ซ
บทที่	
1. บทนำ	1
1.1 ชื่อและที่ตั้งของสถานประกอบการ	1
1.2 ลักษณะธุรกิจของสถานประกอบการ	1
1.3 รูปแบบการจัดองค์กรและการบริหารงาน	3
1.4 ตำแหน่งและหน้าที่งานที่นักศึกษาได้รับมอบหมาย	4
1.5 พนักงานที่ปรึกษาและตำแหน่งของพนักงานที่ปรึกษา	4
1.6 ระยะเวลาที่ปฏิบัติงาน	4
1.7 วัตถุประสงค์หรือจุดมุ่งหมายของการปฏิบัติงาน	4
1.8 ผลที่คาดว่าจะได้รับการปฏิบัติงาน	4
2. ทฤษฎีและเทคโนโลยีที่ใช้ในการปฏิบัติงาน	5
2.1 ทฤษฎีหลักการและเทคโนโลยีที่ใช้ในการพัฒนาโปรแกรม	5
2.1.1 ทฤษฎีเกี่ยวกับเกมคอมพิวเตอร์	5
2.1.2 ความรู้พื้นฐานการเขียน โปรแกรม	13
2.1.3 Adobe Flash (CS3)	27
2.1.4 Adobe Photoshop (CS3)	28
2.1.5 Microsoft Office	29
2.1.6 Action Script	30

สารบัญ (ต่อ)

3. แผนงานการปฏิบัติงานและขั้นตอนการดำเนินงาน	92
3.1 แผนงานปฏิบัติงาน	92
3.2 รายละเอียดงานที่นักศึกษาปฏิบัติในงานสหกิจศึกษา หรือรายละเอียดโครงการที่	93
3.3 ขั้นตอนการดำเนินงานที่นักศึกษาปฏิบัติงานหรือโครงการ	93
3.3.1 ติดตั้ง Software	93
3.3.2 ศึกษาทฤษฎีขั้นพื้นฐานของการเขียนเกม	94
3.3.3 การออกแบบ พัฒนาเกม 2 มิติ	99
4 . ผลการดำเนินงาน การวิเคราะห์และสรุปผลต่างๆ	108
4.1 ขั้นตอนและผลการดำเนินงาน	108
4.2 ผลการวิเคราะห์ข้อมูล	109
4.3 วิเคราะห์และวิจารณ์ข้อมูลโดยเปรียบเทียบผลที่ได้รับกับวัตถุประสงค์ และจุดมุ่งหมายในการปฏิบัติงานหรือการจัดทำโครงการ	110
5. บทสรุปและข้อเสนอแนะ	111
5.1 สรุปผลการดำเนินงาน	111
5.2 แนวทางการแก้ไขปัญหา	111
5.3 ข้อเสนอแนะจากการดำเนินงาน	112
เอกสารอ้างอิง	113
ภาคผนวก	114
ก. การใช้โปรแกรม Flash	115
ข. วิธีการเล่นเกม และการใช้งาน	167
ประวัติผู้วิจัย	174

รายการตาราง

ตาราง	หน้า
3.1 แผนงานปฏิบัติงาน	92
4.1 เปรียบเทียบผลที่ได้รับกับวัตถุประสงค์และจุดมุ่งหมายในการปฏิบัติงาน หรือการผล จัดทำโครงการ	110
ก.1 นามสกุลของไฟล์ภาพนิ่ง	122
ก.2 นามสกุลของไฟล์ภาพเคลื่อนไหว	122



รายการรูปประกอบ

รูป	หน้า
1.1 ภาพแสดงผังการบริหาร	3
2.1 รูปแบบแผนการทำงานของเกม	10
2.2 ฟังก์ชันระบบแสดงการตัดเกรด	16
2.3 ฟังก์ชันโปรแกรมแสดงรายละเอียดการทำงานของโปรแกรมสำหรับตัดเกรด	16
2.4 แสดงฟังก์ชันขั้นตอนวิธีการทำงานของโปรแกรม	25
2.5 หน้า Interface โปรแกรม Adobe Photoshop CS3	29
2.6 Microsoft Office	29
2.7 การเรียนรู้ ActionScript	31
2.8 หน้าต่าง ActionScript	33
2.9 การทำงานของ ActionScript	35
2.10 การทำงานของ Class	36
2.11 การเก็บข้อมูลของ Class	37
2.12 การกำหนดชื่อให้กับ instance หรือ object	37
2.13 การแก้ไขชื่อ Class	38
2.14 ภาพแสดงโครงสร้างแบบต้นไม้กลับหัว โดยมี _root	41
2.15 method ของ MovieClip	52
2.16 ตัวอย่างการใช้งาน Events	56
2.17 ตัวอย่างการใช้งาน Events	57
3.1 ตัวอย่างหน้าต่างที่ใช้แสดงผลต่างๆของเกม เป้ายิงลูก	96
3.2 ตัวอย่างหน้าต่างที่ใช้แสดงผลต่างๆของเกมโดยรวมทั้งหมดของเกมที่ได้ศึกษา	97
3.3 ตัวอย่างหน้าต่างที่ใช้แสดงผลต่างๆของเกม แบบ Pop Up Screen โดยรวมทั้งหมดของเกมที่ได้ศึกษา	97
3.4 ตัวอย่าง Flow Chart ของเกมที่ได้ศึกษา	98
3.5 ตัวอย่างหน้าต่างที่ใช้แสดงผลต่างๆของเกม Sudoku ด้นแบบที่ศึกษา	99
3.6 ตัวอย่างหน้าต่างที่ใช้แสดงผลต่างๆของเกม Sudoku ที่แสดงวิธีการเล่น	100
3.7 หน้าต่างที่ใช้แสดงผลต่างๆของเกม Sudoku ที่ออกแบบ	100

รายการประกอบ (ต่อ)

3.8 ระบบการทำงานทั้งหมดของเกม Sudoku	101
3.9 ตัวอย่าง ตาราง Sudoku 3x3 ที่ได้ศึกษา	101
3.10 ตัวอย่างหน้า ต่างที่ใช้แสดงผลต่างๆของเกม เกม Sudoku 9x9 ในช่วงทดสอบ	103
3.11 หน้าต่างที่ใช้แสดงผลต่างๆของเกม เกม Sudoku หน้าเมนู	105
3.12 หน้าต่างที่ใช้แสดงผลต่างๆของเกม เกม Sudoku หน้า Level	105
3.13 หน้าต่างที่ใช้แสดงผลต่างๆของเกม เกม Sudoku หน้า Game เมื่อเลือก Level Normal	106
3.14 หน้าต่างที่ใช้แสดงผลต่างๆของเกม เกม Sudoku หน้า Game เมื่อต้องการใส่ตัวเลข	106
3.15 หน้าต่างที่ใช้แสดงผลต่างๆของเกม Sudoku หน้า Continue เมื่อเลือก Save	107
3.16 หน้าต่างที่ใช้แสดงผลต่างๆของเกม เกม Sudoku หน้า How to	107
ก.1 หน้าต่างโปรแกรม Adobe Flash CS3	116
ก.2 หน้าการทำงานหลักของโปรแกรม	117
ก.3 แถบเครื่องมือหลัก	117
ก.4 แถบเครื่องมือ (Toolbox)	118
ก.5 Document Tab	119
ก.6 Timeline	119
ก.7 Layer	119
ก.8 ภาพ stage และการวางวัตถุบน Stage	120
ก.9 หน้าต่าง Properties	120
ก.10 รายการคำสั่งในเมนู Window	121
ก.11 Document Properties	123
ก.12 หัวข้อโปรแกรมที่ยังไม่มีการ Save	124
ก.13 การบันทึกในฟอร์แมต GIF	125
ก.14 ตัวอย่างการทำงานแบบ Interlace	126
ก.15 การบันทึกภาพวาดในฟอร์แมต JPEG	126
ก.16 การบันทึกผลงานของ Flash เป็นภาพเคลื่อนไหว	127
ก.17 การปรับขนาดของ Stage	129
ก.18 Properties Panel	130

รายการรูปประกอบ (ต่อ)

ก.19 ภาพตัวอย่างการปรับขนาด Stage ให้เหมาะสมกับหน้าจอ	130
ก.20 การแสดงข้อมูล , ปรับขนาด Grid	131
ก.21 เครื่องมือของ Flash	132
ก.22 รายละเอียดเกี่ยวกับส่วนควบคุมสีใน Toolbox	134
ก.23 รายละเอียดเกี่ยวกับส่วนควบคุมสีใน Toolbox	134
ก.24 หน้าต่างสี	135
ก.25 แสดงขั้นตอนการเลือกสี	135
ก.26 แสดงขั้นตอนการผสมสีใหม่	136
ก.27 การใช้ Properties Panel ในการเลือกสี	136
ก.28 Swatches	137
ก.29 ตัวอย่างการเลือกสีแบบ Fill Type	137
ก.30 ตัวอย่างการเลือกสีแบบ Fill Type Linear หรือ Radial	138
ก.31 การกำหนดจำนวน Gradient Pointer	138
ก.32 การ Add Swatch	139
ก.33 ตัวอย่างความโปร่งใสของวัตถุ	140
ก.34 วิธีการปรับค่าความโปร่งใสของสีวัตถุ (Alpha)	140
ก.35 เครื่องมือวาดภาพ	141
ก.36 สามารถกำหนดสีพื้น, สีเส้นขอบ และลักษณะของเส้นขอบจาก Properties	141
ก.37 ปรับแต่งลักษณะของเส้นขอบได้จาก Stroke Style Dialog Box	142
ก.38 ลบวัตถุด้วย Eraser Tool	143
ก.39 ลบวัตถุด้วย Eraser Tool	143
ก.40 ตัวอย่างการเลือกวัตถุในสภาวะปกติ	144
ก.41 ตัวอย่างการเลือกวัตถุในโหมดจุดเชื่อมต่อ	144
ก.42 ตัวอย่างการเลือกวัตถุที่มีรูปทรงอิสระ	145
ก.43 โครงสร้างของวัตถุที่ประกอบด้วยโครงสร้างย่อย 2 ส่วน	145
ก.44 รูปแสดง “จุด” อันเกิดจากการเลือกบางส่วนของวงกลม	145
ก.45 การเลือกพื้นที่วัตถุ	146
ก.46 การเลือกวัตถุโดยการลากคลุมพื้นที่	146

รายการรูปประกอบ (ต่อ)

ก.47 การเปลี่ยนรูปทรงของวัตถุ	147
ก.48 ตัวอย่างการเปลี่ยนรูปทรงวัตถุ	148
ก.49 แสดงตัวอย่างวัตถุในสถานะ Free Transform	149
ก.50 แสดงตัวอย่างการปรับรูปทรงวัตถุ	149
ก.51 แสดงตัวอย่างการปรับรูปทรงวัตถุแบบต่างๆ	150
ก.52 ตัวอย่าง Stage และวัตถุนอก Stage	150
ก.53 Align Panel	151
ก.54 คัดลอกลักษณะเส้นขอบวัตถุ	151
ก.55 รูปการ์ตูนที่เกิดจาก Shape หลายชิ้นประกอบรวมกัน	152
ก.56 รูปการ์ตูนที่เกิดจากปัญหาการย้าย Shape ต่างๆ	153
ก.57 รูปที่ผ่านการรวมกลุ่มแล้วจะมีเส้นขอบสีฟ้าล้อมรอบ	153
ก.58 รูปภาพในสถานะกลุ่ม	153
ก.59 รูปภาพที่ผ่านการ Break Apart แล้ว	154
ก.60 จอภาพเข้าสู่โหมดการแก้ไข Group	154
ก.61 ตัวอย่างการแก้ไข Group	155
ก.62 Library ที่มี Symbol	156
ก.63 หน้าต่างการเปลี่ยนวัตถุเป็น Symbol	157
ก.64 ขั้นตอนการกลับสู่หน้าจอการทำงานปกติ	158
ก.65 Instance Properties	160
ก.66 ส่วนประกอบของ เฟรมและ Timeline	161
ก.67 หน้าต่าง Timeline	162
ก.68 หน้าต่าง เฟรม	162
ก.69 ตำแหน่งของเฟรมปัจจุบันที่จะปรากฏบน Stage	163
ก.70 ตัวอย่างภาพเคลื่อนไหวแบบ Shape Tween	165
ก.71 การปรับค่า Tween ให้เป็น Shape Tween	166
ก.72 หน้าต่าง Layer	167
ก.73 ตัวอย่างการแสดงผลแบบโครงสร้าง	167
ข.1 หน้าเมนูของเกม	169

รายการรูปประกอบ (ต่อ)

ข.2 หน้าเลือก Level ของเกม	170
ข.3 ส่วนประกอบภายในหน้าเกม	171
ข.4 การเรียกปุ่มกดเพื่อใส่ค่า	172
ข.5 เมื่อตัวเลขซ้ำกัน	172
ข.6 ตัวเลขที่ไม่ซ้ำกัน	173
ข.7 หยุดเกม ก็จะไม่สามารถเล่นเกมได้และเวลาที่จะหยุดเดิน	173
ข.8 การแจ้งเตือนว่าชนะ	174
ข.9 หน้า How to	174



บทที่ 1 บทนำ

1.1 ชื่อและที่ตั้งของสถานประกอบการ

ชื่อของสถานประกอบการ บริษัท ดิจิกราฟท์ จำกัด ที่ตั้งของสถานประกอบการ ตั้งอยู่ที่ เลขที่ 54 อาคาร BB ชั้น 14 ห้อง 1402 ถนนสุขุมวิท 21 (อโศก) แขวงคลองเตยเหนือ เขตวัฒนา กรุงเทพมหานคร 10110

1.2 ลักษณะธุรกิจของสถานประกอบการ

บริษัท ดิจิกราฟท์ จำกัด เป็นบริษัท 1 ใน จำนวน 6 บริษัทในเครือกลุ่มบริษัทอินสตีปจำกัด ซึ่งก่อตั้งและบริหารโดยคนไทย

ทั้ง 6 บริษัทมีแนวการประกอบธุรกิจ ดังนี้

1. บริษัท อินสตีป จำกัด

ดูแลธุรกรรมต่างๆที่เกี่ยวข้องในการดำเนินกิจการในประเทศไทย , ดูแลสวัสดิการ, การประกันสุขภาพ, การรับเงินเดือนและการจ่ายภาษี ตลอดจนงานต่างๆ ของพนักงานในเครือกลุ่มบริษัทอินสตีปจำกัด ทั้งหมด

2. บริษัท อินโนวา ซอฟต์แวร์ จำกัด

ประกอบธุรกิจประเภทงานออกแบบและพัฒนา รวมถึงการให้คำปรึกษาเกี่ยวกับระบบฐานข้อมูล ซอฟต์แวร์สำหรับอุตสาหกรรม, งานวิศวกรรม และธุรกิจทั่วไป ดูแลลูกค้าเช่นการรถไฟแห่งประเทศไทย, การรถไฟแห่งประเทศไทย, องค์การโทรศัพท์, การปิโตรเลียมแห่งประเทศไทย, บริษัทอิตาชิ ประเทศไทย เป็นต้น

3. บริษัท โจวิท จำกัด

ประกอบธุรกิจประเภทงานออกแบบและพัฒนา รวมถึงการให้คำปรึกษาเกี่ยวกับระบบฐานข้อมูล ซอฟต์แวร์ประเภทสองมิติและสามมิติ โปรแกรมและซอฟต์แวร์ต่างๆ มากมาย งานวิจัยทั้งภายนอกและภายในประเทศ และงานข้อมูลต่างๆ คู่มือลูกค้า เช่น บริษัท ช้างงาช้าง จำกัด และบริษัท ซีพียู จำกัด จากประเทศญี่ปุ่น

4. บริษัท ดีไซน์เกตเวย์ จำกัด

ประกอบธุรกิจประเภทงานออกแบบ พัฒนาและให้คำปรึกษางานวิจัยและพัฒนาภายในประเทศ เกี่ยวกับงานชิ้นส่วนอิเล็กทรอนิกส์ ตลอดจนผลิตชิ้นงานต้นแบบ คู่มือลูกค้าบริษัทกราฟฟิคและบริษัทไพรม์ ประเทศญี่ปุ่น เป็นต้น

5. บริษัท ดิจิกราฟท์ จำกัด

ดำเนินธุรกิจทางด้านข้อมูลดิจิทัล

6. บริษัท ไอแน็กซ์ จำกัด

จัดการด้านการตลาดให้กับสินค้าทั้งหมดของบริษัทในเครือ

ซึ่งบริษัท ดิจิกราฟท์ จำกัด ก่อตั้งจากการรวมทุนระหว่างผู้บริหารชาวไทย ร่วมกับบริษัท หลักทรัพย์จัดการกองทุนรวม วรรณ จำกัด และนักธุรกิจจากประเทศญี่ปุ่น เมื่อปี พ.ศ. 2547 โดยมี วัตถุประสงค์เพื่อผลิตงานทางด้าน Digital Content เช่น

1. งานจัดทำข้อมูลดิจิทัล และนำเสนอออกมาในรูปแบบต่างๆ เช่น E -Books, Modeling and Animation

2. งานพัฒนาเกม โดยแบ่งเป็น 2 หมวดใหญ่ๆ คือ

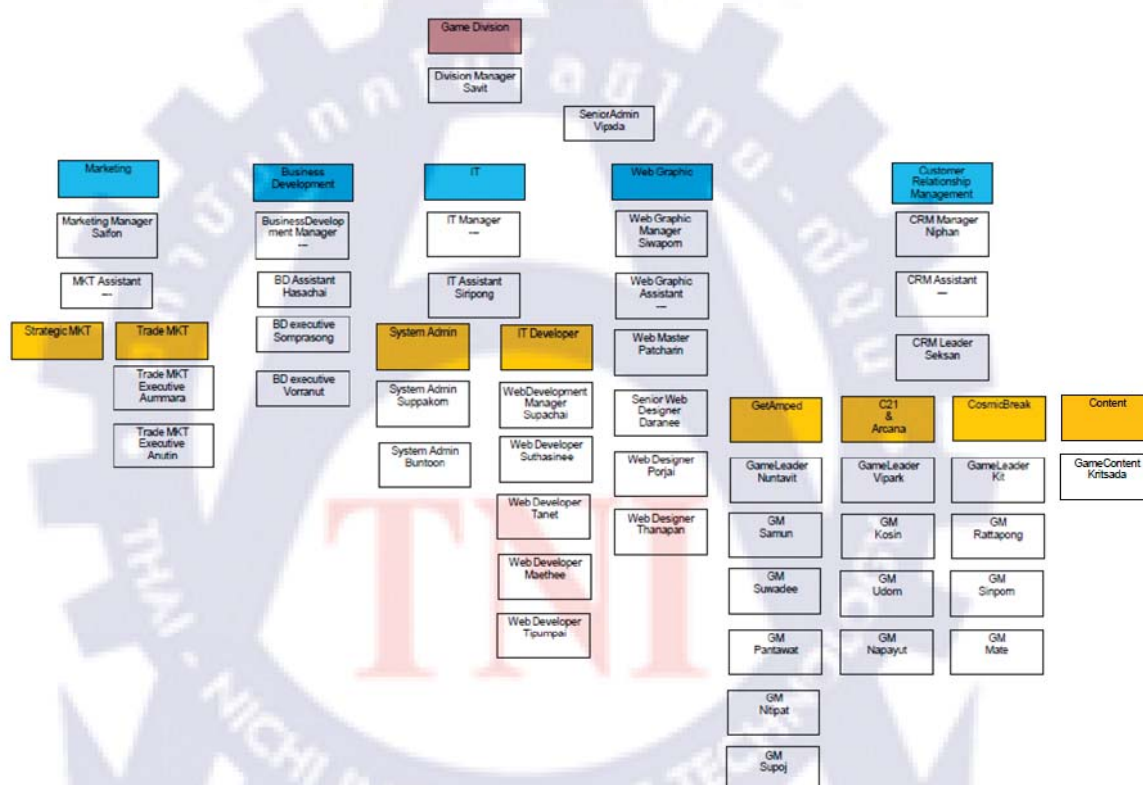
2.1 เกมออนไลน์ เช่น Arcana Online, Arcana Battle Card Online

2.2 เกมบนโทรศัพท์เคลื่อนที่ในระบบ J – Phone ซึ่งให้บริการในประเทศญี่ปุ่น

3. การเป็นตัวแทนลิขสิทธิ์แต่เพียงผู้เดียวในประเทศไทย เพื่อการเผยแพร่เกม GetAmped จากประเทศญี่ปุ่น

4. การพัฒนาเครื่องมือต่าง ๆ เพื่อช่วยในการผลิต Digital Content และการผลิตเกม

1.3 รูปแบบการจัดองค์กรและการบริหารงาน



ภาพที่ 1.1 ภาพแสดงผังการบริหารองค์กร

1.4 ตำแหน่งและหน้าที่งานที่นักศึกษาได้รับมอบหมาย

- ตำแหน่งงาน ผู้พัฒนาและออกแบบเกม 2 มิติ (ด้วยโปรแกรม Adobe Flash)
- หน้าที่ เรียนรู้ ออกแบบ และพัฒนาเกม 2 มิติ

1.5 พนักงานที่ปรึกษาและตำแหน่งของพนักงานที่ปรึกษา

พนักงานที่ปรึกษา นายศิริพงศ์ ชยก่อง ตำแหน่ง IT Assistant Manager

1.6 ระยะเวลาที่ปฏิบัติงาน

วันที่ 1 มิถุนายน 2553 ถึง 30 กันยายน 2554 รวมระยะเวลาทั้งสิ้น 4 เดือน

1.7 วัตถุประสงค์หรือจุดมุ่งหมายของการปฏิบัติงาน

วัตถุประสงค์คือการเรียนรู้การใช้งานโปรแกรม Adobe Flash ขั้นต้น การทำเกมขั้นต้น การวางระบบเกม องค์ประกอบต่างๆของเกม และการสร้างเกม 2 มิติให้แก่ บริษัท ดิจิคราฟต์ จำกัด ซึ่งเป็นเกม Sudoku ที่เน้นการคำนวณและฝึกทักษะการคิดทางตรรกะให้แก่ผู้เล่น

1.8 ผลที่คาดว่าจะได้รับจากการปฏิบัติงาน

ผลที่คาดว่าจะได้รับจากการปฏิบัติโครงการในครั้งนี้คือ ได้เรียนรู้ด้าน Adobe Flash และ Action script 2.0 ในการออกแบบ และพัฒนาเกม 2 มิติ และยังสามารถนำความรู้ตรงนี้ไปใช้ในการจัดทำงานอื่นๆอีกต่อไป

บทที่ 2 ทฤษฎีและเทคโนโลยีที่ใช้ในการปฏิบัติงาน

2.1 ทฤษฎีหลักการและเทคโนโลยีที่ใช้ในการพัฒนาโปรแกรม

2.1.1 ทฤษฎีเกี่ยวกับเกมคอมพิวเตอร์

เกม คือ โปรแกรมชนิดหนึ่งที่มีจุดประสงค์เพื่อดึงดูดให้ผู้ใช้หรือผู้เล่นตอบสนองต่อเหตุการณ์ต่างๆที่เกิดขึ้นและแก้ปัญหาให้บรรลุตามวัตถุประสงค์ที่กำหนดไว้ หรืออาจกล่าวได้ว่า เกม ก็คือ รูปแบบของการพัฒนาโปรแกรมอีกประเภทหนึ่งซึ่งมีจุดประสงค์เพื่อให้ความบันเทิงกับผู้ใช้หรือผู้เล่น โดยโปรแกรมเกมจะใช้การสร้างสรรค์ เทคโนโลยี และระบบมัลติมีเดีย เช่น ภาพ เสียง และการเคลื่อนไหวต่างๆ ผสมผสานกันเพื่อให้โปรแกรมดึงดูดใจและน่าสนใจมากที่สุด โดยทั่วไปการกำหนดประเภทของเกมจะใช้ลักษณะของเกม การดำเนินเรื่อง และวิธีการเล่นเป็นตัวกำหนด

2.1.1.1 เกมนิยมแบ่งเป็นประเภทต่างๆดังนี้

1) Action

ถือว่าเป็นเกมในรูปแบบแรกของวงการเกมเลยก็ว่าได้ อันเนื่องมาจากคำว่า แอคชั่น ก็คือ การกระทำต่าง ๆ นั้นเอง เมื่อใดก็ตามที่เราสามารถบังคับสั่งการตัวละครของเราโดยตรงให้กระทำการใดๆตอบโต้กับสิ่งต่างๆในเกมได้เกมๆนั้นก็ถือว่าเป็นเกม แอคชั่นแล้ว (แต่รายละเอียดย่อยบางอย่างจะแบ่งรูปแบบเกมออกไปอีก) แต่ถ้าจะระบุให้ชัดเจนกว่านั้น ลักษณะของเกมแอคชั่นก็คือ เกมในสไตล์ ลุย ยิง ต่อย เตะ หรืออื่นๆอีกมากมายที่สนองความสนใจให้กับผู้เล่น รูปแบบของเกมแอคชั่นโดยทั่วไปแล้วจะเน้นความเรียบง่ายและเข้าถึงได้ง่ายไม่ว่าจะเป็นมือใหม่หรือมือเก่า ก็สามารถทำความเข้าใจรูปแบบการเล่นของเกมแนวๆนี้ได้ไม่ยาก ยกตัวอย่างเช่น Kingkong , The Suffering , Call Of Duty , Rockmanx8

2) Adventure

รูปแบบของเกม ADV จะมีหลายๆส่วนที่คล้ายกับแอคชั่น (เพราะเกมแอคชั่นส่วนใหญ่ก็มักจะใส่ความเป็น ADV ไปด้วย) แต่ถ้าจะระบุจุดเด่นที่ชัดเจนที่สุดของเกม Adventure ก็คือการผจญภัยเกมแอคชั่นอาจเป็นเกมที่ไม่มีการดำเนินเรื่องราวด้วยตัวเราเองเลยก็ได้ อย่างเช่นเกมต่อสู้ในซีรีส์ Street fighter แต่ถ้าหากเป็นเกม ADVENTURE จะต้องมีส่วนของการผจญภัย การดำเนินเรื่องราวและแก้ปัญหาปริศนาต่างๆในเกมที่เรียกว่า Puzzle เกม จะเน้นที่การกระทำเพื่อแก้ปัญหา มากกว่าโดยอาศัยกลไกและสภาพแวดล้อมต่างๆในเกมให้ผู้เล่นได้ทำการปฏิสัมพันธ์เพื่อ

แก้ปัญหา ต้องคอยสังเกตสิ่งต่างๆ ในเกม อย่างเช่น ถ้ามีศัตรูอยู่ข้างหน้า เกม แอคชั่นก็จะมีบทสรุปที่ค่อนข้างชัดเจนคือ "เข้าไปจัดการมันซิ " แต่เกม Adventure อาจต้องทำอะไรมากกว่านั้น อย่างเช่น ไปเปิดกลไกตรงนั้นตรงนี้ อ้อมไปทางนั้นทางนี้เก็บ Item อันนั้นมาใส่ตรงนี้ หรืออื่นๆอีกเพื่อที่จะจัดการศัตรูตัวนั้น ยกตัวอย่างเช่น MYST , Sybira , Fahrenheit

3) Puzzle

Puzzle Game หรือ Logic Game เป็นเกมที่ต้องใช้ความคิดเพื่อแก้ไขปัญหาต่างๆ โดยเกมประเภทนี้ไม่ได้ใช้เพียงสายตาและมือเท่านั้น แต่จะต้องใช้ความคิดในการแก้ปัญหาทางลอจิกอีกด้วย ตัวอย่างของเกมประเภทนี้ คือ Tetris และเกมหมากรุก (Chess)

4) Simulation

หรือเกมจำลองสถานการณ์หรือเหตุการณ์ต่างๆนั้นเอง เกม SIM มีความหมายบางส่วนที่คล้ายกับRPG คือเป็นการสวมบทบาทเหมือนกัน แต่ต่างกันตรงที่ RPG จะสวมบทบาทตัวละครหนึ่งๆ (อาจเป็นคนหรืออะไรก็ได้ และอาจบังคับพร้อมๆกันหลายคนหรือเปลี่ยนกันบังคับก็ได้) แต่ SIM จะกว้างกว่านั้นเช่นเป็นผู้บริหารประเทศบริหารเมือง จัดการสวนสัตว์ สวนน้ำ หรือสั่งการตัวละครที่เป็นคนจริงๆ ในการดำเนินชีวิตประจำวัน (SIM 2) หรือจำลองรูปแบบอื่นๆอีกเช่นเป็นนักบินคนขับรถถัง เรือดำน้ำ นักแข่งรถ ตัวอย่างของเกมแนว SIMULATION Simcity4 ,The Sim2 ,Falcon4.0

5) Sport

ความหมายชัดเจนในตัวเลย เกม Sports ก็คือเกมกีฬา เกมอะไรก็ตามที่เป็นเกี่ยวกับกีฬาไม่ว่าจะเป็นฟุตบอล บาสเกตบอล เทนนิส ก็ถือว่าเป็นเกมกีฬาทั้งนั้นเพียงแต่อาจประกอบย่อยในส่วนของ Sport จะมีความแตกต่างกันในด้านของรูปแบบการเล่น โดย sport ที่บังคับในสนามจริงๆ สั่งการเหมือนกับที่เราเป็นผู้เล่นเองก็อย่างเช่นพวก Wining Eleven9,FIFA2006,NBA Live 2006 หรือบางเกมก็เป็นเกมวางแผนที่ให้เราจัดการรายละเอียดต่างๆเหมือนกับที่เราเป็นผู้จัดการเอง เช่น Football manager2006 ,CM5 ,FIFA MANAGER

6) Strategy

หรือเกมวางแผนนั้นเอง แยกย่อยออกได้เป็น Real Time Strategy และ Turn Base Strategy

6.1 Real Time Strategy หรือที่เรียกกันย่อๆว่า RTS เป็นเกมวางแผนแบบที่จับเวลาจริงในขณะที่เล่น คือในระหว่างที่เรากำลังเล่นอยู่ (วางแผน จัดรูปแบบทัพ สร้างฐาน หรืออะไรก็ตาม) ฝ่ายตรงข้ามไม่ว่าจะเป็น Player (ผู้เล่นด้วยกัน) หรือ AI (คอมพิวเตอร์บังคับ) ก็จะเล่นไปพร้อมๆกับเรา ดังนั้นลักษณะการเล่นแบบนี้คือการวางแผนแข่งกับเวลา(ส่วนใหญ่) เพราะในระหว่างที่เรา กำลังวางแผนจัดการเรื่องต่างๆอยู่ฝ่ายตรงข้ามก็กำลังจะทำเหมือนกับเรา โดยเกมวางแผนแบบ

RTS ถือเป็นเกมวางแผนที่ได้รับความนิยมมากที่สุดในตอนนี้ (จริงๆก็ได้รับความนิยมมานานแล้ว) อันเนื่องมาจากรูปแบบการเล่นที่เห็นผลของการวางแผนได้รวดเร็วชัดเจนตอบ สนองและเห็นผลเป็นรูปธรรมในการวางแผนของเราได้ทันที (เช่น สร้างยุ닛พร้อมเมื่อไหร่ก็ใช้ลุยกับศัตรูได้ทันที) รูปแบบการเล่นไม่ยืดเยื้อ (เพราะเป็นการวางแผนไปพร้อมๆกัน) สามารถเล่นพร้อมกันทีเดียวได้หลายคน

6.2 Turn Base Strategy (TBS)หรือเกมวางแผนแบบผลัดกันเดิน หลายๆท่านอาจจะไม่เคยเห็นหรือไม่รู้จักเกมแนวนี้มากนัก อันเนื่องมาจากเกม TBS ถือเป็นเกมวางแผนสไตล์ดั้งเดิมที่มานานแล้วมาๆ (Dune2 ต้นกำเนิดของเกม RTS เพิ่งจะมีมาราวๆ10กว่าปีโดยประมาณครับ) TBS เป็นเกมวางแผนที่รับแรงคลาไทม์มาจากเกมกระดานกันนั่นเอง (เกมกระดานคือเกมที่เราเอาแผนที่วางไว้บนโต๊ะแล้วคิดรูปแบบการเล่นขึ้นมาในกระดานหรือแผนที่นั้นๆ อย่างเช่น ผลัดทอยลูกเต๋า แล้วเดินไปตามจุดเพื่อเจอกับ เหตุการณ์ในแต่ละช่อง) ดังนั้นส่วนสำคัญที่ทำให้ TBS ต่างจากเกมแบบอื่นๆคือรูปแบบการเล่น TBS จะใช้การผลัดกันเล่นโดยเมื่อถึง Turn เรา(Turn เราก็คือ Turn ที่เรามีสิทธิ์จะวางแผนและจัดการในด้านต่างๆ) เราจะทำการจัดการวางแผนตามรูปแบบที่เกมกำหนดมา อย่างเช่น ใช้ยุ닛เดินไปช่องนั้นช่องนี้เพื่อเก็บทรัพยากร(ยุ닛จะมีค่าในการเคลื่อนไหวไปตามแผนที่) จัดการสร้างสิ่งก่อสร้างหรือยุ닛ต่างๆ โดยเมื่อเราทำตามทุกอย่างที่ต้องการหมดแล้วก็ต้องกดจบ Turn แล้วก็จะเข้า Turn ผู้เล่นคนอื่น หรือคอม ซึ่งก็จะทำการวางแผนและจัดการด้านต่างๆ เช่นเดียวกับเรา เมื่อจัดการทุกอย่างพอดจบ Turn ก็จะเข้า Turn ของเรา สลับผลัดเปลี่ยนไปแบบนี้เรื่อยๆ จนกว่าจะถึงเงื่อนไขที่กำหนดในการสิ้นสุดเกม (เช่น จัดการศัตรูทั้งหมด ได้ทรัพยากรเท่านั้นเท่านี้) ตัวอย่างของเกมแนวนี้ Civilization 4 , Hero Of Might And Magic4 , SilintStrom,Worm4

7) Role Playing Game

หรือที่เราคุ้นหูกันดีกับตัวย่อว่า RPG นั่นเอง เกม RPG ถือเป็นหนึ่งในแนวเกมที่ค่อนข้างจำแนกในลำดับกในขณะนี้อันเนื่องมาจากรูปแบบเกมสมัยใหม่ที่มีมักจะใส่แนวการเล่นไปมากกว่า 1 เสมอ แต่ถ้าจะเอาตามรากฐานของศัพท์และความหมายแต่ดั้งเดิมแล้ว RPG คือเกมที่เราจะได้สวมบทบาทของตัวละครนั้นๆนั่นเอง แต่เดิมที่เกมประเภท Action หรือ Adventure ถึงเราจะได้บังคับตัวละครก็จริง แต่จะเป็นไปในลักษณะสั่งหุ่นยนต์ให้ทำตามที่เราสั่งมากกว่า แต่ RPG จะลงลึกไปในรายละเอียดตัวละครมากกว่านั้นที่เด่นชัดที่สุดคือรูปแบบของการ พัฒนาตัวละครที่ RPG ทุกเกมจะต้องมีไม่ว่าจะมีในรูปแบบใดก็ตามเมื่อเราเล่น ไปเรื่อยๆ ตัวละครเราจะพัฒนาความสามารถไปเรื่อยๆ หลายๆเกมก็เปิดกว้างให้เราปรับแต่งรายละเอียดทุกอย่างของตัวละครได้แบบลงลึกไปถึงค่า Statusของตัวละครเช่น ความแข็งแกร่ง ความเร็วพลังป้องกัน ความสามารถเฉพาะตัวด้านต่างๆ

อย่างที่ 2 คือ เรื่องราวภายในเกมคงจะกล่าวไม่ผิดนักถ้าจะบอกว่าเกมที่จะทำให้เราเข้าใจโลกของเกมนั้นๆ ได้อย่างชัดเจนเข้าถึงโดยตรงที่สุด หนึ่งในนั้นจะต้องมีเกม RPG อยู่แน่นอนเพราะ RPG จะลงลึกไปถึงขนาดที่ว่าเราไปเป็นผู้ดำเนินเรื่องราวต่างๆ ด้วยตัวเอง (หลายๆ เกมให้อิสระที่จะเปลี่ยนแปลงเหตุการณ์ต่างๆ ในเกมได้ด้วยตัวเองด้วย) คำพูดและบทสนทนาต่างๆ จะทำให้เราเข้าใจโลกของเกมมากขึ้น อย่างเช่นการคุยกับ NPC (ตัวละครที่ควบคุมโดยคอม) คนหนึ่งเราก็อาจจะได้ข้อมูลเกี่ยวกับเรื่องราวต่างๆ ในโลกของเกมๆ นั้น การดำเนินเนื้อเรื่องของบางเกมจะมีการกิจต่างๆ ให้เราทำด้วย ซึ่งเป็นการตอกย้ำถึงการสวมบทบาทไปที่ตัวละครนั้นๆ อย่างชัดเจน ตัวอย่างของเกม RPG Diablo2, Final Fantasy, Dungeon Siege2

8) Racing

หรือเกมขับรถหรืออื่นๆ ที่เกี่ยวกับการแข่งด้านความเร็วเช่น แข่งสกี ขับยาน(เพียงแต่การขับรถจะมากกว่าประเภทอื่นๆ ความจริงเกมแนวนี้จะอยู่ในจำพวกเดียวกับ Sports ก็ได้เพราะหลายๆ คนก็ถือว่าการแข่งรถเป็นกีฬาอย่างหนึ่งแต่ Racing เองอาจจะเป็นการแข่งรถที่อิสระไม่เกี่ยวกับกีฬาใดๆ ก็ได้ อย่างเช่น Need For Speed เกมขับรถตระเวนราตรีในใจหลายๆ คนรูปแบบของเกม Racing โดยรวมๆ แล้วจะเกี่ยวกับการแข่งกันด้านความเร็ว และการบังคับรถ (แต่ก็อาจมีรูปแบบอื่นๆ ที่ไม่ใช่แข่งความเร็วก็ได้) อย่างเช่นแข่งกันทำคะแนน แข่งกันทำท่าผาดโผน

จุดประสงค์ที่สำคัญของการกำหนดประเภทเกมคือ เพื่อตอบสนองความต้องการของผู้เล่นที่เป็นกลุ่มเป้าหมาย ดังนั้นในอุตสาหกรรมเกมจึงต้องมีการกำหนดประเภทของเกมเพื่อเจาะไปยังกลุ่มเป้าหมายต่างๆ

2.1.1.2 มุมมองของเกม (Game View)

มุมมองของเกม (Game View) คือ มุมมองที่ใช้ถ่ายทอดภาพของเกมสู่สายตาของผู้เล่น โดยมุมมองต่างๆ ที่นิยมนำมาใช้เพื่อถ่ายทอดภาพสู่สายตาของผู้เล่นมีหลายรูปแบบ ดังนี้

1) First Person

เป็นมุมมองที่นิยมนำไปใช้กับเกมประเภท Shooting โดยจะใช้วิธีถ่ายทอดภาพจากสายตาของตัวละครที่ผู้เล่นควบคุมหรือบุคคลที่ 1 ตัวอย่างของเกมที่ใช้มุมมองแบบ First Person คือ Half-Life และ Quake

2) Isometric

เป็นมุมมองแบบ 3 มิติที่นิยมนำไปใช้กันอย่างแพร่หลายเพราะสามารถสร้างเกมแบบ 3 มิติได้โดยไม่ต้องใช้เทคนิคการเขียนโปรแกรมแบบ 3 มิติขั้นสูง ตัวอย่างของเกมที่ใช้มุมมองแบบ Isometric คือ Diablo

3) Side

เป็นมุมมองที่นิยมใช้ในการพัฒนาเกมแบบ 2 มิติ โดยจะถ่ายทอดภาพจากสายตาของบุคคลที่ 3 (Third Person) โดยใช้มุมมองทางด้านข้าง ตัวอย่างของเกมที่ใช้มุมมองแบบ Side คือ Mario

4) Top Down

มุมมองแบบ Top Down เป็นการถ่ายทอดภาพจากทางด้านบนหรืออาจเรียกมุมมองแบบนี้ว่า “Bird Eye View” นิยมใช้กับเกมแบบ 2 มิติ ที่มีรูปแบบง่ายๆ ตัวอย่างของเกมที่ใช้มุมมองแบบ Top Down คือ Pac Man

2.1.1.3 ปัจจัยสำคัญในการออกแบบและพัฒนาเกม

เกมที่ได้รับความนิยมและดึงดูดผู้เล่นไม่จำเป็นต้องเป็นเกมที่มี กราฟฟิคที่สวยงามเพียงอย่างเดียว แต่ขึ้นอยู่กับองค์ประกอบและจุดมุ่งหมายของเกมด้วย เช่น การออกแบบเนื้อหา จุดประสงค์ของเกม และกลุ่มผู้เล่น เป็นต้น ดังนั้นก่อนจะเริ่มพัฒนาเกมควรพิจารณาองค์ประกอบให้ ซึ่งหลักการสำคัญต่างๆในการออกแบบเกมที่ต้องพิจารณา มีดังนี้

1) อายุและเพศของผู้เล่น

ปัจจัยสำคัญที่ต้องพิจารณาก่อนการออกแบบเกมคือ อายุและเพศของกลุ่มเป้าหมาย ถ้าเป็นเกมที่มีกลุ่มเป้าหมายเป็นเด็กควรออกแบบเกมให้มีรูปแบบที่เข้าใจง่ายไม่ซับซ้อน เช่น ถ้ากลุ่มเป้าหมายเป็นเด็กอายุระหว่าง 6-12 ปี ควรออกแบบเกมโดยใช้กราฟฟิค ของตัวการ์ตูนเป็นหลัก ในกรณีที่ต้องการออกแบบเกมไว้บน Web Site ควรจะศึกษาอายุและเพศของผู้ที่เข้าเยี่ยมชม Web Site ว่าอยู่ในกลุ่มใด เพื่อสามารถทำให้ออกแบบเกมได้ตรงกับกลุ่มคนที่เข้ามาเยี่ยมชมมากที่สุด โดยกลุ่มอายุของผู้เล่นสามารถแบ่งได้เป็น 6-12 ปี, 9-14 ปี, 12-17 ปี, 18-34 ปี, 35-44 ปี, 45-55 ปี และ 55 ปีขึ้นไป นอกจากอายุของผู้เล่นแล้วปัจจัยสำคัญอีกอย่างที่ต้องพิจารณาคือ เพศของผู้เล่น เพราะผู้เล่นที่เป็นผู้ชาย, ผู้หญิง, เด็กชายและเด็กหญิงนั้นมีมุมมองที่แตกต่างกัน เช่น โดยทั่วไปผู้ชายมักจะชอบเกม Action ที่สมจริงและเนื้อหารุนแรงมากกว่าผู้หญิง เป็นต้น

2) ประสิทธิภาพของคอมพิวเตอร์

ประสิทธิภาพของคอมพิวเตอร์เป็นปัจจัยสำคัญที่ทำให้การแสดงผลภาพเคลื่อนไหวและกราฟฟิคต่างๆ ภายในเกมดูสวยงามและนุ่มนวลมากขึ้น โดยเฉพาะการพัฒนาเกมประเภท Action จะต้องใช้ทรัพยากรของเครื่องคอมพิวเตอร์มากกว่าปกติเนื่องจากเกมประเภท Action ต้องมีเสียงดนตรีและการเคลื่อนไหวของกราฟฟิคต่างๆ เป็นจำนวนมาก โดยจะต้องใช้ทรัพยากรของเครื่องคอมพิวเตอร์เพื่อกำหนดหาผลลัพธ์ต่างๆ ถ้าคอมพิวเตอร์มีประสิทธิภาพไม่สูงพอจะทำให้การแสดงผลช้าและเกิดความไม่ต่อเนื่องได้ ดังนั้นก่อนการออกแบบและพัฒนาเกมควรพิจารณา

ด้วยว่าเกมดังกล่าวต้องการใช้เครื่องคอมพิวเตอร์ที่มีประสิทธิภาพสูงเกินไปหรือไม่ เพราะไม่ว่าเกมที่พัฒนาขึ้นมาจะน่าสนใจและดีเพียงใด แต่ถ้าไม่สามารถทำงานบนเครื่องคอมพิวเตอร์ของผู้เล่นได้ย่อมไม่เกิดประโยชน์

2.1.1.4 การกำหนดวัตถุประสงค์และเนื้อหาของเกม

การกำหนดวัตถุประสงค์ที่เหมาะสมให้กับเกมจะทำให้เกมมีความน่าสนใจมากยิ่งขึ้น ตัวอย่างเช่น เกมแข่งรถที่มีวัตถุประสงค์หลักเพื่อให้ผู้เล่นควบคุมรถให้ถึงเส้นชัยก่อนรถคันอื่น แต่เราสามารถเพิ่มความน่าสนใจของเกมให้มากขึ้นได้โดยกำหนดวัตถุประสงค์ที่มีความสำคัญรองลงมา เช่น เก็บของที่อยู่บนถนนเพื่อสะสมคะแนน เป็นต้น

นอกจากการกำหนดวัตถุประสงค์ของเกมแล้ว อีกสิ่งหนึ่งที่ควรพิจารณาคือ เนื้อหาของเกม ตัวอย่างเช่น เกมแข่งรถที่ตัวละครหลักจะต้องขับรถไปปรับแฟนสาวให้ทันภายในเวลาที่กำหนด โดยระหว่างทางจะมีอุปสรรคต่างๆ เช่น ต้องหลบรถคันอื่นที่อยู่ด้านหน้า หลบของที่ตกจากรถบรรทุก หรือระวังคนที่เดินข้ามถนน เป็นต้น โดยสามารถขับรถชนสิ่งของหรือรถที่อยู่ด้านหน้าได้ไม่เกิน 5 ครั้ง ถ้าผู้เล่นสามารถควบคุมไปยังเส้นชัยได้ตามเงื่อนไขและเวลาที่กำหนด ก็จะแสดงภาพแฟนสาวของตัวละครขึ้นรถไปกับตัวละครหลัก แต่ถ้าไปปรับไม่ทันตามเวลาที่กำหนดตัวละครหลักก็จะโดนแฟนสาวบอกเลิก เป็นต้น

เมื่อกำหนดวัตถุประสงค์และเนื้อหาของเกมแล้ว ต่อไปจะกล่าวถึงแบบแผนการทำงานของ เกมโดยทั่วไปเกมจะใช้จอภาพต่างๆ เพื่อควบคุมลำดับการทำงานดังรูป



ภาพที่ 2.1 รูปแบบแผนการทำงานของเกม

รูปข้างต้นแสดงแบบแผนการทำงานของเกมโดยเริ่มต้นจากจอภาพ Loading... ซึ่งแสดงข้อความให้ผู้ใช้อรการโหลดข้อมูล เมื่อโหลดข้อมูลเสร็จแล้วจะแสดงจอภาพ Welcome เพื่อ

ต้อนรับผู้เล่นเข้าสู่ตัวเกมในจอภาพนี้อาจบอกเนื้อหาและแนะนำเกม จากนั้นจึงแสดงจอภาพ How to play เพื่อบอกวิธีการเล่นให้ผู้เล่นได้ทราบ เช่น บอกวิธีการบังคับ หรือวัตถุประสงค์ต่างๆของเกม เป็นต้น

เมื่อเข้าสู่จอภาพ How to play แล้ว ต่อไปผู้เล่นจึงเริ่มต้นเล่นเกม และเมื่อเกมสิ้นสุดลงจะเข้าสู่จอภาพ The End เพื่อบอกผู้เล่นว่าเกมได้สิ้นสุดลงแล้ว จากนั้นจึงกลับไปแสดงจอภาพ Welcome เพื่อเริ่มต้นเกมใหม่อีกครั้ง แบบแผนดังกล่าวเป็นเพียงตัวอย่างการกำหนดจอภาพต่างๆ ภายในเกมถ้าเกมที่พัฒนามีระดับความยากมากกว่า 1 ระดับ หรือมีโหมดพิเศษที่จะแสดงหลังจากเกมสิ้นสุดลง ผู้พัฒนาเกมสามารถนำแนวความคิดดังกล่าวไปประยุกต์ใช้เพื่อสร้างแบบแผนของตนเองได้

2.1.1.5 องค์ประกอบพื้นฐานสำหรับเพิ่มความน่าสนใจของเกม

1) Live and Health

Live หมายถึง จำนวนชีวิตของตัวละคร และ Health หมายถึง พลังชีวิตของตัวละคร ถ้าพลังชีวิตของตัวละครหมดลง เกมจะกลับเริ่มต้นใหม่ในจุดที่กำหนดและจะลดค่าของจำนวนชีวิตลง 1 แต่ถ้าจำนวนชีวิตหมดแล้วจะสิ้นสุดการเล่นและไม่สามารถเริ่มต้นเกมในจุดดังกล่าวได้ แต่จะต้องเริ่มต้นเกมใหม่ที่จุดเริ่มต้นใหม่และปรับค่าของ Live และ Health เป็นค่าเริ่มต้นเพื่อเริ่มต้นเกมใหม่

จำนวนและพลังชีวิตใช้เพื่อควบคุมระยะเวลาในการเล่น เนื่องจากเกมไม่สามารถดำเนินไปโดยไม่มีจุดสิ้นสุดได้ ดังนั้นจึงต้องคิดค้นวิธีใช้สำหรับกำหนดจุดสิ้นสุดของเกมขึ้นมา โดยสามารถใช้จำนวนและพลังชีวิตของตัวละครเพื่อกำหนดจุดสิ้นสุดของเกมได้ ถ้าจำนวนและพลังชีวิตของตัวละครหมดลงหมายความว่า เกมนี้สิ้นสุดลง หรือ Game Over

2) Score

ระบบคะแนนเป็นการวัดความสามารถและประสิทธิภาพของผู้เล่น นอกจากนี้ยังใช้เพื่อเปรียบเทียบความสามารถของผู้เล่นแต่ละคนได้ด้วย ระบบคะแนนจึงเป็นเครื่องมือที่ทำให้การเล่นเกมน่าสนใจยิ่งขึ้นระบบคะแนนนอกจากจะใช้เพื่อวัดประสิทธิภาพของผู้เล่นแล้วยังใช้สำหรับกำหนดค่าต่างๆของเกมด้วยเช่น ระดับความยากของเกม นอกจากนี้ยังสามารถใช้คะแนนเป็นวัตถุประสงค์ของเกมได้ด้วย เช่น ผู้เล่นที่ทำคะแนนได้สูงสุดคือ ผู้ชนะ เกมโดยทั่วไปจะใช้คะแนนเพื่อกำหนดระดับความยากของเกมโดยเริ่มต้นจากระดับง่าย เมื่อผู้เล่นสามารถสะสมคะแนนได้มากขึ้น ระดับความยากของเกมจะเพิ่มมากขึ้นตามไปด้วย

3) Level

Level คือการใช้เทคนิควนซ้ำเพื่อกระตุ้นให้ผู้เล่นเกิดความท้าทายใหม่ๆ ทำให้ผู้เล่นเกิดความพยายามและเล่นเกมนั้นซ้ำอีก ระดับความยากของเกมเป็นองค์ประกอบที่ทำให้เกมน่าสนใจมากขึ้น เพราะการปรับระดับความยากจะทำให้เกิดความท้าทายใหม่ๆ กับผู้เล่น โดยทั่วไประดับความยากจะเริ่มต้นจากระดับง่ายเพื่อสอนพื้นฐานการเล่นเกม จากนั้นระดับความยากจะเพิ่มขึ้นตามความสามารถและคะแนนของผู้เล่น แต่ละ Level จะมีระดับความยากและอุปสรรคต่างๆ เพิ่มขึ้น ทำให้ผู้เล่นสามารถผ่านฉากได้ยากขึ้นดังนั้นผู้เล่นจะต้องปรับปรุงและพัฒนาการเล่นมีประสิทธิภาพมากขึ้นเพื่อที่จะผ่านไปเล่นในระดับที่ยากขึ้น

4) Bonus and Power Up

Bonus and Power Up เป็นเครื่องมือที่เพิ่มความน่าสนใจให้กับเกมและเป็นองค์ประกอบสำคัญของเกมโดยสามารถนำไปใช้ได้หลายรูปแบบ เช่น แต่ละ Level จะมี Bonus เมื่อผู้เล่นสามารถเก็บของที่กำหนดไว้ได้หรือเมื่อผ่านแต่ละ Level แล้ว โปรแกรมจะ Power Up (หรือเพิ่มความสามารถ) ตัวละครให้มีประสิทธิภาพและความสามารถในการด้านต่างๆ เพิ่มขึ้น เป็นต้น

2.1.1.6 รูปแบบการแข่งขันกันของผู้เล่นเกม

ปัจจุบันเกมที่มีการแข่งขันกันระหว่างผู้เล่นกำลังได้รับความนิยมอย่างสูง เช่น Racknarok โดยจุดเด่นของเกมชนิดนี้คือ การแข่งขันกันเพื่อหาผู้ชนะระหว่างผู้เล่น โดยนักเล่นเกมจำนวนมากชื่นชอบการแข่งขันภายในกลุ่ม ดังนั้นการพัฒนาเกมชนิดนี้จึงเป็นรูปแบบที่ได้รับความนิยมอย่างมาก

1) High Score List

เป็นรูปแบบที่ใช้การจัดอันดับคะแนนของผู้เล่นและเป็นรูปแบบที่นิยมใช้กับเกม Arcade ลักษณะของการเล่นจะเป็นแบบคนเดียว (Single Player) โดยจะใช้วิธีเก็บคะแนนของผู้เล่น 10 อันดับแรกที่ทำคะแนนได้สูงสุด การพัฒนาเกมแบบ High Score List จะต้องใช้ Server Side Component เพื่อบันทึกข้อมูลไว้บน Server โดยจะดึงข้อมูลที่เป็นคะแนนของผู้เล่นมาจาก Server แล้วนำมาเปรียบเทียบกับคะแนนของผู้เล่นหลังจากเกมสิ้นสุดลง ถ้าคะแนนอยู่ในช่วงอันดับที่กำหนดไว้ก็จะเพิ่มคะแนนลงในรายการ ส่วนคะแนนที่อยู่ในอันดับต่ำสุดจะถูกลบออกจากรายการ

2) Real-Time Multiplayer

เป็นรูปแบบการแข่งขันกันของผู้เล่นเกมหลายคนในเวลาเดียวกันหรือ Real-Time โดยผู้เล่นแต่ละคนจะต้องแข่งขันกันเพื่อเป็นผู้ชนะ โดยทั่วไปเกมรูปแบบนี้จะเชื่อมต่อกันผ่านทางระบบอินเทอร์เน็ตความเร็วสูงและใช้ Server Side Component ในการส่งข้อมูลระหว่างผู้เล่น

3) Challenge Game

Challenge Game คือ เกมใช้วิธีการส่งข้อมูลซึ่งเป็นคะแนนที่ได้จากการเล่นเกมให้กับผู้เล่นคนอื่นแล้วใช้คะแนนเพื่อเปรียบเทียบความสามารถของผู้เล่นแต่ละคน โดยจะใช้ Server Side Component เพื่อส่งข้อมูลซึ่งเป็นคะแนนที่ได้หลังจากเกมสิ้นสุดและจะใช้วิธีการส่งคะแนนไปยังผู้เล่นอีกคนผ่านทาง E-mail ความแตกต่างระหว่าง Challenge Game กับ Multiplayer Game คือ Challenge Game จะเป็นการเล่นเกมของผู้เล่นแต่ละคนแบบ Single Player แล้วเมื่อเกมสิ้นสุดจะนำคะแนนส่งให้กับผู้เล่นคนอื่น ๆ แต่ Multiplayer Game ผู้เล่นทั้งหมดจะเล่นเกมพร้อมกันในเวลาเดียวกัน (Real Time) เพื่อหาผู้ชนะ

2.1.1.7 ประเภทของเกมคอมพิวเตอร์แบ่งออกเป็น 3 ประเภท ดังนี้

- 1) เล่นคนเดียว หมายถึง การเล่นเกมที่ผู้เล่นเป็นผู้แก้ปัญหาของเกมนั้นด้วยตัวเองโดยที่คอมพิวเตอร์ไม่ได้มีส่วนในการจัดการกับเกมนั้นเลย เช่น เกมพายุเรือในคลองคด ซึ่งผู้เล่นจะต้องใช้ความสามารถในการควบคุมเรือ จากจุดเริ่มต้นให้ไปถึงที่หมายโดยให้เรือกระทบกับฝั่งน้อยที่สุด เกมประเภทนี้ คอมพิวเตอร์ไม่ได้เข้ามามีส่วนร่วมในการเล่นเพียงแค่ทำหน้าที่เป็นเครื่องมือสำหรับเล่นเกมเท่านั้น
- 2) เล่นสองคน หรือเล่นแบบหลายคน (Multi Player) หมายถึง ผู้เล่นเกมมี 2 คน หรือมากกว่า แต่ละคนจะเล่นเพื่อให้ชนะคู่ต่อสู้ เช่น เกมชกมวย ซึ่งผู้เล่นแต่ละคนต่างพยายามชกให้ คู่ต่อสู้แพ้ เกมประเภทนี้คอมพิวเตอร์ทำหน้าที่เป็นเครื่องมือเล่นเกมเท่านั้น
- 3) เล่นกับคอมพิวเตอร์ หมายถึง การที่มีผู้เล่นเกมฝ่ายหนึ่งเป็นคน อีกฝ่ายหนึ่งเป็นเครื่องคอมพิวเตอร์ โดยฝ่ายที่เป็นคนอาจจะมีคนเดียว หรือสองคนก็ได้ เช่น เกมยิงศัตรูต่างดาว ซึ่งคอมพิวเตอร์จะมีหน้าที่สร้างมนุษย์ต่างดาวที่มีความสามารถต่าง ๆ ออกมาสู้กับคนฝ่ายคนก็จะต้องคอยยิงต่อสู้เพื่อป้องกันไม่ให้มนุษย์ต่างดาวเข้ามาในโลก ในกรณีเช่นนี้จะเห็นได้ว่าคอมพิวเตอร์ได้เข้ามามีบทบาททั้งในฐานะที่เป็น เครื่องมือเล่นเกมและเป็นผู้เล่นเกมด้วย

2.1.2 ความรู้พื้นฐานการเขียนโปรแกรม

2.1.2.1 ขั้นตอนวิธีการทางคอมพิวเตอร์ (computer algorithms)

ก่อนที่จะเราเขียนโปรแกรมคอมพิวเตอร์ภาษาใดภาษาหนึ่งเพื่อแก้ปัญหทางคอมพิวเตอร์ เราควรศึกษาขั้นตอนวิธีการทางคอมพิวเตอร์ เพื่อให้เกิดความรู้ความเข้าใจที่ถูกต้องตามขั้นตอนวิธีดังนี้

- 1) การวิเคราะห์งาน (job analysis)
- 2) การเขียนผังงาน โปรแกรม (program flowcharting)
- 3) การเขียนโปรแกรม (programming)
- 4) การทดสอบ และแก้ไขโปรแกรม (testing and editing program)
- 5) การจัดทำเอกสารประกอบ และการบำรุงรักษาโปรแกรม (documentation and maintenance program)

1) การวิเคราะห์งาน (job analysis)

ในการวิเคราะห์งานเราจะต้องกำหนดจุดประสงค์ของการวิเคราะห์งาน และขั้นตอนวิธีการวิเคราะห์งานแต่ละอย่างให้ได้ ซึ่งงานแต่ละอย่างมีรายละเอียดดังนี้

1.1) จุดประสงค์ของการวิเคราะห์งาน

ในการวิเคราะห์งานแต่ละอย่างมีจุดประสงค์ของการวิเคราะห์งานที่สำคัญดังนี้

1. เพื่อหาวัตถุประสงค์ของการเขียนโปรแกรม
2. เพื่อหารูปแบบผลลัพธ์ที่ต้องการ
3. เพื่อหาข้อมูลนำเข้าที่ต้องใส่เข้าไปในโปรแกรม
4. เพื่อหาตัวแปรที่จำเป็นต้องใช้ในโปรแกรม
5. เพื่อหาขั้นตอนวิธีการทำงานของโปรแกรม

1.2) ขั้นตอนวิธีการวิเคราะห์งาน

ในการวิเคราะห์งานแต่ละอย่างมีขั้นตอนวิธีการวิเคราะห์งานที่สำคัญดังนี้

1) การหาวัตถุประสงค์ของการเขียนโปรแกรม

ผู้เขียนโปรแกรมจะต้องหาวัตถุประสงค์จากงานที่จะเขียนโปรแกรมว่า ต้องการเขียนโปรแกรมเพื่อแก้ปัญหาอะไรบ้างซึ่งจะทำให้เขียนโปรแกรมได้ตรงกับความต้องการหรือวัตถุประสงค์ของงานนั้น ๆ

2) การหารูปแบบผลลัพธ์ที่ต้องการ

เมื่อผู้เขียนโปรแกรมหาวัตถุประสงค์ของการเขียนโปรแกรมได้แล้ว ขั้นตอนต่อมาคือการกำหนดรูปแบบผลลัพธ์ที่ต้องการจากโปรแกรม ซึ่งรูปแบบผลลัพธ์อาจอยู่ในลักษณะของข้อความ หรือตัวเลข หรือตาราง หรือแผนภูมิ หรืออาจใช้ผสมกันระหว่างตัวเลขกับข้อความ หรือข้อความกับตัวเลข และตารางก็ได้ ขึ้นอยู่กับผู้เขียนโปรแกรมเป็นผู้กำหนดเอง แต่โดยส่วนมากนิยมแสดงผลลัพธ์ของโปรแกรมให้อยู่ในรูปแบบที่เข้าใจง่ายมากกว่ารูปแบบที่ซับซ้อน

3) การหาข้อมูลนำเข้าที่ต้องใส่เข้าไปในโปรแกรม

ผู้เขียนโปรแกรมจะต้องหาข้อมูลนำเข้าจากผลลัพธ์ที่ได้จากโปรแกรม โดยคำนึงถึงขั้นตอนวิธีการคำนวณ และข้อมูลที่จำเป็นต้องใส่เข้าไปเพื่อให้ได้ผลลัพธ์ที่ต้องการ

4) การหาตัวแปรที่จำเป็นต้องใช้ในโปรแกรม

ตัวแปร หมายถึง ชื่อที่ผู้เขียนโปรแกรมสามารถตั้งขึ้นเองตามหลักการตั้งชื่อตัวแปรของภาษาคอมพิวเตอร์ที่นำมาเขียนโปรแกรม เพื่อใช้ในการอ้างอิงการเก็บข้อมูลและเรียกใช้ข้อมูลภายในตัวแปร ดังนั้นผู้เขียนโปรแกรมจะต้องตั้งชื่อตัวแปรที่ใช้ในการเก็บข้อมูลทั้งหมดภายในโปรแกรม รวมถึงตัวแปรบางตัวที่ใช้ในการนับจำนวนรอบของการทำงานในโปรแกรมอีกด้วย

5) การหาขั้นตอนวิธีการทำงานของโปรแกรม

การหาขั้นตอนวิธีการทำงานของโปรแกรมจะขึ้นอยู่กับลักษณะของการแก้ปัญหา ดังนั้นนักเขียนโปรแกรมที่มีความชำนาญในระดับหนึ่งแล้วจะสามารถหาขั้นตอนวิธีการทำงานของโปรแกรมได้โดยไมยาก แต่สำหรับผู้เริ่มต้นศึกษาการเขียนโปรแกรม ควรหาขั้นตอนวิธีการทำงานของโปรแกรมตามคำแนะนำโดยเรียงตามลำดับดังนี้

2) การเขียนผังงานโปรแกรม (program flowcharting)

สำหรับหัวข้อการเขียนผังงาน โปรแกรมจะประกอบด้วยเนื้อหาที่สำคัญดังนี้คือ ความหมายของผังงาน ประเภทของผังงาน สัญลักษณ์มาตรฐานที่ใช้เขียนผังงานโปรแกรม หลักการเขียนผังงานโปรแกรม และขั้นตอนการเขียนผังงานโปรแกรม ดังรายละเอียดต่อไปนี้

- ความหมายของผังงาน

ผังงาน (flowchart) ทางคอมพิวเตอร์ หมายถึง แผนภาพที่เขียนขึ้นโดยใช้สัญลักษณ์มาตรฐาน เพื่อแสดงขั้นตอนวิธีการทำงานของโปรแกรม หรือระบบงานที่ต้องการ

- ประเภทของผังงาน

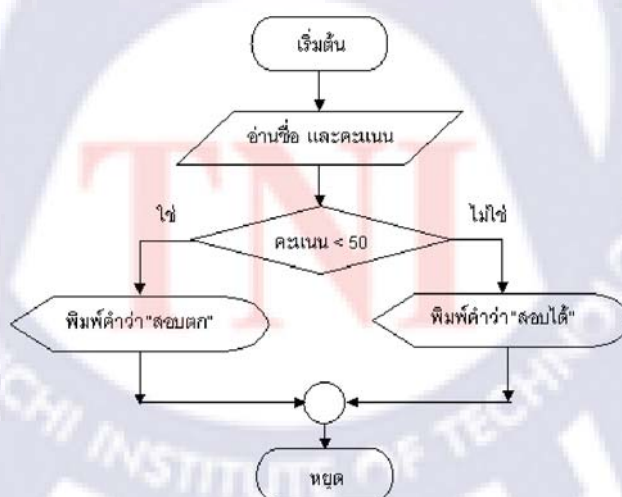
2.1) ผังงานสามารถจำแนกออกได้ 2 ประเภท คือ

1) ผังงานระบบ (system flowchart) หมายถึง ผังงานที่แสดงถึงขั้นตอนวิธีการทำงานของระบบงานใดระบบหนึ่ง ซึ่งในผังงานระบบจะแสดงให้เห็นถึงสื่อที่รับข้อมูล บันทึกข้อมูล วิธีการประมวลผล ขั้นตอนการทำงานและความสัมพันธ์ของส่วนต่างๆ ในระบบงานอย่างกว้างๆ เพื่อให้เห็นภาพรวมของระบบงานใดระบบงานหนึ่ง เช่น



ภาพที่ 2.2 ผังงานระบบแสดงการตัดเกรด

2) ผังงานโปรแกรม (program flowchart) หมายถึงผังงานที่แสดงขั้นตอน วิธีการทำงานของโปรแกรมที่ต้องการเขียนขึ้น ภายในผังงานโปรแกรม จะแสดงให้เห็นขั้นตอน วิธีการทำงานของโปรแกรมอย่างละเอียด เพื่อให้คอมพิวเตอร์สามารถทำงานตามที่เขียนไว้ในผังงานโปรแกรมได้อย่างถูกต้อง โดยเราสามารถนำขั้นตอน วิธีการทำงานของโปรแกรมในขั้นการวิเคราะห์งาน มาเขียนเป็นผังงานโปรแกรมได้ทันที จากนั้นก็สามารถนำผังงานโปรแกรม ไปเขียนโปรแกรมตามที่ต้องการได้ ดังนั้นในเรื่องของการเขียนผังงาน ที่จะกล่าวต่อไปนี้จะเกี่ยวข้องกับผังงานโปรแกรมทั้งสิ้น เช่น



ภาพที่ 2.3 ผังงานโปรแกรมแสดงรายละเอียดการทำงานของโปรแกรมสำหรับตัดเกรด

2.2) สัญลักษณ์มาตรฐานที่ใช้เขียนผังงานโปรแกรม

สัญลักษณ์ที่นิยมนำมาใช้เขียนผังงานโปรแกรม ดังต่อไปนี้

1. สัญลักษณ์ Process

Process

ความหมายของ Process คือ การกำหนดค่าหรือการคำนวณค่า

ตัวอย่างแสดงการกำหนดค่าให้ตัวแปร Total



ตัวอย่างรูปแสดงการคำนวณค่าผลรวมของตัวแปร Total

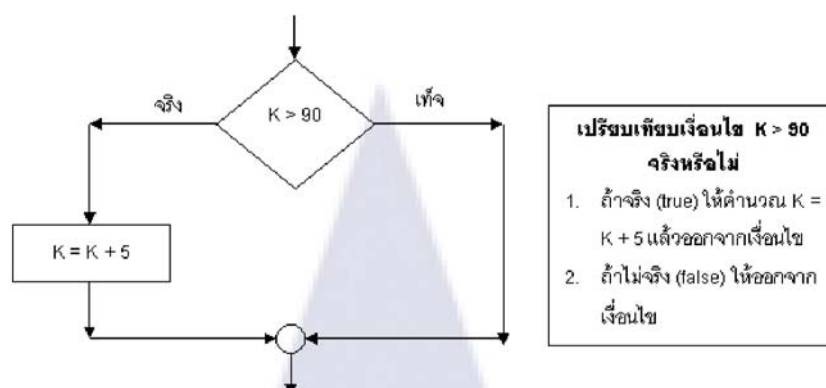


2. สัญลักษณ์ Decision

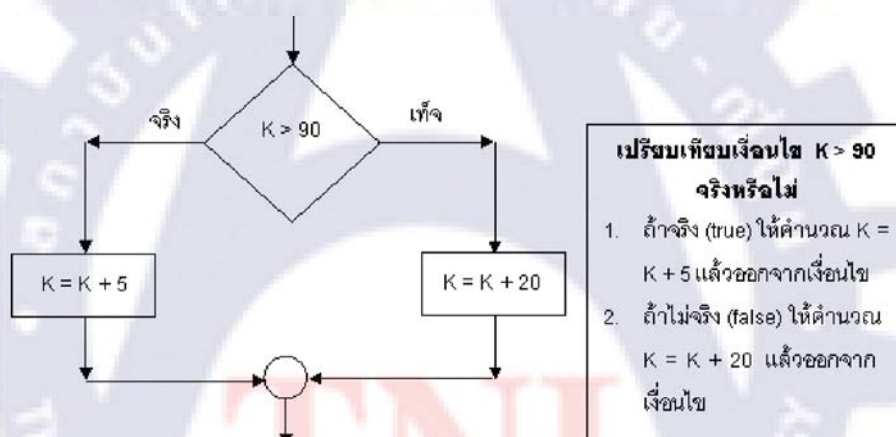
Decision

ความหมายของ Decision คือ การทดสอบเงื่อนไขหรือเปรียบเทียบเงื่อนไข เพื่อตัดสินใจทำงานอย่างใดอย่างหนึ่งตามที่ต้องการ

ตัวอย่างรูปแสดงการเปรียบเทียบเงื่อนไข 1 ทางเลือก



ตัวอย่างรูปแสดงการเปรียบเทียบเงื่อนไข 2 ทางเลือก



3. สัญลักษณ์ Start/ Stop program หรือ Terminal Interrupt

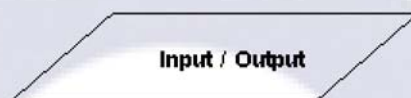
Start / Stop program

ความหมายของสัญลักษณ์ คือ การเริ่มต้นโปรแกรม หรือการสิ้นสุดการทำงานของโปรแกรม หรือหยุดการทำงานชั่วคราว

ตัวอย่างรูปแสดงการใช้งานสัญลักษณ์ Start/ Stop

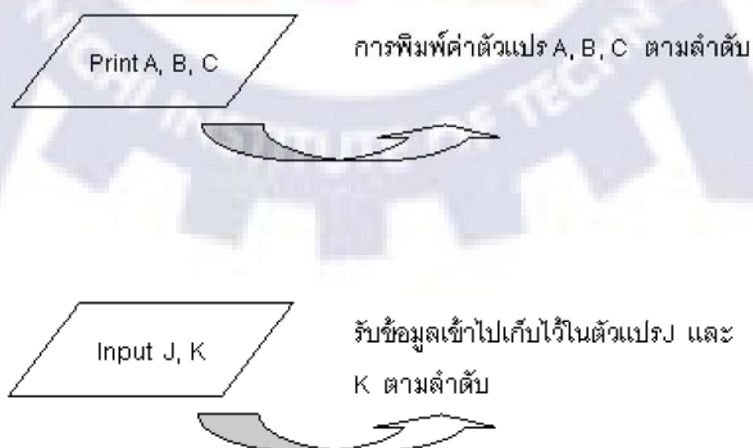


4. สัญลักษณ์ Input / Output



ความหมายของสัญลักษณ์ คือ การรับข้อมูล (input) หรือการแสดงผลลัพธ์ (output) โดยไม่ได้ระบุสื่อที่ใช้

ตัวอย่างรูปแสดงการใช้งานสัญลักษณ์ Input/ Output

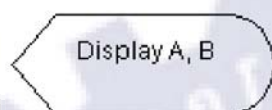


5. สัญลักษณ์ Display



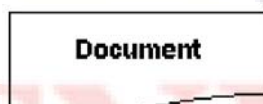
ความหมายของสัญลักษณ์ คือ การแสดงผลลัพธ์ทางจอภาพ

ตัวอย่างรูปแสดงการใช้งานสัญลักษณ์ display



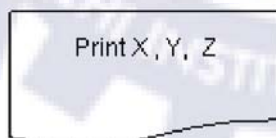
แสดงค่าตัวแปร A, B ออกทางจอภาพ
ตามลำดับ

6. สัญลักษณ์ Document



ความหมายของสัญลักษณ์ คือ การพิมพ์ผลลัพธ์ออกทางกระดาษพิมพ์

ตัวอย่างรูปแสดงการใช้งานสัญลักษณ์ Document



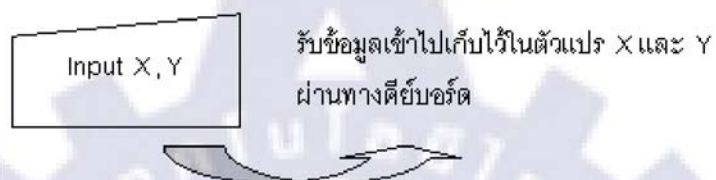
พิมพ์ค่าตัวแปร X, Y, Z ออกทาง
กระดาษพิมพ์ตามลำดับ

7. สัญลักษณ์ Manual Input

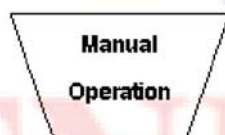


ความหมายของสัญลักษณ์ คือ การรับข้อมูลหรือคำสั่งผ่านทางคีย์บอร์ด

ตัวอย่างรูปแสดงการใช้งานสัญลักษณ์ Manual Input



8. สัญลักษณ์ Manual Operation



ความหมายของสัญลักษณ์คือการรับข้อมูลหรือแสดงผลลัพธ์โดยไม่ระบุสื่อ



ตัวอย่างรูปแสดงการใช้งานสัญลักษณ์ Manual Operation



9. สัญลักษณ์ Connector



ความหมายของสัญลักษณ์ คือ แสดงจุดเชื่อมต่อจากที่หนึ่งไปยังอีกที่หนึ่งของผังงาน โดยที่จุดเชื่อมต่ออยู่ในหน้าเดียวกัน

ตัวอย่างรูปแสดงการใช้งานสัญลักษณ์ Connector

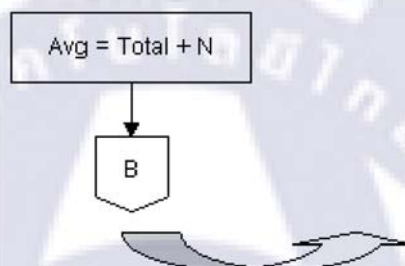


10. สัญลักษณ์ Off page Connector



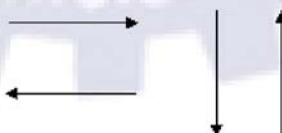
Off page Connector ความหมายของสัญลักษณ์ คือ แสดงจุดเชื่อมต่อจากที่หนึ่งไปยังอีกที่หนึ่งของผังงาน โดยที่จุดเชื่อมต่ออยู่ต่างหน้ากัน

ตัวอย่างรูปแสดงการใช้งานสัญลักษณ์ Off page Connector



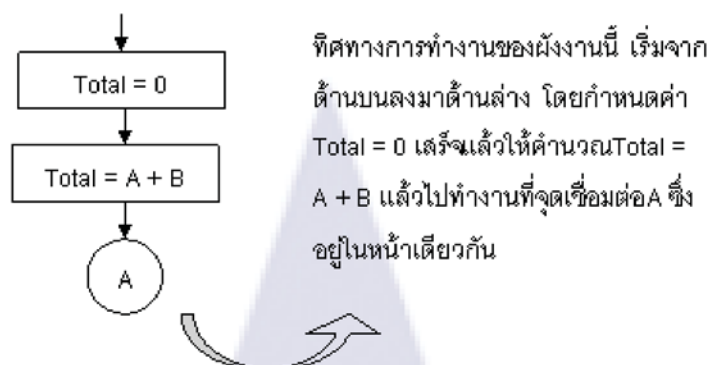
คำนวณค่า $Avg = Total + N$ เสร็จแล้ว ไปทำงานต่อไปที่จุด B โดยที่จุด B อยู่ต่างหน้ากัน

11. สัญลักษณ์ flow line and arrowheads



ความหมายของสัญลักษณ์ คือ แสดงทิศทาง การทำงานของผังงาน โดยใช้ลูกศร เป็นตัวกำหนดทิศทาง การทำงานของผังงาน

ตัวอย่างรูปแสดงการใช้งานสัญลักษณ์ flow line and arrowheads



12. สัญลักษณ์ Punched Card

Punched Card

ความหมายของสัญลักษณ์คือการรับข้อมูลหรือแสดงผลลัพธ์บนบัตรเจาะรู

ตัวอย่างรูปแสดงการใช้งานสัญลักษณ์ Punched Card



2.2.1 หลักการเขียนผังงานโปรแกรม

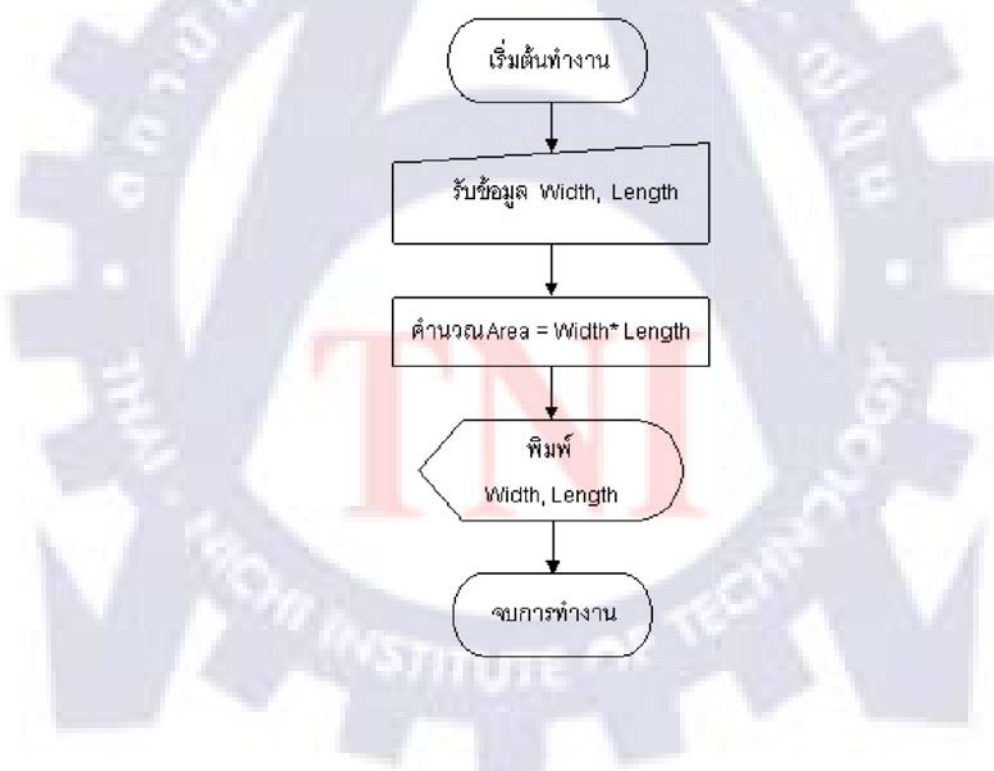
- 1) เขียนผังงานจากด้านบนลงด้านล่าง
- 2) ใช้สัญลักษณ์ให้ตรงกับ ความหมายของผังงาน
- 3) ใช้เส้น flow line และ arrowheads แสดงทิศทางการทำงานของผังงาน

- 4) ใส่คำอธิบายลงในสัญลักษณ์สั้น ๆ แต่เข้าใจง่าย จะใช้ภาษาไทยหรืออังกฤษก็ได้
- 5) หลีกเลี่ยงการโยงเส้นทิศทางของผังงานให้ตัดกันเพราะจะทำให้สับสน ควรใช้จุดเชื่อมต่อ (connector) หน้าเดียวกันแทน จะเข้าใจได้ง่ายและเป็นระเบียบ
- 6) พยายามเขียนผังงานให้จบในหน้ากระดาษเดียวกัน แต่ถ้าไม่จบในหน้าเดียวกัน ควรใช้จุดเชื่อมต่อต่างหน้ากระดาษ (off page connector)

2.3) ขั้นตอนการเขียนผังงานโปรแกรม

เพื่อความเข้าใจเกี่ยวกับขั้นตอนการเขียนผังงานโปรแกรมได้ถูกต้องให้ศึกษาจากตัวอย่างดังต่อไปนี้

การเขียนผังงาน โปรแกรม จะนำขั้นตอนวิธีการทำงานของโปรแกรม ที่ได้จากการวิเคราะห์งาน มาเขียนผังงานได้ดังนี้



ภาพที่ 2.4 แสดงผังงานขั้นตอนวิธีการทำงานของโปรแกรม

ที่มา : สมชาย รัตนเลิศสุพรรณ, 2545 : 17.

3) การเขียนโปรแกรม (programming)

ในหัวข้อเรื่องการเขียนโปรแกรมจะกล่าวถึงเนื้อหา 2 ส่วน คือความหมายของโปรแกรมคอมพิวเตอร์ และภาษาคอมพิวเตอร์ดังมีรายละเอียดต่อไปนี้

3.1) ความหมายของโปรแกรมคอมพิวเตอร์

โปรแกรมคอมพิวเตอร์ (computer program) หมายถึง ชุดของคำสั่งที่เขียนขึ้นเพื่อสั่งให้คอมพิวเตอร์ทำงานตามที่ต้องการ ส่วนการเขียนโปรแกรม (programming) หมายถึง การเขียนชุดคำสั่งด้วยภาษาคอมพิวเตอร์ภาษาใดภาษาหนึ่ง เพื่อให้คอมพิวเตอร์ทำงานตามโปรแกรมที่เราเขียนขึ้น

ดังนั้น ผู้เขียนโปรแกรม (programmer) จึงจำเป็นต้องเรียนรู้ภาษาคอมพิวเตอร์ภาษาใดภาษาหนึ่งหรือหลายภาษา แล้วฝึกฝนทักษะการเขียนโปรแกรมด้วยภาษาคอมพิวเตอร์ที่ต้องการใช้งานให้เกิดความชำนาญจึงจะสามารถเขียนโปรแกรมคอมพิวเตอร์ได้อย่างมีประสิทธิภาพ

4) การทดสอบและแก้ไขโปรแกรม (testing and editing program)

หลังจากที่เราเขียนโปรแกรมเสร็จแล้ว ขั้นตอนต่อไป คือ การทดสอบโปรแกรมที่เขียนขึ้นซึ่งอาจพบความผิดพลาดได้ 2 ชนิด ดังนี้

1) ความผิดพลาดทางไวยากรณ์ภาษา (syntax error) เป็นความผิดพลาดที่เกิดจากการเขียนคำสั่งของภาษาคอมพิวเตอร์ผิด ตัวอย่างเช่น คำสั่ง `printf()` ในภาษา C ต้องเขียนด้วยตัวอักษรตัวเล็ก แต่เขียนเป็น `PRINTF()` เป็นต้น โดยส่วนมากความผิดพลาดทางไวยากรณ์ จะถูกตรวจสอบพบเมื่อมีการแปลโปรแกรม (compile) ให้เป็นภาษาเครื่อง ซึ่งเราสามารถแก้ไขโดยการเขียนคำสั่งให้ถูกต้องตามไวยากรณ์ของภาษานั้น ๆ

2) ความผิดพลาดทางตรรกะ (logical error) เป็นความผิดพลาดที่เกิดจากการลำดับการทำงานผิดหรือป้อนสูตรคำนวณผิด ตัวอย่างเช่น ต้องการหาค่า $X = X + Y$ แต่ป้อนสูตรเป็น $X = X * Y$ อย่างนี้ เป็นต้น วิธีการตรวจหาความผิดพลาดแบบนี้ คือ ตรวจสอบการคำนวณผลลัพธ์ของโปรแกรมว่าตรงกับผลลัพธ์ที่คำนวณด้วยมือหรือเครื่องคิดเลขหรือไม่ ถ้าไม่ตรงกันแสดงว่าเกิดความผิดพลาดทางตรรกะขึ้น วิธีการแก้ไขก็คือ การแก้ไขสูตรให้ถูกต้อง หรือแก้ไขลำดับการทำงานให้ถูกต้อง

ดังนั้น จะเห็นได้ว่าผู้เขียนโปรแกรมจะต้องทำการทดสอบโปรแกรมและแก้ไขโปรแกรมให้สามารถทำงานได้ตามที่ต้องการ

5)การจัดทำเอกสารประกอบและการบำรุงรักษาโปรแกรม

เมื่อผู้เขียนโปรแกรมได้ทำการทดสอบโปรแกรมและแก้ไขความผิดพลาดที่เกิดขึ้นเรียบร้อยแล้ว ขั้นตอนต่อไปก็ควรจัดทำคู่มือการใช้โปรแกรม เพราะจะทำให้ผู้ใช้โปรแกรมสามารถใช้งานได้อย่างถูกต้อง โดยคู่มือการใช้โปรแกรมที่ดีควรจัดทำในลักษณะที่แสดงการทำงานเป็นขั้นตอน ผู้ใช้โปรแกรมสามารถปฏิบัติตามได้จริง ส่วนการบำรุงรักษาโปรแกรมที่พัฒนาขึ้นก็ต้องมีคำแนะนำให้ผู้ใช้โปรแกรมปฏิบัติ ตัวอย่างเช่น การเก็บโปรแกรมต้นฉบับ ควรเก็บไว้ในกล่องที่มิดชิด ป้องกันฝุ่นได้ ไม่ควรเก็บไว้ในโต๊ะทำงาน ไม่ควรเก็บไว้ในรถยนต์เพราะอาจถูกอากาศร้อนทำให้เสียได้ นอกจากนี้ต้องมีการสำเนาโปรแกรมต้นฉบับเอาไว้ อย่างน้อย 1 ชุด แล้วนำชุดที่สำเนาไปใช้ ไม่ควรใช้โปรแกรมต้นฉบับโดยตรง ควรเก็บเอาไว้สำหรับกรณีที่โปรแกรมสำเนาเกิดปัญหาจะได้นำโปรแกรมต้นฉบับมาทำสำเนาและใช้งานได้ทันที

2.1.3 Adobe Flash (CS3)

2.1.3.1 ความหมายของ Flash

Adobe Flash ซึ่งยังหมายถึง Macromedia Flash Player และโปรแกรมอื่น ๆ ที่ใช้ในการเขียนสื่อมัลติมีเดียที่เอาไว้ใช้สร้างเนื้อหาเกี่ยวกับ Flash ซึ่งตัว Flash Player พัฒนาและเผยแพร่โดย Adobe System (เริ่มต้นพัฒนาโดยบริษัท Future Ware ตอนหลังเปลี่ยนเป็น Generator ซึ่งภายหลังถูกควมรวมกิจการเข้ากับ Adobe) ซึ่งเป็นโปรแกรมที่ทำให้ Web Browser สามารถแสดงตัวมันได้ ซึ่งมันมีความสามารถในการรองรับ ภาพแบบเวกเตอร์ และ ภาพแบบแรสเตอร์ และมีภาษาสคริปต์ที่เอาไว้ใช้เขียนโดยเฉพาะเรียกว่า Action Script และยังสามารถเล่นเสียงและวิดีโอ แบบสตรีมได้ แต่ในความหมายจริงๆ แล้ว Flash คือ โปรแกรมแบบ Integrated Development Environment (IDE) และ Flash Player คือ Virtual Machine ที่ใช้ในการทำงานงานของไฟล์ Flash ซึ่งในภาษาพูดเราจะเรียกทั้งสองคำนี้ในความหมายเดียวกัน Flash ยังสามารถความความถึงโปรแกรมเครื่องมือต่างๆ ตัวแสดงไฟล์หรือ ไฟล์โปรแกรม

เทคโนโลยี Flash ได้กลายมาเป็นที่นิยมในการเสนอ Animation และ Interactive ใน Webpage และในโปรแกรมหลายๆ โปรแกรมระบบ และเครื่องมือต่าง ๆ ที่มีความสามารถในการแสดง Flash ได้ และ Flash ยังเป็นที่นิยมในการใช้สร้าง Animation โฆษณาออกแบบส่วนต่าง ๆ

ของ Webpage ไล่วิดีโอบนเว็บ และอื่น ๆ อีกมากมายไฟล์ Flash ในบางครั้งอาจเรียกว่า Flash Movies โดยทั่วไปกับไฟล์ที่มีนามสกุล .swf และ .flv

2.1.3.2 รูปแบบไฟล์และนามสกุลไฟล์ที่เกี่ยวข้อง

รูปแบบไฟล์ Flash

- .swf ไฟล์ .swf เป็นไฟล์ที่สมบูรณ์ ถูก Compiled และ published ไฟล์แล้ว ซึ่งไม่สามารถแก้ไขด้วย Macromedia Flash ได้อีกต่อไป อย่างไรก็ตาม ยังมีโปรแกรม .swf decompile อยู่ด้วย
- .fla ไฟล์ .fla เป็นไฟล์ต้นฉบับของโปรแกรม Flash โปรแกรมที่ใช้เขียน Flash สามารถแก้ไขไฟล์ fla และ compile มันให้เป็นไฟล์ .swf ได้ อย่างไรก็ตาม รูปแบบไฟล์ fla ยังคงไม่กำหนดเป็นแบบเปิด
- .flv ไฟล์ .flv เป็นไฟล์ VDO Flash ซึ่งสร้างโดย Macromedia Flash, Sorenson Squeeze, หรือ On2 Flix.
- .avi ไฟล์ avi เป็นไฟล์ VDO เป็นคำย่อของ Audio Video Interleave. ซึ่ง Flash สามารถสร้างไฟล์ในรูปแบบนี้ได้.
- .spa ไฟล์ .spa คือไฟล์เอกสารของ Future Splash.
- .xml ไฟล์ .xml คือไฟล์ Configuration ของ flash ซึ่งใช้เก็บข้อมูลที่ไม่ต้องการคอมไพล์ใหม่ เช่น link เป็นต้น

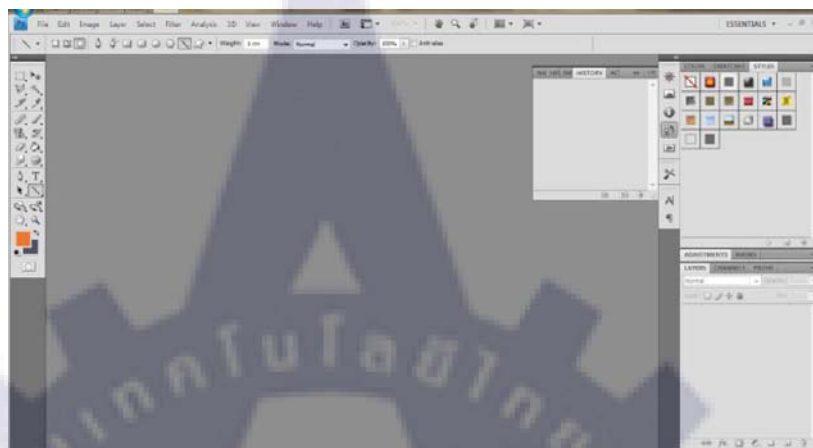
2.1.3.3 ไฟล์ Flash สามารถใช้ได้

- .avi ไฟล์ avi เป็นไฟล์วิดีโอ เป็นคำย่อของ Audio Video Interleave.
- .gif ภาพเคลื่อนไหว
- .png ไฟล์ png คือ ไฟล์ซึ่งสามารถมาแก้ไขได้ภายหลัง (ซึ่งยังมีการแบ่ง Layer ไว้) หลังจากที่ถูกบันทึกไว้
- .ssk ไฟล์ .ssk คือไฟล์ Smart Sketch.
- .piv ไฟล์ .piv คือไฟล์ Animation Pivot StickFigure.

2.1.4 Adobe Photoshop (CS3)

Photoshop คือโปรแกรมที่ใช้สำหรับการจัดการเกี่ยวกับภาพหรืองานกราฟิกที่ต้องการความละเอียดสูง Photoshop เป็นโปรแกรมกราฟิกที่จัดระบบความละเอียดของงานเป็นแบบ Raster Graphic ก็คือภาพที่อยู่ในโปรแกรมนั้นจะแบ่งความละเอียดของภาพออกเป็นตารางสี่เหลี่ยมลายหมากรุกซึ่งในแต่ละช่องหรือแต่ละตารางนั้นจะเรียกว่า Pixel ซึ่งความละเอียดของงาน

จะสัมพันธ์ในเชิงผกผันกับขนาดของการแสดงผลทางหน้าจอ ความสามารถของโปรแกรมนี้ คือ โปรแกรมนี้ สามารถทำภาพ วน, เว้า, ย่อหรือขยายขนาดของภาพ, ใส่เงาของภาพ, ทำตัวอักษรวน, รวมภาพเข้าไว้ด้วยกัน, และอื่นๆอีกมากมาย มาย และโปรแกรมนี้ก็สามารถมี Plug-IN ที่สามารถไปหา Download มาเสริมโปรแกรมของเราได้ เพื่อเพิ่มความสามารถของโปรแกรมได้อีก



ภาพที่ 2.5 หน้า Interface โปรแกรม Adobe Photoshop CS3

2.1.5 Microsoft Office



ภาพที่ 2.6 Microsoft Office

2.1.5.1 Microsoft Word

เป็นโปรแกรมประมวลผลคำ (Word Processing Application) ที่สามารถทำงานได้ทั้งภาษาไทย ภาษาอังกฤษ และภาษาอื่นๆ โปรแกรมมีความสามารถในการใช้งานด้านเอกสาร เช่น จดหมาย รายงาน บันทึกข้อความ ใบประกาศ และคู่มือการใช้งานต่างๆ การจัดการกับข้อความใน

เอกสารได้อย่างง่าย เช่น การแก้ไขข้อความ ตกแต่งด้วยสี และแบบตัวอักษร การใส่ตาราง รูปภาพ การตรวจสอบตัวสะกดและไวยากรณ์ การทำจดหมายเวียน และการใช้งานเอกสารร่วมกัน นอกจากนี้ยังสามารถบันทึกเอกสารในรูปแบบของ XML เพื่อให้ผู้ใช้สามารถแยกเนื้อหาของเอกสารออกจากขอบเขตของรูปแบบของเอกสาร

2.1.5.2 Microsoft Excel

Microsoft Excel เป็นโปรแกรมประเภท spreadsheet หรือโปรแกรมตารางงานซึ่งใช้เก็บข้อมูลต่าง ๆ สูตรคำนวณ ลงบนแผ่นตารางงานคล้ายกับการเขียนข้อมูลลงไปในสมุดที่มีการตีช่องตารางทั้งแนวนอนและแนวตั้ง ตารางแต่ละช่องจะมีชื่อกำกับไว้ในแนวตั้งของตารางเป็นตัวอักษรภาษาอังกฤษเริ่มจาก A,B,C,...เรื่อยไปจนสุดขอบตารางทางขวา มีทั้งหมด 256 สดมภ์(Column) แนวนอนมีหมายเลขกำกับเป็นบรรทัดที่ 1,2,3,...เรื่อยไปจนถึงบรรทัดสุดท้ายจำนวนบรรทัดจะต่างกันในแต่ละโปรแกรมในที่นี้เท่ากับ 65,536 แถว(Row) ช่องที่แนวตั้งและแนวนอนตัดกันเรียกว่า เซลล์ (Cell) ใช้บรรจุข้อมูล ข้อความ หรือสูตรคำนวณ โปรแกรมประเภท spreadsheet มีผู้พัฒนาขึ้นมาหลายโปรแกรม เช่น ซุปเปอร์แคล (SuperCalc), วิสิแคล (VisiCalc), โลตัส 1-2-3 (Lotus 1-2-3), ออราเคิล (Oracle) และไมโครซอฟท์เอ็กเซล(Microsoft Excel) เป็นต้น

2.1.6 Action Script

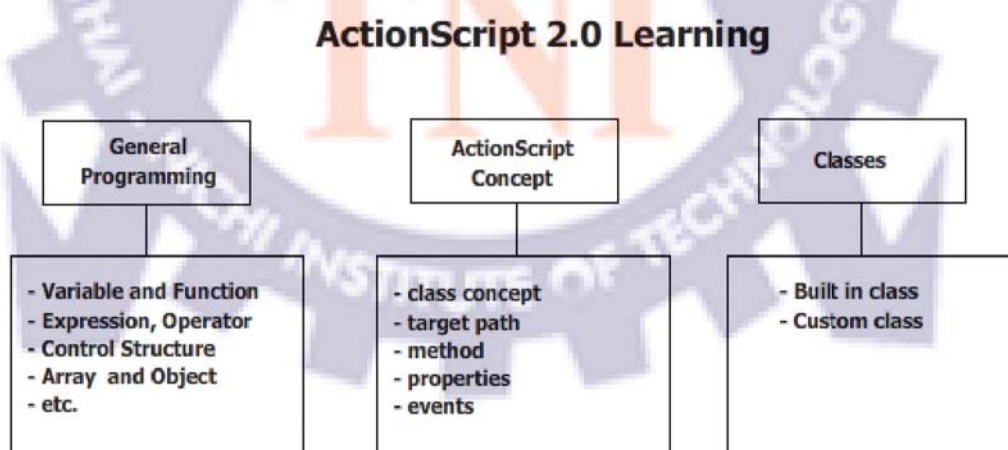
ภาษา Action Script (เรียกย่อๆว่า AS) นั้นเป็นภาษาที่ใช้งานกับไฟล์ flash เป็นหลัก แต่สามารถเขียนพัฒนาได้จากหลายโปรแกรมอื่นๆในปัจจุบัน เช่น AIR (เน้นพัฒนา application บน desktop), flex builder (flash builder 4 ในปัจจุบัน) หรือโปรแกรมโอเพ่นซอร์ส (Open Source หมายถึงโปรแกรมที่ไม่มีการปิดบังโค้ด เพื่อให้คนอื่นสามารถร่วมพัฒนาต่อยอดได้ และที่สำคัญคือ ฟรี) อย่าง flashdeveloper หรืออื่นๆได้เช่นกัน โดยประโยชน์ของ AS นั้น คือการช่วยให้ผู้ชม flash สามารถทำการอินเตอร์แอคทีฟหรือตอบโต้กับวัตถุต่างๆได้ นอกจากนี้ เมื่อ AS ก้าวกระโดดพัฒนาไปมากขึ้น การเชื่อมต่อกับแหล่งข้อมูลอื่นๆ ก็ทำได้มากขึ้น รวมถึงบางสิ่งที่ flash ต้องอาศัยภาษาอื่นช่วยในการทำ ก็สามารถทำได้ด้วยตัวมันเองในภายหลัง ซึ่งเป็นหน้าที่ของเหล่า developerทั้งหลายที่จะต้องคอยติดตามการพัฒนาที่มากมายเหล่านี้ อยู่สม่ำเสมอ

สำหรับ flash ในเวอร์ชันยุคแรกเริ่มนั้น เวอร์ชันที่ใช้คือ 1.0 ซึ่งการทำงานนั้นก็ค่อนข้างจะครบถ้วนสมบูรณ์ระดับหนึ่ง ในสมัย flash 5 (เวอร์ชัน 4 บางอย่างยังคงต้องอาศัยไทม์ไลน์เป็นตัวช่วย) และทำงานได้มากขึ้นใน เวอร์ชัน 6 หรือ MX

แต่ทั้งนี้กระแสการ เขียนโปรแกรมเชิงวัตถุ (Object Oriented Programming, เป็นการเขียนโปรแกรมโดยมองสิ่งต่างๆเป็นวัตถุที่มี คุณสมบัติ, การดำเนินการต่างๆ ฯลฯ) นั้นเป็นสิ่งที่ขาดเสียไม่ได้เลยที่จะมองข้าม ถึงแม้ว่า AS 1.0 นั้นจะเขียน OOP ได้ก็ตาม แต่ไม่ใช่รูปแบบการเขียน OOP ที่เป็นมาตรฐานทั่วไปสักเท่าไรนัก (AS 1.0 นั้นใช้คำสั่ง prototype ที่แปลว่าต้นแบบ ในการสร้างคลาส (Class) ดังนั้น AS 2.0 จึงได้ถูกนำมาใช้ โดยเราสามารถเขียน Action Script เป็นไฟล์สำหรับคลาส แยกออกมาต่างหาก โดยมีการจัดวางรูปแบบการเปิดและปิด การประกาศตัวแปรต่างๆ ให้ตรงตามลักษณะของ OOP มากขึ้น

และ เมื่อ flash cs3 ภายใต้แบรนด์ของ Adobe ถูกปล่อยออกมา ความเป็น OOP ที่สมบูรณ์มากขึ้นของ AS ก็ถูกปล่อยออกมาในนาม AS 3.0 ที่ค่อนข้างจะเปลี่ยนรูปแบบการเขียนแบบเดิมๆ ของ AS ไปเลย และ การอ้างอิงวัตถุ ตัวแปร หรือฟังก์ชันต่างๆ ก็ค่อนข้างจะต้องรัดกุมมากขึ้น เพราะมีข้อบังคับ ของประเภทข้อมูลอย่างเช่นตัวแปรและอื่นๆ มากขึ้น นอกจากนี้การประกาศคำสั่งอีเวนต์ เพื่อตรวจสอบว่าวัตถุนั้นๆควรจะทำงานเมื่อใดก็เปลี่ยนไป โดยไม่สามารถใช้รูปแบบการเขียนแบบเก่ามาผสมได้

ถึง แม้ AS 3.0 จะเปลี่ยนแปลงรูปแบบการเขียนไปหลายอย่างก็ตาม แต่พื้นฐานความเข้าใจเบื้องต้นของ AS2.0 ก็ถือว่ายังใช้ได้อยู่ นอกจากนี้ ไม่ใช่ทุกคนซะทีเดียวที่จะพัฒนางานด้วย AS 3.0 ฉะนั้นในเบื้องต้น เราจะทำความเข้าใจ AS 2.0 ให้เข้าใจอย่างถ่องแท้ซะก่อนๆ จะก้าวไปสู่ระดับที่สูงขึ้น (แต่ developer ที่มีพื้นฐานการเขียนโปรแกรมแบบ OOP อยู่แล้วบางคนนั้นเริ่มต้นกับ AS 3.0 เลย ก็ถือว่าไม่ได้ผิดอะไร แต่อาจจะงัดข้ออ้างงานที่เขียนด้วย AS 2.0)



ภาพที่ 2.7 หัวข้อการเรียนรู้ ActionScript

การเรียนรู้คำสั่งพื้นฐานเชิงโปรแกรมมิ่งโดยทั่วไป ซึ่งประกอบด้วย

- 1.1 ตัวแปร (Variable) และ Function (Function)
- 1.2 Expression การใช้ Operator ต่างๆ
- 1.3 Control Structure
- 1.4 Array และเปรียบเทียบการใช้งานกับ Object
- 1.5 อื่นๆ

ซึ่งความรู้เหล่านี้ ถือว่าเป็นเรื่องทั่วไปที่ไม่่ว่าภาษาไหนก็จะไม่ค่อยแตกต่างกันมากนักซึ่ง คนที่มีพื้นฐานการเขียนโปรแกรมอยู่แล้วสามารถอ่านและทำความเข้าใจได้ไม่ยาก ไม่จำเป็นต้อง เสียเวลามาก เรียนรู้เพียงบางจุดคำสั่งที่มีเฉพาะใน ActionScript หรือจะอ่านเพื่อให้เข้าใจมากยิ่งขึ้น ก็ได้

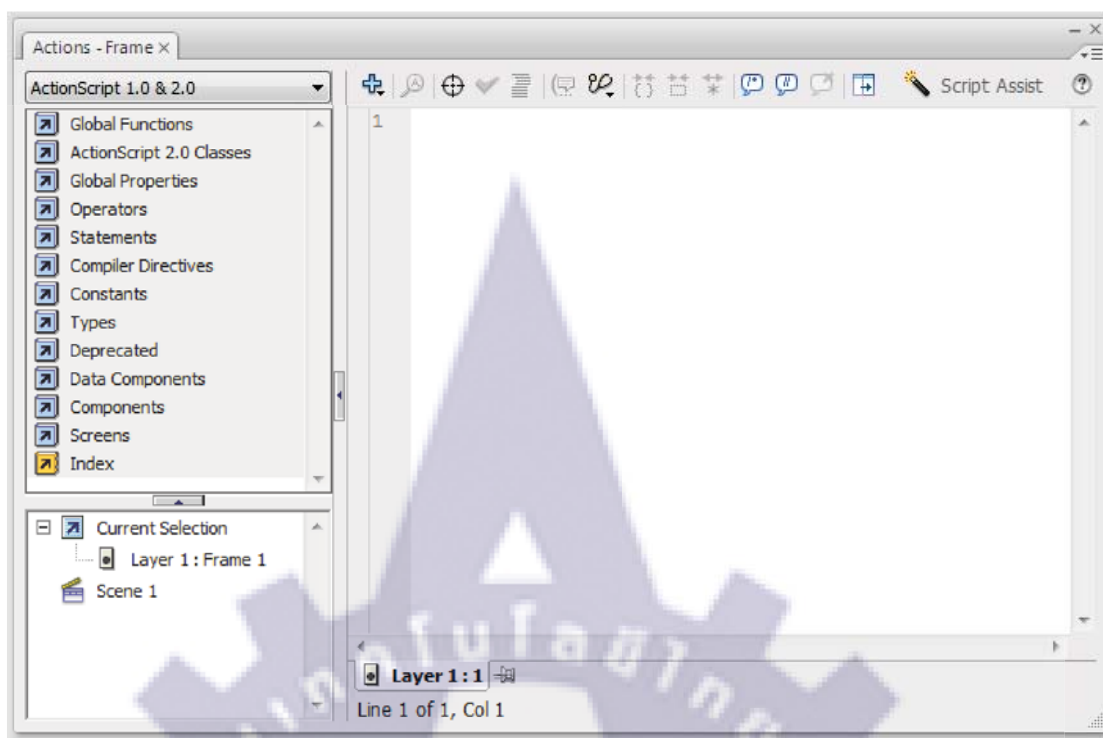
2.1.6.1 ส่วนประกอบหลักของ flash ได้แก่

- 2.1.6.1.1 target path
- 2.1.6.1.2 method
- 2.1.6.1.3 properties
- 2.1.6.1.4 events

ซึ่งจะทำการเข้าใจคอนเซปต์เบื้องต้นในเชิงวัตถุก่อน และทำความรู้จักว่าทั้ง 4 ส่วนนี้มี หลักการอย่างไรบ้าง ซึ่งถือว่าเป็นหัวใจสำคัญในการเขียน ActionScript เบื้องต้น

2.1.6.2 อัลกอริทึม (Algorithm)

อัลกอริทึม จะหมายถึงลำดับขั้นตอนการทำงาน โดยอัลกอริทึมนั้นเป็นเรื่องของ ประสบการณ์ของแต่ละคน เนื่องจากการเขียนโปรแกรมไม่มีหลักการตายตัว บางคนอาจเขียนให้ โปรแกรมทำงานได้ตามที่ต้องการ แต่โค้ดยาว ลำดับขั้นตอนเยอะ ซับซ้อน แต่บางคนเขียนโค้ดได้ ผลลัพธ์เดียวกัน แต่โค้ดสั้นกระชับ ลำดับขั้นตอนการทำงานน้อยมาก ซึ่งเรื่องเหล่านี้ ก็อยู่บน พื้นฐานความเข้าใจในการใช้งานคำสั่ง และพื้นฐานการจัดวางโครงสร้างของโปรแกรมเช่นเดียวกัน นอกจากนี้ ยังมีเรื่องอื่นๆให้ศึกษาอีก อาทิเช่น Frame Work ซึ่งเกี่ยวกับรูปแบบที่จัดเอาไว้สำหรับการ เขียนโปรแกรมรวมไปถึงคำสั่งต่างๆที่จัดเตรียมมาให้ใช้งานได้สะดวกมากขึ้น และ Design Pattern ซึ่งเกี่ยวข้องกับการจัดวางโครงสร้างของโปรแกรมให้ได้มาตรฐาน



ภาพที่ 2.8 หน้าต่าง ActionScript

2.1.6.3 วิธีการเขียน ActionScript

การเขียน ActionScript นั้นจะแบ่งออกเป็น 3 รูปแบบคือ

1. การเขียนแบบ Script Assist ซึ่งจะเป็นการกรอกข้อมูลลงในช่องแบบฟอร์มที่กำหนดมาให้ (เคยถูกตัดไปในเวอร์ชัน MX และนำกลับมาใหม่)
2. การเขียน ActionScript แบบ advance หรือแบบที่เขียนโค้ดเองทั้งหมด
3. การเขียน ActionScript โดยใช้ พาเนล Behavior ซึ่งจะเป็นการกำหนดโค้ดให้กับวัตถุ โดยใช้วิธีการเลือกคำสั่งต่างๆ แบบรวดเร็ว และจะปรากฏคำสั่งใน Actions Panel

เราสามารถเขียน ActionScript ได้ทั้งหมด 3 ที่หลักๆคือ

1. เขียนไว้บน Frame โดยทำการคลิกที่เฟรมใดๆ แล้วก็เขียนโค้ดลงไป
2. เขียนไว้บนตัว instance (Movie clip หรือ Button) โดยคลิกที่ตัว instance ที่ต้องการแล้วใส่โค้ดลงไป โดยมีเงื่อนไขพิเศษคือ ต้องทำการกำหนด event ก่อนเสมอ มิฉะนั้นโปรแกรมจะแจ้งเตือน error (เราจะได้เรียนรู้เพิ่มเติมในหัวข้อเกี่ยวกับ event)

3. เขียนโค้ดไว้ในไฟล์นามสกุล as ซึ่งวิธีการนำมาใช้นั้นมีหลายรูปแบบ ทั้งการใช้คำสั่ง include เพื่อดึงเข้ามา หรือถ้าเขียนไว้ในรูปแบบของ คลาส จะใช้คำสั่ง import นอกจากนี้ยังสามารถนำไปกำหนดให้กับ document (AS3) หรือ มูฟวี่คลิปในไลบรารีได้ด้วย โดยจะมีช่องชื่อว่า class เพื่อให้กำหนด as ไฟล์ลงไป

การแสดงผลว่า "hello world" ฉะนั้นเราควรจะรู้ว่า flash แสดงคำสั่งได้อย่างไร ซึ่งจะยกตัวอย่างการเขียนลงบน timeline โดยมีขั้นตอนดังนี้

1. คลิกที่เฟรมที่ 1 ของ fla ที่ถูกเปิดขึ้นมาใหม่
2. กด <F9> เพื่อเปิด Actions panel
3. พิมพ์คำสั่งลงไปว่า

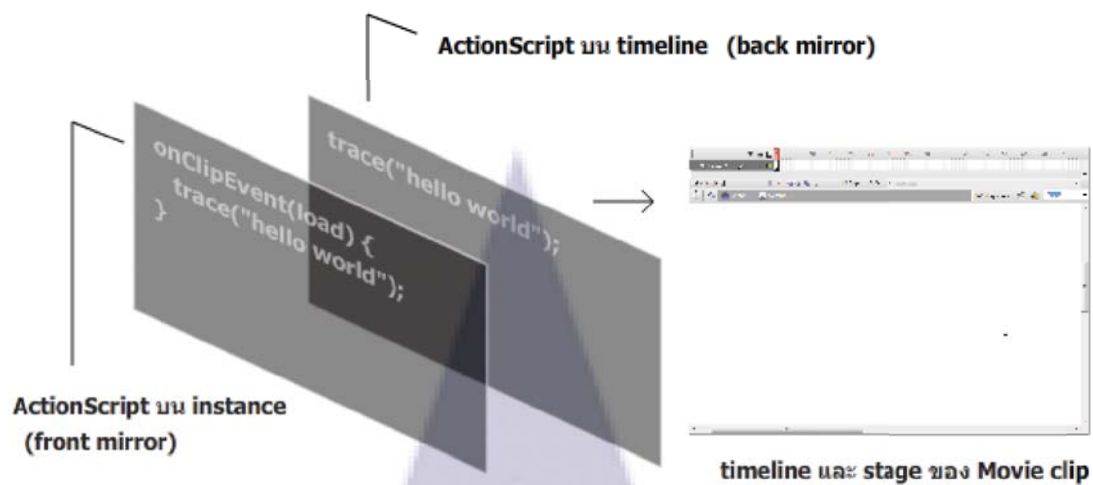
Code:

```
trace("hello world");
```

4. กดทดสอบว่า <Ctrl+Enter> ซึ่งคำสั่งดังกล่าวจะถูกแสดงผลออกมาให้เห็นทางหน้าต่าง Output ว่า hello world

สำหรับ flash นั้นการเขียนคำสั่งสามารถทดสอบผลลัพธ์และแสดงผลด้วยคำสั่ง trace ได้ ทางหน้าต่าง output ภายในโปรแกรม flash ได้เลย และกรณีที่มีคำสั่งส่วนใดเกิดข้อผิดพลาด ก็จะสามารถผ่านหน้าต่าง Compiler Errors ซึ่งกรณีที่เป็นภาษาทาง server side script อย่างเช่น php จะใช้คำสั่งว่า echo ซึ่ง echo ในที่นี้ใช้ทั้งตรวจสอบค่า และแสดงผลพร้อมทั้งออกทางบราวเซอร์ได้ เช่นเดียวกัน ข้อแตกต่างก็คือ คำสั่ง trace นี้จะไม่ถูกแสดงใน swf หรือเมื่อทดสอบผ่านทางบราวเซอร์นั่นเอง (แต่ก็มี plugin สำหรับ firefox เพื่อตรวจสอบค่าที่ trace ออกมาได้)

ซึ่งการเขียนโปรแกรมที่ดี เราควรตรวจสอบได้ว่า คำสั่งตรงไหนที่น่าจะผิดพลาด ตรงไหนทำ ตรงไหนไม่ทำ ตรงไหนมีค่า ตรงไหนไม่มีค่า ฉะนั้นมันใช้ trace เพื่อตรวจสอบให้เป็นนิสัย ซึ่งจะช่วยให้เราพบปัญหาได้ง่ายขึ้นมากกว่าไล่โค้ดดูนานๆแล้วหาข้อผิดพลาดไม่เจอ



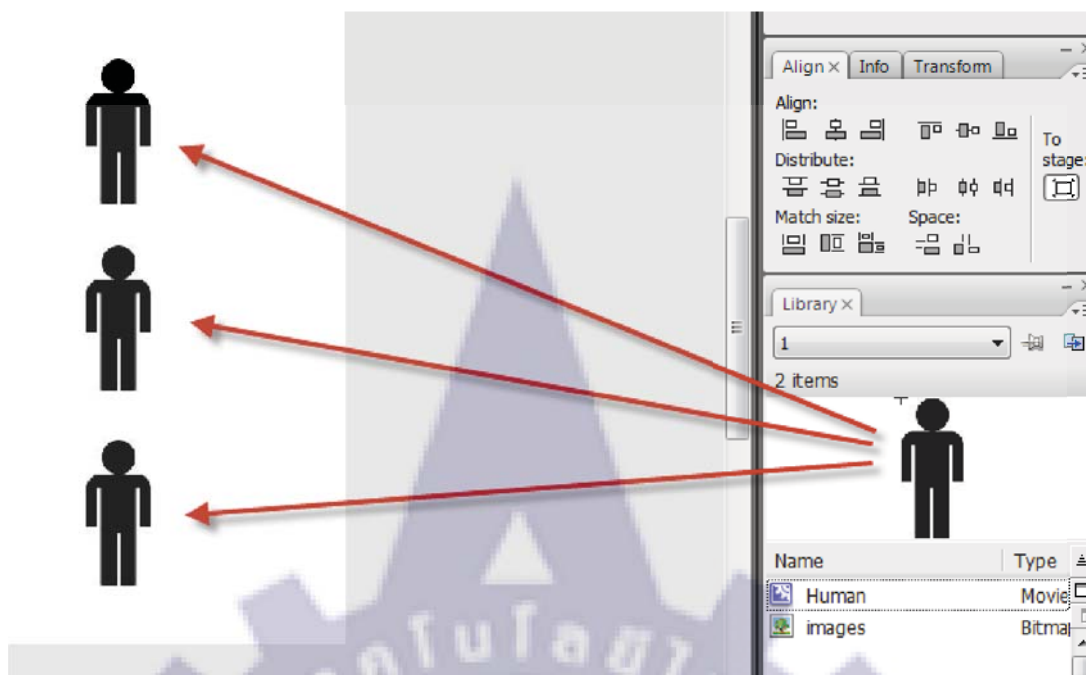
ภาพที่ 2.9 การทำงานของ ActionScript

จากรูปจะแยกเป็นกระจกแผ่นหน้าและแผ่นหลัง เพื่อให้เห็นความแตกต่างระหว่างการเขียนโค้ดทั้งสองที่ แต่มันที่จริงแล้วก็เป็นกระจกแผ่นเดียวกัน ซึ่ง class นั้นเปรียบเสมือนต้นแบบ ของวัตถุ ก็ว่าได้

2.1.6.4 Class ที่เป็น รูปธรรม และ นามธรรม

สำหรับ class นั้น เราไม่ได้จำกัดมันกับแค่สิ่งมีชีวิต จะเป็นสิ่งไม่มีชีวิตก็ได้ ซึ่งคลาสนั้น เมื่อถูกสร้างออกไปใหม่ เราจะเรียกว่า วัตถุ (Object) หรือ ตัวแทน (Instance) และจะสามารถทำงานได้เหมือนกับ class ทุกอย่าง

กรณีใน flash นั้น เมื่อเราสร้าง symbol ขึ้นมา มันจะถูกเก็บอยู่ในไลบรารี และเมื่อเราทำการลากจากไลบรารีมาวางบน stage ก็ถือว่ามันได้ถูกสร้างขึ้นใหม่โดยลอกเลียนแบบทุกอย่างจากต้นแบบออกมาเรียบร้อยแล้ว และวัตถุที่สร้างใหม่นี้ เราเรียกว่า instance แต่จะมองว่ามันคือ object ก็ได้เช่นเดียวกันเพราะอันที่จริงมันคือสิ่งเดียวกัน ซึ่งเราอาจจะมองว่า symbol ก็เปรียบเสมือน คลาส ที่เป็นรูปธรรม คุณสามารถเห็นมันได้ เพราะมันมีรูปร่าง (shape)



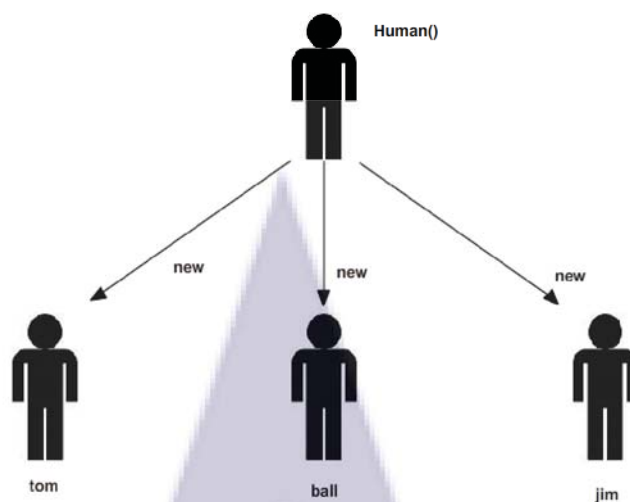
ภาพที่ 2.10 การทำงานของ Class

ใน ActionScript 3.0 นั้นเราสามารถกำหนด symbol ให้เป็นคลาสทางไลบรารี แล้วสร้าง instance ด้วย ActionScript ได้เลย

แต่ในการเขียน ActionScript หรือภาษาอื่น ๆ นั้น คุณจะสร้าง class ที่ออกจะเป็น นามธรรม เพราะมันไม่มีรูปร่างให้คุณเห็น แต่มันก็ยังสามารถทำงานได้อย่างเช่น ต้องการสร้าง มนุษย์ (Human) ขึ้นมา ก็จะได้ คลาส Human ซึ่งเป็นต้นแบบหลักๆของมนุษย์ทั่วโลก และผมสามารถสร้าง มนุษย์ออกมาหลายๆคน ด้วยกระบวนการที่ชื่อว่า Construct โดยรูปแบบการเขียน ActionScript จะเป็นดังนี้

Code:

```
tom = new Human();
ball = new Human();
jim = new Human();
```



ภาพที่ 2.11 การเก็บข้อมูลของ Class

2.1.6.5 การกำหนดชื่อให้กับ instance หรือ object

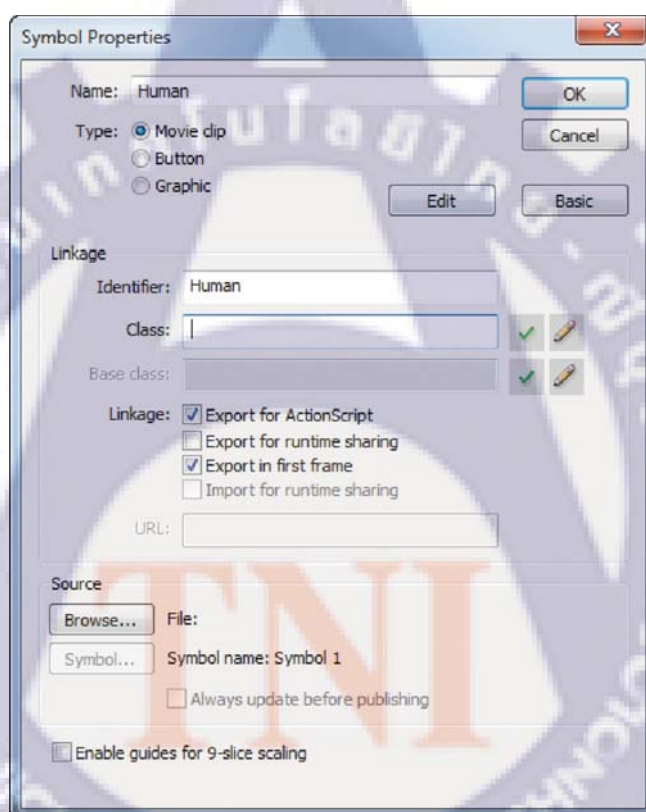
สำหรับ instance บน stage อย่าง Movie clip และ Button เราสามารถกำหนดชื่อให้กับมัน เพราะถ้าไม่มีชื่อ เราก็ไม่สามารถอ้างอิงมันได้ ก็เหมือนมนุษย์ที่เกิดมา ถ้าไม่ตั้งชื่อ เราก็ไม่สามารถที่จะอ้างอิงตัวเขาได้ถูก โดยการตั้งชื่อสำหรับ Movie clip และ Button สามารถตั้งชื่อได้โดยการคลิกเลือก instance ตัวนั้นๆ แล้วกำหนดชื่อลงไปในช่วง Instance Name ที่พาเนล Properties



ภาพที่ 2.12 การกำหนดชื่อให้กับ instance หรือ object

สำหรับกรณีการสร้าง object ด้วยการ construct นั้น ชื่อ object ดังกล่าวสามารถใช้อ้างอิงได้เลยทันที จากตัวอย่างก่อนหน้านี้ก็คือชื่อ jim, ball, tom นั่นเอง

นอกจากนี้ Human ถือว่าเป็นชื่อคลาสที่สามารถอ้างอิงได้ แต่สำหรับกรณี symbol นั้น ชื่อ symbol ถือว่าเป็นชื่อเพื่อให้ทราบว่า instance ตัวนั้นถูกสร้างจาก symbol ตัวใด แต่ยังไม่ใช่ชื่อที่ใช้ในการอ้างอิงสำหรับ ActionScript ได้จริงๆ ซึ่งวิธีการตั้งชื่อให้สำหรับ symbol นั้น เราจะกำหนดผ่านทาง library โดยตั้งค่าผ่านช่อง linkage โดยคลิกเมาส์ขวาที่ symbol เลือก Properties แล้ว กดปุ่ม Advance หรือเลือกเมนู linkage โดยตรง จะปรากฏส่วนกรอกชื่อ Identifier โดยให้ติ๊กเลือก Export for ActionScript ซึ่งโปรแกรมจะกรอกชื่อเดียวกับ symbol มาให้อัตโนมัติในช่อง Identifier แต่เราสามารถแก้ไขได้



ภาพที่ 2.13 การแก้ไขชื่อ Class

instance หรือชื่อของ Identifier ไม่สามารถตั้งซ้ำกันได้ เพราะจะทำให้โปรแกรมไม่สามารถอ้างอิงได้ เนื่องจากไม่รู้จะอ้างอิงตัวไหน เหมือนเวลาเราเรียกชื่อคนสองคนที่ชื่อเดียวกันพร้อมๆกัน เขาจะหันมางว่า เรียกใครกันแน่

กรณีที่เป็น ActionScript 3.0 Identifier นั้นจะถูกตัดออกไป แต่จะให้ทำการตั้งชื่อที่ช่อง Class เพื่อให้ symbol นั้นสามารถทำการ construct จาก ActionScript ได้เลย ซึ่งอันที่จริงแล้วช่อง Class นี้จะใช้เป็นชื่อเพื่ออ้างอิง หรือนำ Class File ที่ถูกเขียนในรูปแบบไฟล์ as มาผูกเอาไว้ก็ได้ เพื่อให้ symbol นั้นได้รับคุณสมบัติจาก Class ที่ระบุลงไป (as file) ซึ่งวิธีการระบุ Class File นี้สามารถทำได้ใน ActionScript 2.0 เช่นเดียวกัน แต่ไม่สามารถทำได้ถึงการ construct ด้วย ActionScript ยังคงต้องใช้ Identifier เป็นชื่ออ้างอิง

2.1.6.1.1 Target Path

target path นั้นถ้าแปลตรงๆก็คือ เส้นทางสู่เป้าหมาย ให้ออกนึกถึงเวลาเรานั่งรถไฟ เราจำเป็นต้องนั่งไปที่สถานีตามลำดับ ไม่สามารถลัดสถานีได้ แต่เส้นทางที่รถไฟวิ่งไปนั้น อาจจะ มีเส้นทางแยกที่จะวิ่งไปสถานีอื่นๆที่แตกต่างกันได้

ยกตัวอย่างเช่นรถไฟฟ้า BTS ถ้าเริ่มต้นจากสถานี สยาม ซึ่งเป็นสถานีแยก จะไปได้ 3 สถานีคือ ชิดลม, สนามกีฬาแห่งชาติ, ราชดำริ

ถ้าเราไปตามแต่ละสถานีจะเขียนได้ดังนี้

อ้างอิง

ชิดลม : สยาม --> ชิดลม
สนามกีฬาแห่งชาติ : สยาม --> สนามกีฬาแห่งชาติ
ราชดำริ : สยาม --> ราชดำริ

หรือในเครื่องคอมพิวเตอร์เราโดยปกติแล้ว ก็มีเรื่องของพารามิเตอร์ของไฟล์เดอร์เช่นเดียวกัน อาทิเช่น

อ้างอิง

Computer --> Local Disk (C:) --> Folder A

Computer --> Local Disk (C:) --> Folder B

Computer --> Local Disk (D:) --> Folder A

อย่างกรณีที่มีรูปวีคลิปป้อนกันดังต่อไปนี้ (แต่ละชื่อคือชื่อ instance name ที่ถูกกำหนดตรง พาเนล properties ไม่ใช่ชื่อของ symbol ในไลบรารี)

Code:

mc1

- mc2

- mc3

กรณีการเขียนโค้ดไว้ที่เฟรมของไทม์ไลน์หลัก ณ จุดเริ่มต้น การอ้างอิงไปยัง mc3 ซึ่งอยู่ระดับชั้นในสุดก็จะเป็นดังนี้

Code:

```
mc1.mc2.mc3
```

ซึ่งจะเห็นว่าไม่ได้มีอะไรซับซ้อน ขอเพียงแค่ระบุตามลำดับชั้นของ instance ที่ซ้อนกันอยู่ แต่เนื่องจากว่า เราสามารถเขียน Action Script ณ ระดับไทม์ไลน์ใดก็ได้ ฉะนั้น การอ้างอิงวัตถุ จะถูกแบ่งออกเป็น 2 รูปแบบคือ

1. Absolute path

โดย Absolute path เป็นการอ้างอิงไทม์ไลน์ที่อยู่ระดับสูงที่สุดขณะแสดงผลงานนั้นๆ (เราสามารถเล่น swf ได้หลายตัวพร้อมๆกันภายในงานๆหนึ่ง ซึ่งจะได้ศึกษาต่อไป) ซึ่งระดับไทม์ไลน์สูงสุด เราจะเรียกว่า root โดยวิธีการอ้างอิง root เราจะใช้คำสั่งชื่อว่า

Code:

```
_root
```

เมื่อเริ่มต้นสร้างไฟล์ fla ขึ้นมา ไทม์ไลน์แรกสุดที่เราเห็นก็คือไทม์ไลน์ที่อยู่สูงที่สุดของงานชิ้นนั้น โดยวิธีการทดสอบพารามิเตอร์นั้นสามารถทดสอบด้วยคำสั่ง trace() ซึ่งใช้เพื่อแสดงผล ซึ่งถ้าเราเขียนคำสั่ง

Code:

```
trace(_root)
```

ไว้ที่ระดับไทม์ไลน์ไหนก็ตาม ก็จะได้ผลลัพธ์เดียวกันคือ _level0 ซึ่งถือเป็นระดับ root การใช้ root นั้นจะมีประโยชน์เมื่อเรามีวัตถุ ที่ซ้อนกันอยู่หลายระดับชั้นมากๆ แล้วต้องการอ้างอิงสิ่งใดๆบนไทม์ไลน์ระดับที่อยู่สูงมากๆ การใช้ _root ถือว่าเป็นทางเลือกที่ดีในการอ้างอิงจากจุดเริ่มต้น ซึ่งเมื่อเราอ้างอิงไปยังระดับไทม์ไลน์สูงสุด เราสามารถไล่ระดับพารามิเตอร์ย้อนกลับลงมาได้ ตัวอย่างเช่นการเขียนโค้ดไว้ที่เฟรมของไทม์ไลน์ มูฟวี่คลิ๊ป mc3 เราต้องการอ้างอิงไปยัง mc1 สามารถระบุถึง mc1 ด้วยการเขียนโค้ดดังนี้

Code:

```
_root.mc1
```

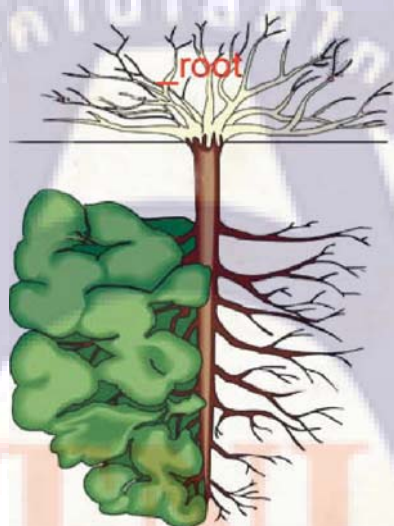
โดยสามารถทดสอบพารด้วยการเขียนดังนี้

Code:

```
trace(_root.mc1);
```

ซึ่งจะแสดงผลเป็น _level0.mc1

กรณีที่เราไม่สามารถอ้างอิงรูปวีคลิปดังกล่าวได้ หน้าต่าง output จะแสดงเป็น undefined การเก็บตัวแปรทั้งหมดเอาไว้ที่ root ที่เดียว ก็ถือเป็นการจัดระเบียบของตัวแปร เพื่อการอ้างอิงและแก้ไขได้สะดวก แต่ก็ให้พิจารณาตามโครงสร้างของงานชิ้นนั้นๆด้วย



ภาพที่ 2.14 ภาพแสดงโครงสร้างแบบต้นไม้กลับหัว โดยมี _root (root อยู่จุดเริ่มต้นของโครงสร้าง)

2. Relative path

relative ถ้าแปลจะหมายถึง เครื่องมือ บุคคลใกล้ชิด สำหรับการเขียนโปรแกรมนั้น relative path จะเป็นการพาไปยังเป้าหมายที่ต้องการโดยอิงจากตำแหน่ง ณ จุดที่เริ่มต้นทำการอ้างอิง โดยการอ้างอิงจากจุดเริ่มต้นนั้น สามารถที่จะอ้างอิงไปในระดับที่ลึกกว่า หรือระดับที่สูงกว่าก็ได้ (ระบุพารย้อนกลับหรือดอยหลัง)

การอ้างอิงแบบ relative นั้นจะใช้คำสั่งอยู่ 2 ตัวคือ

1. this ซึ่งหมายถึงตำแหน่งปัจจุบัน

2. `_parent` เป็นการอ้างอิงแบบย้อนกลับขึ้นไปหนึ่งระดับชั้น

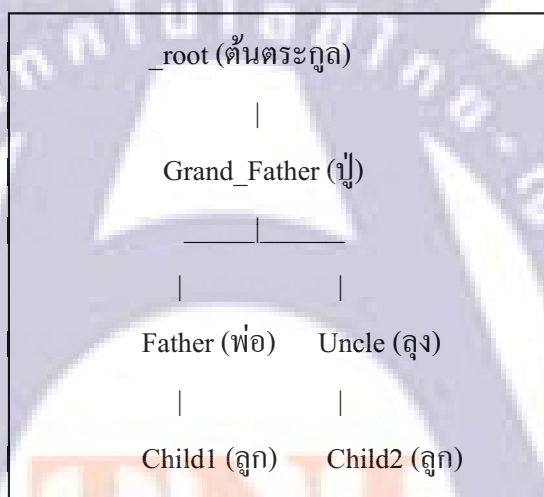
โดยแต่ละตัวมีวิธีการใช้งานดังนี้

1. `this` แปลได้ตรงๆว่า `this` ซึ่งหมายถึงตำแหน่งนี้ ฉะนั้นเมื่อมีการประกาศ `this` ก็หมายถึงการเริ่มต้นอ้างอิงจากตำแหน่งปัจจุบัน

ข้อดีของการใช้ `relative path` มีประโยชน์ตรงที่ เราสามารถอ้างอิงมูฟวีคลิปกจากระดับชั้นที่ลึกมากๆได้เลย

แทนที่จะอ้างอิงจาก `_root` ไปลำดับลงมา ซึ่งอาจจะมีหลายระดับชั้นที่ซ้อนกันอยู่

2. `_parent` ถ้าแปลจะได้ว่า พ่อ แม่ ซึ่งหมายถึงระดับที่อยู่สูงกว่าตำแหน่งปัจจุบันย้อนไปหนึ่งระดับชั้น ซึ่งผังโครงสร้างการซ้อนกันแบบต้นไม้ ถ้าจะมองเป็นแบบผังเครือญาติ ก็ย่อมาได้ ถ้าเราวาดผังออกมาเป็นลำดับตระกูลและเครือญาติ ดังนี้



กรณีที่เราริเขียนโค้ดที่ `Child1` ถ้าเราต้องการจะอ้างอิงถึง `Father` โดยเริ่มต้นเขียนโค้ดที่ `Child1` เราจะอ้างอิงดังนี้

Code:

```
trace(this._parent); //แสดงผลเป็น _level0.Grand_Father.Father
จะเห็นว่ามีอ้างอิง this เพื่ออ้างอิงถึงตำแหน่งปัจจุบันก่อน
```

กรณีที่เราริการอ้างอิง `_parent` ของ `_parent` ก็เท่ากับเป็นการย้อนขึ้นไปสองครั้งตั้งแต่ `Child1` โดยให้เขียนโค้ดดังนี้

Code:

```
trace(this._parent._parent); //ผลคือได้ _level0.Grand_Father
กรณีเขียนโค้ดที่ Child1 แล้วต้องการอ้างอิงไปยัง Child2 เราไม่สามารถระบุ
```

Code:

```
trace(this.Child2); //output ได้ undefined
```

ซึ่งถ้าดูจากผังระดับชั้นแล้ว ดูเหมือนว่า Child1 อยู่ระดับเดียวกับ Child2 แต่อันที่จริงแล้ว การจะมองเห็นกัน ต้องดูที่การซ้อนกันของมูฟวี่คลิป์เป็นหลัก ดังนั้นการที่จะอ้างอิงวัตถุใดๆ จำเป็นต้องไล่ตามพาธในผังเท่านั้น ฉะนั้นการอ้างอิงที่ถูกต้องตามเส้นพาธจะเป็นดังนี้

Code:

```
trace(this._parent._parent.Uncle.Child2); //output แสดง _level0.Grand_Father.Uncle.Child2
```

ซึ่งเมื่อไรที่มีการอ้างอิง `_parent` หลายครั้ง ก่อนที่จะระบุวัตถุในตำแหน่งอื่นๆ เราอาจจะทำการ `trace(this._parent._parent)` ดูก่อนว่าเราซ้อนพาธไปจนถึงระดับที่ต้องการหรือไม่ หรือถ้าเห็นว่าการระบุย้อนกลับไม่ต่างระดับชั้นกับการใช้ `_root` มากก็สามารถใช้ `_root` ได้เลย

การอ้างอิงพาธแบบ Dynamic (Dynamic Path)

นอกจากการอ้างอิงแบบปกติที่กล่าวมาแล้ว เรายังสามารถที่จะอ้างอิงด้วยรูปแบบที่ยืดหยุ่นกว่าเดิมด้วยการอ้างอิงพาธแบบ dynamic ซึ่งจะช่วยลดการอ้างอิงแบบซ้ำๆ ลงไปได้ โดยรูปแบบการอ้างอิงพาธแบบ dynamic จะเป็นดังนี้

กรณี instance , ตัวแปร, properties และ dynamic text เราสามารถอ้างอิงได้เหมือนกันคือ

Code:

```
path[String] หรือ path[String + variable]
```

กรณี function รวมถึง method เราสามารถอ้างอิงด้วยรูปแบบดังนี้

Code:

```
path[String]() หรือ path[String + variable]()
```

โดย path คือ พารามิเตอร์ก่อนหน้าเป้าหมายที่เราต้องการอ้างอิง โดยระบุพารามิเตอร์ตามปกติ แต่สำหรับหน้า [] ให้ละเว้นการระบุเครื่องหมาย "."

String คือ ค่าสตริงใดๆ

variable คือ ตัวแปรใดๆก็ตาม (กรณีที่ variable มีการบวกค่ากับตัวเลขใดๆ ให้ใช้วงเล็บครอบเอาไว้ด้วยเช่น path[String + (variable+1)] มิฉะนั้นค่าหมายเลข 1 จะถูกแปลงให้เป็น String โดยอัตโนมัติ

สำหรับตัวอย่าง dynamic path จะเป็นดังนี้ กรณีที่เรามีมูฟวี่คลิป์ ซ้อนกันอยู่ดังต่อไปนี้

Code:

mc1

- mc2

- mc3

กรณีเราอ้างอิง path ขึ้นต้นแบบ Absolute โดยต้องการอ้างอิงถึง mc3 เราจะระบุดังต่อไปนี้

Code:

_root.mc1.mc2["mc3"]

หรือ

Code:

_root.mc1.mc2["mc"+3];

สำหรับกรณีที่เรามีมูฟวี่คลิป์หลายๆตัวเช่น mc1,mc2, mc3 วางอยู่ที่เดียวกันหมด การอ้างอิงถึงมูฟวี่คลิป์แต่ละตัวจะเป็นการเสียเวลา ฉะนั้นเรามักนิยมที่จะนำคำสั่ง for ซึ่งใช้ในการวนลูปมาช่วงดังนี้

Code:

```
for (var i=1,i<=3;i++) { // สั่งให้วนลูป 3 รอบ โดยค่า i จะมีค่าตั้งแต่ 1 ถึง 3
    trace(_root["mc" + i]);
}
```

Code:

```
_level0.mc1
_level0.mc2
_level0.mc3
```

ซึ่งจะเห็นว่า เรานำตัวแปร i มาบวกเข้ากับ String และใช้ [] ครอบเท่านั้นเอง ก็จะช่วยลดระยะเวลาในการเขียนโค้ดได้มากขึ้น และยังทำให้โค้ดมีความกระชับและกะทัดรัดอีกด้วย

ตัวอย่างอ้างอิงฟังก์ชันแบบ dynamic

สำหรับการใช้กับฟังก์ชัน ก็เพียงแค่เติม () ต่อท้าย [] ก็จะทำให้เราสามารถเรียกฟังก์ชันที่ต้องการได้แล้ว

ตัวอย่างการเรียกฟังก์ชันแบบ dynamic เช่น เราเขียนฟังก์ชันเอาไว้ฟังก์ชันหนึ่ง และต้องการให้เรียกฟังก์ชันที่ต้องการหลังจากทำงานฟังก์ชันนั้นจบ แต่บางครั้งฟังก์ชันที่จะเรียกหลังทำงานจบ อาจจะไม่ใช่ฟังก์ชันเดียวกันเสมอไป ฉะนั้นเราจะเขียนโค้ดดังนี้

Code:

```
function testDynamicFunction(path, callback, param) {
    //statement
    path[callback](param);
}

function myCallBack(param) {
    trace(param);
}

testDynamicFunction(this,"myCallBack", "myCallBack is called");
```

ซึ่งจะแสดงผล myCallBack is called ซึ่งการเรียกฟังก์ชัน testDynamicFunction จะส่งพารามิเตอร์เข้าไป โดยการระบุ this ก็คือตำแหน่งปัจจุบัน และส่งฟังก์ชันที่ต้องการเรียกกลับหลังจากจบการทำงาน ก็คือ myCallBack และมีการส่งพารามิเตอร์ให้กับฟังก์ชันอีกด้วย เมื่อ myCallBack ถูกเรียก จึงนำค่าพารามิเตอร์ดังกล่าวมาแสดงผล

- การประกาศ ตัวแปร (variable) และ ฟังก์ชัน (function) ให้กับ target

เราสามารถทำการประกาศตัวแปร และฟังก์ชันให้กับ target หรือ instance ที่เราต้องการ ด้วยรูปแบบง่าย ๆ ดังต่อไปนี้

Code:

```
target.variable = value
```

หรือแบบ dynamic คือ

Code:

```
target["variable"] = value
```

โดย variable คือชื่อตัวแปรและ value คือค่าที่ต้องการกำหนดให้ส่วนการประกาศฟังก์ชัน จะมีรูปแบบดังต่อไปนี้

Code:

```
target.variable = function() {
```

```
    //statement
```

```
}
```

หรือ

Code:

```
function functionName = function() {
```

```
    //statement
```

```
}
```

```
target.variable = functionName
```

ซึ่งทั้งสองรูปแบบสามารถเรียกใช้ด้วยการระบุ

Code:

```
target.variable();
```

หรือจะประกาศโดยใช้หลักการแบบ dynamic ก็ย่อมได้เช่นเดียวกันดังนี้

Code:

```
target["functionName"] = function() {
    //statement
}
```

หรือ

Code:

```
target["functionName"+variable] = function() {
    //statement
}
```

ส่วนวิธีการเรียกใช้นั้น จะเรียกใช้แบบอ้างอิงชื่อโดยตรง หรือแบบ dynamic ก็ย่อมได้เช่นเดียวกัน

- สิ่งที่ [] ทำไม่ได้ แต่ eval ทำได้

สมัยก่อนนั้น ก่อนที่จะใช้ [] มาใช้ในการอ้างอิง path แบบ dynamic หลายคนส่วนใหญ่จะนิยมใช้ eval มาก่อน แต่เนื่องจากว่าผมได้ใช้ [] จนชิน เพราะ [] นั้นเขียนสั้นและเข้าใจง่ายกว่า ซึ่งการใช้ eval ในวิธีการเดียวกับ [] สามารถยกตัวอย่างได้ดังนี้

Code:

```
var variable1:Number = 10;
var no:Number = 1;
trace(eval("variable"+no)); //จะได้ค่าเท่ากับ 10
ซึ่งการเขียนด้านบนนี้จะเขียนแทนด้วย [ ] ดังนี้ trace(this["variable"+no]);
```

แต่ทั้งนี้ [] ก็มีข้อจำกัดอยู่อย่างหนึ่งก็คือ มันไม่สามารถนำ path ในรูปแบบ String ที่มากกว่า 1 ระดับมาเปลี่ยนเป็น path จริงๆได้ ตัวอย่างเช่นมีมูฟวีคลิ๊ปซ้อนกันอยู่แบบนี้

Code:

```
mc1
    -mc2
ถ้าเรามี String เช่น "mc1.mc2"
```

ถ้าหากเราเขียนโค้ดว่า

Code:

```
var myPath:String = "mc1.mc2";
trace(this[myPath]); //undefined
trace(eval(myPath)); //_level0.mc1.mc2
```

ซึ่ง [] จะไม่สามารถอ้างอิง mc1.mc2 ได้ แต่ eval จะสามารถอ้างอิงมูฟวี่คลิปลตาม path ดังกล่าวได้ ซึ่งถือว่าเป็นข้อดีที่ควรเลือกใช้ให้เหมาะสมในบางกรณี เช่นค่าตัวแปรที่รับค่ามาจากนอก swf ในรูปแบบ String ซึ่งบางครั้งต้องการส่งค่าตัวแปรเป็น path ของมูฟวี่คลิปลมากกว่า 1 ระดับ ก็สามารถใช้คำสั่งนี้ช่วยได้

2.1.6.1.2 Method

เมธอดคือวิธีการ สำหรับกรณีนี้จะเรียกว่าการกระทำ หรือ การดำเนินการ แต่ถ้าแปลให้เข้าใจง่ายในเชิง OOP เมธอดก็เปรียบเสมือนความสามารถที่วัตถุนั้นสามารถกระทำได้นั่นเอง ซึ่งอันที่จริงแล้วเมธอดนั้นก็คือฟังก์ชันที่ถูกกำหนดไว้ใน คลาสนั้นๆนั่นเอง โดยรูปแบบการใช้งานจะไม่ต่างอะไรกับฟังก์ชันมากนักโดยรูปแบบการใช้งานเป็นดังนี้

Code:

```
target.method();
```

หรือ

Code:

```
target.method(parameter1,parameter2,...,parameterN);
```

ซึ่งค่าพารามิเตอร์ที่จะให้กรอกนั้น ขึ้นอยู่กับว่าเมธอดนั้นถูกกำหนดให้สามารถส่งค่าอะไรได้บ้าง นอกจากนี้ method เดียวกันไม่จำเป็นต้องส่งค่าพารามิเตอร์รูปแบบเดียวกันเสมอไป (มีการทำ overloading เอาไว้)

- การควบคุม Timeline และ Movie Clip Timeline

โดยปกติแล้ว เรามักจะได้เรียนรู้วิธีการควบคุมไทม์ไลน์ หรือตำแหน่งของ play head เป็นพื้นฐานก่อนเสมอในเบื้องต้น โดยจะควบคุมการเล่น การหยุด หรือการกระโดดไปยังเฟรมอื่น โดยคำสั่งที่ใช้ควบคุมทั้งหมดจะมีดังนี้

play()	สำหรับเล่น
stop()	สำหรับหยุด
nextFrame()	ไปข้างหน้า 1 เฟรมจากตำแหน่งเฟรมปัจจุบัน
prevFrame()	ถอยหลัง 1 เฟรมจากตำแหน่งเฟรมปัจจุบัน
gotoAndPlay(frame)	กระโดดไปยังเฟรมที่ต้องการและทำการเล่นต่อ (frame กำหนดค่าเป็นตัวเลข)
gotoAndPlay(label)	กระโดดไปยังเฟรมที่มีการประกาศชื่อลาเบลที่ระบุ (label กำหนดค่าเป็น String)
gotoAndStop(frame)	กระโดดไปยังเฟรมที่ต้องการและทำการหยุด ณ เฟรมนั้น
gotoAndStop(label)	กระโดดไปยังเฟรมที่มีการประกาศชื่อลาเบลที่ระบุ และทำการหยุด ณ เฟรมนั้น
gotoAndPlay(Scene,frame)	กระโดดไปยังซีนที่ต้องการ และกำหนดค่าเฟรมที่ต้องการในซีนนั้นๆ และกำหนดให้เล่นต่อจากเฟรมดังกล่าว(Scene กำหนดค่าเป็น String)
gotoAndStop(Scene,frame)	กระโดดไปยังซีนที่ต้องการ และกำหนดค่าเฟรมที่ต้องการในซีนนั้นๆ และกำหนดให้หยุดที่เฟรมดังกล่าว

กรณีที่เราได้เขียนโค้ดดังกล่าวเอาไว้ที่ไทม์ไลน์ และไม่มีการระบุพารามิเตอร์นำหน้าคำสั่ง ให้ถือเสียว่าเป็นการระบุด้วย this เพื่ออ้างอิงตัวไทม์ไลน์ หรือมูฟวี่คลิป์ซึ่งเป็นเจ้าของไทม์ไลน์นั้นๆ

กรณีมีการกำหนดให้ข้ามเฟรมโดยระบุค่าเป็นชื่อซีน ถือเป็นกรณีพิเศษที่ห้ามระบุพารามิเตอร์นำหน้าทั้งสิ้น ซึ่งถ้ามีการกำหนดพารามิเตอร์นำหน้าคำสั่ง จะทำให้คำสั่งไม่สามารถทำงานได้ ซึ่งการกระโดดแบบซีนนั้น จะใช้ได้เมื่อเขียนโค้ดเอาไว้ในไทม์ไลน์ใดๆ หรือกำหนดภายใน event ของ instance ประเภท Button เท่านั้น (Button มีคุณสมบัติต่างจาก MovieClip อยู่อย่างหนึ่งก็คือ ถ้าเราไม่ระบุพารามิเตอร์ที่เป็นค่า default จะเปรียบได้กับการระบุ _parent ซึ่งจะทำให้ย้อนขึ้นไปหนึ่งระดับชั้นแทนที่จะหมายถึงตัว Button นั้นๆ) ให้ทำความเข้าใจในหัวข้อ event เสริมเพิ่มเติมในภายหลัง

กรณีที่ต้องการควบคุมไทม์ไลน์ของ MovieClip อาทิเช่น เรามี MovieClip ตั้งชื่อ instance name ไว้ว่า my_mc เราก็สามารถกำหนดได้ดังต่อไปนี้ (เขียนโค้ดบนไทม์ไลน์ในระดับที่ my_mc วางอยู่)

Code:

```
my_mc.play();
```

จะเห็นว่าวิธีการนั้นง่าย ๆ ก็คือการกำหนดเป้าหมายหรือระบุ target path ก่อนแล้วจึงทำการเรียก method ที่ต้องการ

- ความสัมพันธ์ระหว่าง method และ function

จากได้บอกไปแล้วในตอนต้นว่า method ก็คือ function ที่อยู่ใน Class ฉะนั้นจะเห็นว่ารูปแบบการใช้งาน method จึงไม่ต่างอะไรกับการใช้งานฟังก์ชัน

และถ้าสังเกตให้ดี การเรียกใช้งาน method นั้นก็เหมือนการเรียก function ซึ่งอยู่ในมูฟวี่คลิบ ฉะนั้นรูปแบบที่เรียกจึงออกมาแบบ mc.function() นั่นเอง ซึ่ง MovieClip ก็ถือว่าเป็น Class ประเภทหนึ่ง

เราอาจจะพูดได้ว่า method ก็คือ built in function หรือ function ภายในที่ได้ถูกสร้างเอาไว้แล้วภายใน Class นั้น เราจึงสามารถเรียกใช้งานได้ทันที

- คำสั่งที่เป็นทั้ง method และ function

บางครั้งบาง method นั้นก็สามารถที่จะใช้ในรูปแบบฟังก์ชันได้ด้วย สาเหตุเนื่องจากแต่เดิมทีนั้น คำสั่งนั้นถูกใช้งานในรูปแบบฟังก์ชันมาก่อน ก่อนที่จะมีการใช้งาน target path ตัวอย่างเช่นคำสั่ง duplicateMovieClip() ซึ่งใช้ในการสร้าง MovieClip ขึ้นมาใหม่จากต้นแบบ target ที่กำหนด โดยเมื่อเราพิมพ์คำสั่ง

duplicateMovieClip(โปรแกรมจะแสดง short cut เพื่อแสดงค่าพารามิเตอร์ที่สามารถกำหนดได้ดังนี้

Code:

```
duplicateMovieClip(target,newname="",depth);
```

ซึ่งจะเห็นว่ามีกาให้กำหนด target หรือ MovieClip ต้นแบบที่จะนำมาสร้างใหม่, ชื่อใหม่ ของ MovieClip ที่ถูกสร้าง และสุดท้ายคือระดับความลึกของ MovieClip ที่ถูกสร้าง

ซึ่งรูปแบบดังกล่าวเป็นรูปแบบการใช้งานแบบ function แต่กรณีที่เราพิมพ์คำสั่งในรูปแบบ method เช่น

my_mc.duplicateMovieClip(โปรแกรมจะแสดง short cut เพื่อแสดงค่าพารามิเตอร์ที่สามารถ กำหนดได้ดังนี้

Code:

```
my_mc.duplicateMovieClip(newname="",depth, [initObject])
```

ซึ่งการกำหนดค่า target จะหายไป แต่จะมีให้กำหนด initObject มาแทน ซึ่งเป็นวัตถุที่เก็บ คุณสมบัติต่างๆเอาไว้และเราต้องการถ่ายทอดคุณสมบัติดังกล่าวให้กับ MovieClip ที่ถูกสร้างขึ้น มาใหม่

ในการกำหนดค่าพารามิเตอร์ ซึ่งค่าใดที่มี [] ครอบอยู่ จะหมายถึงทางเลือกที่คุณสามารถส่ง ค่าหรือไม่ก็ได้ (โดยส่วนใหญ่ เอกสารคู่มือหรือ help สำหรับแต่ละภาษาจะใช้สัญลักษณ์นี้ เช่นเดียวกัน)

- การใช้ method ของ Class หนึ่งเพื่อควบคุม MovieClip และ Class ที่ทำงานด้วยตัวเองในการสร้าง object จาก class ใดๆนั้น

การสร้าง class นั้นจะแบ่งเป็น class อยู่ 2 ประเภทคือ

1. class ที่สร้าง object ขึ้นมาเองโดยไม่อิงกับ MovieClip เช่น Date, Mouse, Keyboard และอื่นๆ
2. class ที่สร้าง object ขึ้นมาโดยอิงกับ MovieClip เช่น Sound, Color และอื่นๆ

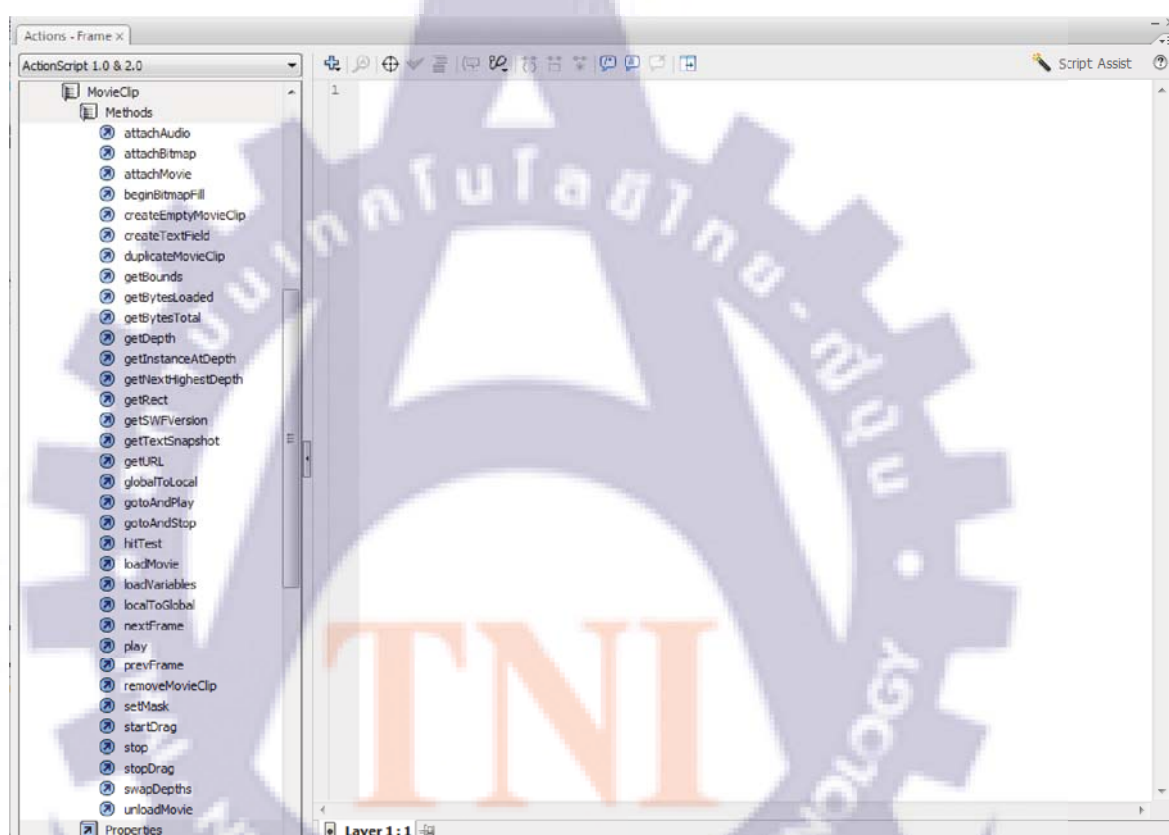
โดยการสร้าง class ประเภทที่ 2 นั้นจะต้องทำการอ้างอิง MovieClip ตอนสร้าง ตัวอย่างเช่น Class Sound หรือ Color เป็นต้น โดยตัวอย่างการสร้างจะเป็นดังนี้

Code:

```
my_sound = new Sound(my_mc);  
my_color = new Color(my_mc);
```

สำหรับ Sound นั้นกรณีไม่ระบุ target ลงไป ก็จะทำให้การอ้างอิงใหม่ในโค้ดที่เขียนโค้ดนั้นๆ โดยอัตโนมัติ ส่วน Color นั้น สาเหตุที่เราจำเป็นต้องระบุ target ก็เพื่อให้รู้ว่าวัตถุที่เราต้องการทำการเปลี่ยนสีนั่นเอง

จากด้านล่างเป็นตัวอย่างของการเรียกใช้ method ของแต่ละ Class โดย MovieClip นั้นสามารถใช้ method ชื่อ duplicateMovieClip() ในการสร้างวัตถุนั้นๆขึ้นมาใหม่ และสามารถลบทิ้งได้ด้วยคำสั่ง removeMovieClip() เป็นต้น นอกจากนี้ เมื่อ target ที่ถูกอ้างอิงหายไป object ที่สร้างขึ้นจาก Class นั้นก็จะไม่สามารถทำงานได้อีก



ภาพที่ 2.15 method ของ MovieClip

2.1.6.1.3 Properties

สำหรับ properties นั้น หรือ property แปลได้ว่า คุณสมบัติ ซึ่งคุณสมบัติเป็นสิ่งที่แสดงลักษณะอย่างใดอย่างหนึ่งของวัตถุหรือ Class ใดๆ อย่างเช่นเรื่องของคุณภาพ ความสูง ความกว้าง หรือแม้แต่เสียง (Sound) ก็มีเรื่องของความดัง (volume) เป็นต้น (แต่บางครั้งอาจจะถูกกำหนดค่าผ่าน method ก็ได้)

ถ้ามองในแง่ของ OOP หรือการเขียนโปรแกรมเชิงวัตถุ properties นั้นก็เทียบได้กับตัวแปร (variable) ที่ถูกกำหนดภายใน Class นั้นๆ ซึ่งหนังสือบางเล่มอาจจะเรียกว่า instance variable (instance หรือ object นั้นถูกสร้างขึ้นจาก Class) ดังนั้นรูปแบบการใช้งาน properties จึงไม่ต่างอะไรกับการอ้างถึงตัวแปรที่อยู่ภายในเป้าหมาย (target) ที่เราต้องการ (ถ้านึกไม่ออก ให้คิดถึงตัวแปรที่ประกาศอยู่ใน MovieClip)

- รูปแบบการใช้งาน properties

รูปแบบการใช้งานจะแบ่งเป็น 2 กรณีดังต่อไปนี้

1. property แบบที่อ่านค่าได้อย่างเดียว

ซึ่งจะสามารถเรียกใช้งานเพื่อดูค่าได้เท่านั้น ไม่สามารถกำหนดค่าอะไรให้ได้ โดยจะมีรูปแบบคือ

Code:

target.property

โดยทำการระบุ target ที่ต้องการตามหลักการระบุ target path ก่อน หลังจากนั้นจึงทำการระบุ property ของ Class นั้นๆ

2. property แบบที่อ่านค่าได้และกำหนดค่าได้

สำหรับการอ่านค่านั้นจะมีรูปแบบเดียวกันกับข้อ 1 ส่วนวิธีการกำหนดค่า จะใช้รูปแบบดังนี้

Code:

target.property = value

ซึ่งวิธีการกำหนดค่า เพียงแค่ใช้เครื่องหมายเท่ากับ และทางด้านขวาก็คือค่าที่จะกำหนด ซึ่งต้องมีชนิดข้อมูลตรงกับที่ property นั้นต้องการ ซึ่งจะเห็นว่ารูปแบบการกำหนดค่าเหมือนกับการกำหนดค่าให้กับตัวแปรนั่นเอง

- properties ยุคแรกกับยุคหลัง

ในช่วงแรกของ ActionScript นั้น รูปแบบคำสั่ง properties นั้นจะใช้ "_" นำข้างหน้า ตัวอย่างเช่น properties หลายๆตัวของ MovieClip จะเป็นดังนี้

- | | |
|------|--|
| - _x | อ่านค่าหรือกำหนดตำแหน่งในแนวแกน x บน Stage |
| - _y | อ่านค่ากำหนดตำแหน่งในแนวแกน y บน Stage |

- _width	อ่านค่าหรือกำหนดความกว้างให้กับ MovieClip หน่วยเป็นพิกเซล
- _height	อ่านค่าหรือกำหนดความสูงให้กับ MovieClip หน่วยเป็นพิกเซล
- _xscale	อ่านค่าหรือกำหนดความกว้างให้กับ MovieClip หน่วยเป็นเปอร์เซ็นต์ (ปกติคือ 100%)
- _yscale	อ่านค่าหรือกำหนดความสูงให้กับ MovieClip หน่วยเป็นเปอร์เซ็นต์ (ปกติคือ 100%)

จากด้านบน เป็นตัวอย่าง properties บางส่วนของ MovieClip เท่านั้น ซึ่งจะเห็นว่าจะมี _ นำหน้าอยู่ แต่ตอนหลัง properties หลายๆตัวได้ถูกตัด _ ออกไป (ส่วนใหญ่จะมีเหลือแค่ของ MovieClip ที่ยังคงไว้ตามเดิมเพื่อยังคงสนับสนุนไฟล์งานที่พัฒนาเอาไว้ก่อนหน้านี้) โดยเฉพาะ AS3 จะทำการตัดออกไปเลย จาก _x ก็จะเป็น x ฉะนั้นบางทีเราไม่ควรกำหนดตัวแปรให้กับมูฟวี่คลิปด้วยชื่อที่ซ้ำกับ properties ถึงแม้จะไม่มีข้อห้ามก็ตาม

- properties ที่ถูกกำหนดค่าและดึงค่าด้วย method แทน

บางครั้งการกำหนดค่า properties หรือ ดึงค่า properties นั้นไม่ได้อยู่ในรูปแบบของการกำหนดค่าแบบตัวแปรตรงๆเสมอไป แต่อาจจะกำหนดค่าผ่านทาง method อื่นๆก็ได้ ซึ่งการเลือกใช้ method เพื่อกำหนด properties นั้นจะมีข้อดีกว่าตรงที่เราสามารถกำหนดเงื่อนไขให้กับค่าที่จะกำหนดให้กับ properties นั้นๆได้

รูปแบบของคำสั่ง

Code:

```
setProperty()  
getProperty()
```

ซึ่ง method ที่ขึ้นต้นด้วย set ก็คือการกำหนดค่า และ method ที่ขึ้นต้นด้วย get ก็คือสำหรับดึงค่า ซึ่งในตอนนี้อาจยังไม่ได้เริ่มต้นเขียนโปรแกรมในรูปแบบของ OOP (ซึ่งจะได้ศึกษาในภายหลัง ตอนนี้เพียงแค่ว่า) ฉะนั้นจะยกตัวอย่างในรูปแบบการเขียน function ภายในมูฟวี่คลิป

โดยสมมุติว่า บนไทม์ไลน์มูฟวี่คลิปชื่อ my_mc มีการเขียน function ดังนี้

Code:

```
function setSize(w,h) {
    this._width = w; //กำหนดความกว้าง
    this._height = h; //กำหนดความสูง
}
function getWidth() {
    return this._width; //คืนค่าความกว้าง
}
function getHeight() {
    return this._height; //คืนค่าความสูง
}
```

เมื่อเรียกใช้ ก็จะเรียกใช้ในรูปแบบ method ตามปกติ

Code:

```
my_mc.setSize(100,50);
myWidth = my_mc.getWidth();
myHeight = my_mc.getHeight();
trace(myWidth); //100
trace(myHeight); //50
```

สำหรับการกำหนดหรือดึงค่า properties ในรูปแบบ method นั้นจะมีการใช้งานอย่างมากในตอนที่มีการสร้าง Class ขึ้นมาเอง

ตัวอย่าง method ที่ดึงและกำหนดค่า properties ของ Class ที่มีอยู่แล้ว อาทิเช่น

Sound - จะมี setVolume(), getVolume()

Color - จะมี setRGB(), getRGB()

ซึ่งจะกำหนดให้การกำหนด properties นั้นถูกกระทำผ่านตัวแปรโดยตรง หรือผ่าน method ก็ขึ้นอยู่กับวิธีการดีไซน์ Class ของแต่ละคน

สำหรับหัวข้อนี้เราได้เรียนรู้การใช้งาน properties ในการกำหนด และดึงค่ามาใช้เรียบร้อยแล้ว จะเห็นว่ารูปแบบการใช้งานนั้นไม่ได้ต่างอะไรกับการใช้งานตัวแปรเลยแม้แต่น้อย

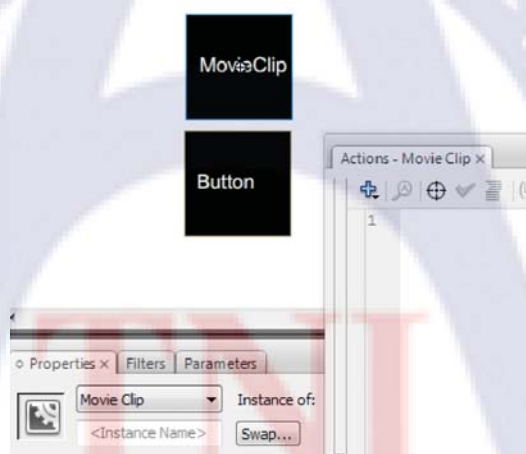
2.1.6.1.4 Events

Events เป็นตัวที่กำหนดการตอบสนองสิ่งต่างๆให้เกิดขึ้นเมื่อ Events นั้นๆเกิดขึ้น เช่นเรา กำหนด Events ให้กับปุ่มๆหนึ่ง ซึ่งอาจทำการกำหนดเอาไว้ว่า ถ้าโดนกด ก็จะทำการเปลี่ยนเฟรม เป็นต้น ซึ่ง Class แต่ละประเภทจะมีรูปแบบ Events หลากๆแบบรองรับ

-รูปแบบการกำหนด Events ให้กับ instance ใดๆ

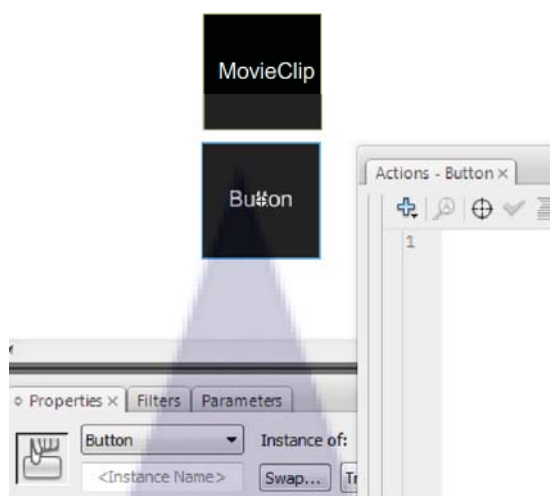
1. การกำหนด Events บนตัว Instance

การกำหนดอีเวนต์บนตัว Instance นั้นสามารถทำได้โดยการคลิกที่ instance ที่เราต้องการ แล้วทำการเปิด Actions Panel ขึ้นมา ซึ่งจากภาพตัวอย่างจะเห็นว่า MovieClip สีเหลี่ยมนั้นโดนคลิกเลือกอยู่ (ขึ้นกรอบสีน้ำเงิน) และให้สังเกตชื่อตรง Actions Panel จะบอกว่าเรากำลังเขียนโค้ดอยู่ที่ไหน



ภาพที่ 2.16 ตัวอย่างการใช้งาน Events

จะเห็นว่าขึ้น Actions – MovieClip และเมื่อทำการคลิกเลือก instance ประเภท Button ก็จะขึ้นแสดงผลว่า Actions - Button ดังภาพ



ภาพที่ 2.17 ตัวอย่างการใช้งาน Events

สำหรับการประกาศอีเวนต์บนตัว instance จะมีรูปแบบดังต่อไปนี้

- กรณี instance เป็น MovieClip จะประกาศว่า

Code:

```
onClipEvent(event) {
    //statement
}
```

โดย event คือชื่ออีเวนต์ต่างๆที่ MovieClip รองรับ

- กรณี instance เป็น Button จะประกาศว่า

Code:

```
on(event) {
    //statement
}
```

โดย event คือชื่ออีเวนต์ต่างๆที่ Button รองรับ

สำหรับชื่อ event ของ MovieClip และ Button นั้น เราจะได้ศึกษาในหัวข้อหลังจากนี้

Note - การเขียนโค้ดบนตัว instance นั้นจำเป็นต้องเขียนภายใต้ events เสมอ มิฉะนั้นโปรแกรมจะทำการเตือน error ว่า Statement must appear within on/onClipEvent handler ซึ่งมักจะเกิดขึ้น

บ่อยครั้งเมื่อเราเข้าใจว่ากำลังเขียนโค้ดอยู่บน Timeline ในขณะที่ตำแหน่งจริงๆของโค้ดนั้นอาจจะอยู่บน instance ตัวใดตัวหนึ่ง (สามารถคลิกสองที่ที่ error ดังกล่าวเพื่อให้แสดงตำแหน่งที่เขียนโค้ดผิดได้ทันที แต่บางครั้งก็ต้องดูคำเตือนด้วยว่าอยู่ ณ บรรทัดใดกันแน่)

2. การกำหนด Event บน Timeline

หลังจาก flash mx (flash 6) ออกมา ทำให้เราสามารถประกาศ events ต่างๆให้กับ instance ได้สะดวกมากขึ้น โดยสามารถที่จะกำหนด events โดยการเขียนโค้ดบน Timeline ณ ตำแหน่งใดๆก็ได้ โดยอาศัยหลักการระบุ target path เข้ามาช่วยอ้างอิง instance ที่ต้องการ ซึ่งรูปแบบการประกาศ events บน Timeline จะเป็นดังนี้ (Actions Panel จะแสดงตรงหัวว่า Actions -Frame เพื่อให้รู้ว่ากำลังเขียนโค้ดอยู่บนเฟรมของไทม์ไลน์)

Code:

```
target.onEvent = function() {
    //statement
}
```

หรือ การประกาศฟังก์ชันใดๆไว้แล้วนำฟังก์ชันดังกล่าวมาผูกหรือเชื่อมต่อ (handle) เข้ากับ event นั้นๆ

Code:

```
function functionName() {
    //statement
}
target.onEvent = functionName
หรือ
```

Code:

```
functionName = function() {
    //statement
}
target.onEvent = functionName
```

โดย Event คือชื่ออีเวนต์ต่างๆที่ target นั้นๆ รองรับ (ให้ขึ้นต้นด้วยตัวอักษรใหญ่เป็นตัวแรก ซึ่งจะเป็นชื่อเดียวกับการประกาศอีเวนต์บนตัว instance) และ functionName คือชื่อฟังก์ชันใดๆที่เราได้ประกาศไว้ และต้องการให้ทำงานเมื่อเกิด Event ขึ้น

สำหรับหัวข้อนี้เราได้เรียนรู้รูปแบบการประกาศอีเวนต์แบบต่างๆให้กับ instance เรียบร้อยแล้วซึ่งเราจะได้เรียนรู้เพิ่มเติมอีเวนต์พื้นฐานต่างๆของ Button และ MovieClip มากขึ้น รวมถึงการตอบโต้และตอบสนองในรูปแบบต่างๆที่เกิดจากการประกาศอีเวนต์นั้นๆ

- Events พื้นฐานต่างๆของ Button และ MovieClip ที่ควรรู้

สำหรับอีเวนต์พื้นฐานของทั้ง Button และ MovieClip นั้นขอแสดงในรูปแบบการประกาศอีเวนต์บน Timeline เพื่อให้ตรงกับที่แสดงผลในโปรแกรม ซึ่งจะขอยกตัวอย่างเมื่อนำไปใช้ในรูปแบบที่เขียนบน instance สักเล็กน้อยเช่นคำสั่ง onPress ถ้าเขียนไว้บนตัว instance จะใช้คำที่อยู่หลังคำว่า on และเป็นตัวอักษรเล็กขึ้นต้น โดยจะได้ว่า

Code:

```
on(press) {  
}
```

Button

- onDragOut
- onDragOver
- onKeyDown
- onKeyUp
- onKillFocus
- onPress
- onRelease
- onReleaseOut
- onRollOut
- onRollOver
- onSetFocus

MovieClip

- onData
- onDragOut
- onDragOver
- onEnterFrame
- onKeyDown
- onKeyUp
- onKillFocus
- onLoad
- onMouseDown
- onMouseMove-onPress
- onRelease
- onReleaseOutside
- onRollOut
- onRollOver
- onSetFocus
- onUnload

- การอ้างอิง this ภายใต้อัน events หรือ function ที่ผูกอยู่กับ events

โดยปกติแล้ว เมื่อเรามีการอ้างอิงถึง this ภายใต้อัน function ทั่วไป จะหมายถึงการอ้างอิงถึง timeline ที่ function นั้นโดนประกาศไว้ แต่กรณีที่มีการอ้างอิง this ภายใต้อัน event function ให้เราจำเอาไว้เสมอว่า this นั้นจะหมายถึงวัตถุที่ทำการประกาศ events เอาไว้ ตัวอย่างเช่น

Code:

```
my_mc.onPress = function() {
    trace(this); // _level0.my_mc โดยที่ my_mc อยู่บน _root หรือ Main Timeline
}
```

หรือกรณีที่ function นั้นถูกนำมาผูกกับ event ก็จะได้ผลเช่นเดียวกัน เช่น

Code:

```
function myEventHandle() { //หรือ myEventHandle = function() {
    trace(this); // _level0.my_mc โดยที่ my_mc อยู่บน _root หรือ Main Timeline
}
my_mc.onPress = myEventHandle;
```

ซึ่งการใช้ this เพื่ออ้างอิง จะช่วยให้เราอ้างอิงวัตถุต่างๆได้ในทันที รวมถึงวัตถุที่อยู่ในระดับเดียวกัน โดยใช้ this._parent ช่วย

การเรียกใช้อีเวนต์ที่กำหนดไว้ในรูปแบบ function

เมื่อเราทำการประกาศอีเวนต์ให้กับ object หรือ instance ผ่านทาง Timeline นั้น ซึ่งถ้าดูรูปแบบการประกาศซึ่งอยู่ในลักษณะดังต่อไปนี้

Code:

```
my_btn.onPress = function() {
    trace("press");
}
```

จะเห็นได้ว่า รูปแบบของอีเวนต์ที่โดนประกาศจะไม่ได้ต่างอะไรกับการประกาศฟังก์ชันให้กับมูฟวี่คลิปสักเท่าไร ดังนั้นเราสามารถที่จะทำการเรียกอีเวนต์ที่ถูกประกาศในรูปแบบของฟังก์ชันได้ดังนี้

Code:

```
my_btn.onPress = function() {
    trace("press");
}
my_btn.onPress(); //output ได้ press
```

นอกจากนี้ เรายังสามารถกำหนดตัวแปรพารามิเตอร์ได้อีกด้วย

Code:

```
my_btn.onPress = function(param) {
    trace(param);
}
my_btn.onPress("press"); //output ได้ press
```

ซึ่งการเรียกใช้งานในรูปแบบฟังก์ชันนี้ จะมีประโยชน์เช่น บางครั้งเราทำการประกาศอ็อบเจกต์ให้กับปุ่มหลายๆด้วยการใช้คำสั่งวนลูปและอ้างอิงปุ่มแต่ละปุ่มแบบ dynamic ซึ่งบางทีเราต้องการจะให้ปุ่มแรกนั้นโดนกดเอาไว้เรียบร้อยแล้ว ฉะนั้นเราสามารถเช็คค่าตัวแปรว่าเป็นค่าแรกของคำสั่งวนลูปหรือไม่ ซึ่งถ้าใช่ เราก็สามารถที่จะกำหนดให้มีการกดปุ่มนั้นเกิดขึ้น ดังตัวอย่างต่อไปนี้

Code:

```
//อ้างอิง btn1, btn2, btn3
for (var i=1;i<=3;i++) {
    var btn = _root["btn"+i];
    btn.onPress = function() {
        trace(this);
    }
    if(i==1) {//ถ้าค่า i เท่ากับ 1
        btn.onPress(); //output ได้ btn1
    }
}
```

- ข้อแตกต่างระหว่างการประกาศอีเวนต์ทั้ง 2 ตำแหน่ง

หลังจากที่ได้เรียนรู้ไปเรียบร้อยแล้วถึงวิธีการประกาศอีเวนต์ทั้ง 2 ตำแหน่ง ในหัวข้อนี้เราจะทำความเข้าใจในข้อแตกต่างระหว่าง 2 รูปแบบเพื่อเลือกใช้ตามความเหมาะสม ซึ่งมีดังต่อไปนี้

1. การฝังและคงอยู่ของอีเวนต์ สำหรับตัว instance

การประกาศอีเวนต์บนตัว instance นั้นจะเปรียบเสมือนเป็นการฝังโค้ดลงไปในตัว instance นั้นๆเลย ซึ่งกรณีที่มีการใช้คำสั่งในการสร้าง instance ตัวใหม่ขึ้นมา โดยทำการอ้างอิงตัว instance ดังกล่าวที่ได้ประกาศอีเวนต์เอาไว้เป็นตัวต้นแบบ จะทำให้ตัว instance ตัวใหม่นั้นได้รับอีเวนต์ที่ได้ประกาศเอาไว้ไปด้วย และทำงานด้วยอีเวนต์เดียวกันโดยไม่จำเป็นต้องประกาศใหม่

ในขณะที่การประกาศอีเวนต์ให้กับ instance บน Timeline นั้น จะให้ผลที่แตกต่างกันคือ instance ที่ถูกสร้างใหม่จะไม่ได้รับอีเวนต์จากตัวต้นแบบไปด้วย โดยจะยึดถือว่าตัวต้นแบบนั้นเป็น instance ที่ยังไม่มีประกาศอีเวนต์ใดๆเอาไว้ทั้งสิ้น เนื่องจากการประกาศอีเวนต์ที่เกิดขึ้นภายหลังซึ่งจะมีผลกับ instance ตัวนั้นๆเพียงตัวเดียว ฉะนั้นการที่จะให้ instance ที่ถูกสร้างใหม่ทำงานด้วยอีเวนต์เดียวกัน จึงจำเป็นต้องทำการประกาศอีเวนต์ด้วยวิธีการเดียวกันกับ instance ที่เป็นตัวต้นแบบ

2. การทำงานที่เกิดขึ้นแบบ หลายอีเวนต์ (multi process) และ ทำงานเพียงอีเวนต์เดียว (one process)

ข้อแตกต่างอีกข้อของการประกาศอีเวนต์บนตัว instance กับ การประกาศบน Timeline จะให้ผลที่แตกต่างกันก็คือ กรณีที่มีการประกาศอีเวนต์ซ้ำๆกันบน instance ทุกๆอีเวนต์จะทำงานทั้งหมด เช่น ประกาศอีเวนต์ให้บวกค่าตัวแปรใดๆ ทีละ 1 ใน 1 วินาที ถ้าหากเราประกาศอีเวนต์ซ้ำกัน 2 อีเวนต์ (ชื่ออีเวนต์เดียวกัน) จะทำให้ 1 วินาทีนั้นมีการทำงานอีเวนต์เท่ากับ 2 ครั้งพร้อมๆกัน ซึ่งการทำงานแบบนี้ถือว่าเป็นแบบเกิดขึ้นหลายอีเวนต์ พร้อมๆกัน

แต่สำหรับกรณีที่มีการประกาศอีเวนต์บน Timeline ให้กับ instance กรณีที่มีการประกาศอีเวนต์ซ้ำๆกันหลายๆอีเวนต์ ผลที่เกิดขึ้นก็คือคำสั่งอีเวนต์ที่ถูกประกาศเอาไว้ก่อนหน้าจะถูกทับด้วยอีเวนต์ใหม่แทน หรือพูดง่ายๆก็คือ จะทำงานโดยอีอีเวนต์ใหม่สุดเป็นหลัก ฉะนั้นจะเป็นรูปแบบการทำงานเพียงอีเวนต์เดียวเท่านั้น ซึ่งต่างจากกรณีแรก

3. การเรียกใช้งานแบบ function

การประกาศอีเวนต์บนตัว instance ไม่สามารถเรียกใช้งานในรูปแบบ function ได้ แต่การประกาศอีเวนต์บน Timeline สามารถทำได้ เนื่องจากมีรูปแบบเดียวกับการกำหนด function ให้กับ instance

- ข้อแตกต่างระหว่าง Button และ MovieClip ในกรณีใช้งานเป็นปุ่มโดยมีการประกาศอีเวนต์บนตัว Instance

ความแตกต่างระหว่างการประกาศอีเวนต์บนตัว Instance ของ Button และ MovieClip ก็คือ ธรรมชาติของ Button นั้นกรณีที่การระบุพาธ this หรือ _parent ภายใน event function ให้จำไว้เสมอว่า Button นั้นจะมีการอ้างอิงพาธประเภท _parent ให้เองโดยอัตโนมัติจำนวน 1 ครั้ง ฉะนั้นกรณีที่มิ Button ชื่อว่า my_btn ถูกวางอยู่ภายใต้โมฟวี่คลิปชื่อว่า my_mc ดังนี้

Code:

my_mc

- my_btn

เมื่อทำการประกาศอีเวนต์บนตัว Instance ของ my_btn ว่า

Code:

```
on(rollOver) {
    trace(this); //แสดงผลเป็น my_mc
    trace(_parent); //แสดงผลเป็น _level0
}
```

ซึ่งจะเห็นว่า แทนที่ this จะแสดงผล my_btn เอง แต่กลับแสดง my_mc แทน เนื่องจากว่ามีการอ้างอิงพาธแบบ _parent โดยอัตโนมัติหนึ่งครั้ง (แต่การระบุ _parent.this ไม่สามารถทำได้เองโดยผู้เขียนโปรแกรม) ส่วนการ trace(_parent) ก็จะเหมือนการระบุแบบ this._parent._parent ตามปกติในโมฟวี่คลิป ฉะนั้นจะได้ผลลัพธ์เท่ากับ _level0 ซึ่งเป็นระดับสูงสุดซึ่งก็คือ Timeline หลักของ swf ขึ้นนั้นแล้ว

สำหรับกรณีที่ประกาศไค้ดบนตัว MovieClip จะได้ผลลัพธ์ดังนี้

Code:

```
on(rollOver) {
    trace(this); //แสดงผลเป็น my_btn
    trace(_parent); //แสดงผลเป็น my_mc
}
```

- ปัญหา Events บัง Events

สำหรับข้อนี้เป็นปัญหาที่สำคัญที่หลายๆคนมักจะพบเจอแต่ไม่ทราบว่าจะเกิดอะไรขึ้น จึงทำให้ instance ดังกล่าวไม่สามารถทำงานได้ถึงแม้ว่าจะประกาศอีเวนต์ไว้เรียบร้อยแล้วก็ตาม ตัวอย่างเช่น มี mc2 อยู่ใน mc1 ดังนี้

mc1

- mc2

กรณีที่เราระบุอีเวนต์ให้กับ mc1 และ mc2 ทั้งคู่ ดังนี้

Code:

```
mc1.onPress = function() {
    trace(this); //output _level0.mc1;
    this._alpha = 50; //ปรับค่าความโปร่งใสเท่ากับ 50%
}
mc1.mc2.onPress = function() {
    trace(this); //output _level0.mc1.mc2;
    this._alpha = 50; //ปรับค่าความโปร่งใสเท่ากับ 50%
}
```

โดยจากโค้ดดังกล่าวนี้ เมื่อทำการคลิกตัว mc2 แล้ว หน้าต่าง output ควรจะแสดง path ตัวมันเอง แต่ในความเป็นจริงนั้นคือ เมื่อเราทำการกด mc2 หน้าต่าง output จะไม่แสดงผลว่าเกิดการคลิก mc2 แต่จะมองว่า mc2 เป็นเพียงแค่ส่วนหนึ่งของ mc1 เท่านั้น ฉะนั้นเมื่อคลิก mc2 ก็จะแสดงอีเวนต์ของ mc1 แทน ซึ่งจะได้ผลลัพธ์ก็คือ _level0.mc1 แทน

ด้านล่างคือตัวอย่างอีเวนต์ของ mc1 ซึ่งยังการทำงานของ อีเวนต์ตัวที่ 2 หรือ mc2 ซึ่งอยู่ภายใน mc1 หรือบนไทม์ไลน์ของ mc1 นั่นเอง

- วิธีแก้ปัญหามีอีเวนต์บังอีเวนต์

วิธีแก้สำหรับปัญหานี้ก็คือ การนำอีเวนต์ไปประกาศที่ระดับชั้นเดียวกัน โดยทำการสร้างพื้นที่สำหรับมูฟวี่คลิประดับชั้นนอกโดยนำมาวางไว้ที่ระดับเดียวกับมูฟวี่คลิประดับชั้น เพื่อไม่ให้เกิดการบังเกิดขึ้น ส่วนการอ้างอิง path ก็ให้เปลี่ยนไปใช้การอ้างอิงแบบ absolute (_root) หรือ relative (this._parent) เข้ามาช่วยในการอ้างอิงแทน

Code:

```
mc1.area_mc.onPress = function() {
    trace(this); //output _level0.mc1.area_mc;
    trace(this._parent); //output _level0.mc1;
    this._parent._alpha = 50; //ปรับค่าความโปร่งใสเท่ากับ 50%
}
mc1.mc2.onPress = function() {
    trace(this); //output _level0.mc1.mc2;
    this._alpha = 50; //ปรับค่าความโปร่งใสเท่ากับ 50%
}
```

จะเห็นว่าผลลัพธ์เหมือนกันเมื่อคลิกที่ mc1 แต่จะต่างกันว่า mc2 ก็สามารถคลิกกดได้ด้วย เนื่องจากไม่เกิดปัญหาอีเวนต์บังอีเวนต์เพราะเราได้กำหนดอีเวนต์ให้กับวัตถุที่อยู่ระดับชั้นเดียวกันนั่นเอง

- การกำหนดอีเวนต์ด้วย Listeners

ในเวอร์ชันหลังๆนั้น จะมีรูปแบบการกำหนดอีเวนต์ด้วย listener เพิ่มขึ้นมา และจะมีการใช้รูปแบบนี้เป็นวิธีการหลักใน AS3 อีกด้วยเพียงแต่จะต่างออกไปตรงชื่อคำสั่งนิดหน่อย สำหรับการทำความเข้าใจในเรื่องของ listener นั้นสามารถทำความเข้าใจได้ไม่ยาก โดยให้นึกถึง listener เปรียบเสมือนตัวแทนสำหรับการรับฟังขึ้นมา ซึ่งมีหน้าที่เอาไว้ฟังอีเวนต์ต่างๆ และวัตถุใดๆที่ได้รับ listener นั้นไว้ ก็จะทำงานตามที่ listener ได้รับคำสั่งหรืออีเวนต์ที่เกิดขึ้นหรือผมจะทำให้เข้าใจมากยิ่งขึ้น โดยจำลองว่า listener เปรียบเสมือน headphone อันนี้ ซึ่งถ้าใครฟังแล้วจะเหมือนโดนสะกดจิตให้ทำตามคำสั่งต่างๆตามที่ได้ฟังจาก headphone ส่วนวัตถุที่จะรับคำสั่ง ก็ลองนึกถึง Mouse เปรียบเสมือนหนูตัวหนึ่งที่เอาหูฟังนั้นมาสวมหัวแล้วทำตาม ดังภาพเป็นต้น

Code:

```
my_listener = new Object(); //สร้าง listener ขึ้นมาเองด้วยชื่ออะไรก็ได้
my_listener.onEventListener = function() { //statement
}
Object.addListener(my_listener); //ประกาศอีเวนต์ listener ให้กับ Object หรือ Class ที่ต้องการ
```

สำหรับ Class ที่สามารถกำหนด Listeners ได้จะมี Mouse, Keyboard, Stage, MovieClipLoader (สำหรับการโหลดไฟล์), Selection นอกจากนี้เรายังสามารถพบการประกาศรูปแบบนี้ได้ component อีกด้วย เพียงแต่จะใช้คำสั่ง addEventListener แทน (ใช้คำสั่งนี้ใน AS3 เช่นกัน) โดยการกำหนดอีเวนต์รูปแบบนี้ถือว่าการประกาศ Events โดยปกติอยู่แล้ว ฉะนั้นจะถือว่าเป็น Events ใดๆ ไม่ได้เรียกว่า Listeners

ตัวอย่างการประกาศ event listener ให้กับ Mouse ให้ประกาศโค้ดดังกล่าวที่ Timeline แล้ว กด Ctrl+Enter เพื่อทดสอบ

Code:

```
my_ls = {}; //สร้าง listener ขึ้นมาจาก Object โดยเขียนในรูปแบบย่อคือ {} ซึ่งแทน Object
my_ls.onMouseDown = function() { //ทำเมื่อกดคลิกเมาส์
    trace("onMouseDown");
}
my_ls.onMouseMove = function() { //ทำเมื่อปล่อยเมาส์
    trace("onMouseMove");
}
my_ls.onMouseUp = function() { //ทำเมื่อปล่อยเมาส์
    trace("onMouseUp");
}
my_ls.onMouseWheel = function() { //ทำเมื่อหมุน mouse wheel หรือล้อหมุน
    trace("onMouseWheel");
}
Mouse.addListener(my_ls); //ใส่ listener ให้กับ Mouse
```

2.1.6.6 ตัวแปร (Variable) และ ฟังก์ชัน (Function)

ตัวแปร และฟังก์ชัน ซึ่งถือว่าเป็นพื้นฐานการเขียนโปรแกรมอย่างหนึ่ง ซึ่งทุกภาษาจะไม่แตกต่างกันมากนัก แต่อาจจะมีข้อกำหนดเงื่อนไขพิเศษแตกต่างกันไป

ตัวแปร (Variable)

สำหรับตัวแปรนั้น หน้าที่ของมันโดยพื้นฐานก็คือการเก็บค่า (value) ต่างๆ ซึ่งบางครั้งอาจจะนำมาใช้ในการคำนวณค่าต่างๆ หรือ การเก็บค่าใดๆ เพื่อไว้สำหรับอ้างอิงหรือตรวจสอบเมื่อต้องการ

ตัวอย่างของการประกาศตัวแปร เป็นดังนี้

Code:

```
var x;
```

ถ้าแบบนี้เป็นการประกาศตัวแปรเฉยๆ แต่จะนำไปใช้ทีหลัง สรุปก็คือเป็นตัวแปรที่ยังไม่มีค่าอะไรทั้งสิ้น

Code:

```
var x=10;
```

สำหรับกรณีนี้เป็นการประกาศตัวแปรพร้อมกำหนดค่าด้วยเครื่องหมายเท่ากับ หรืออาจจะเขียนแบบไม่ใช่ var ได้ว่า

Code:

```
x=10;
```

- สิ่งที่เราต้องรู้เกี่ยวกับตัวแปรทั่วไปมีดังต่อไปนี้

1. การตั้งชื่อตัวแปร (Naming Variables)

- ข้อจำกัดและคำสงวน คำที่ห้ามใช้ (Reserved Word)

- การใช้ตัวเล็ก ตัวใหญ่ (Case Sensitive)

1.1 การตั้งชื่อตัวแปร (Naming Variable)

สำหรับการตั้งชื่อตัวแปรนั้นจะมีเงื่อนไข กฎเกณฑ์ในการตั้งชื่อ ซึ่งการใช้สัญลักษณ์บางอย่างนั้นไม่สามารถใช้ได้ในการตั้งชื่อ ซึ่งรายละเอียดต่างๆพอสรุปได้ดังนี้

- ห้ามขึ้นต้นชื่อตัวแปรด้วยตัวเลข เช่น "1a" นอกจากนี้สามารถใช้ "_" หรือ \$ ขึ้นต้นได้ (php ก็ใช้ \$ สำหรับประกาศว่าค่านั้นคือตัวแปร) แต่สำหรับในแพลชนั้น จะนิยมขึ้นด้วยตัวอักษร

- ห้ามเว้นวรรคตัวแปรหรือใช้เครื่องหมาย สัญลักษณ์ต่างๆกัน เช่น -, ' ถ้าจะใช้ควรวี_เป็นตัวกัน เช่น my_variable หรือใช้การแบ่งตัวเล็กตัวใหญ่ เช่น myVariable ที่จะใช้ V ซึ่งเป็นตัวแรกของคำเป็นตัวคั่นระหว่างคำแต่ละคำเพื่อให้อ่านได้ง่าย

1.2 คำสงวน (Reserved Word)

นอกจากนี้ การตั้งชื่อตัวแปร ไม่ควรตั้งให้ตรงกับคำที่เป็นคำสงวน ซึ่งห้ามไม่ให้ใช้ เพราะเป็นคำที่เป็นมาตรฐานทั่วไป หรือว่ามีกาใช้สำหรับโปรแกรมแล้ว ตัวอย่างเช่น เราไม่สามารถตั้งตัวแปรว่า function, case, var, if และอื่นๆ เนื่องจากคำเหล่านี้เป็นคำสั่งพื้นฐานที่ใช้ในการเขียนโปรแกรมนั่นเอง

สำหรับตัวอย่างคำสงวนมีดังต่อไปนี้

Flash Player 2 - on

Flash Player 3 - ifFrameLoaded

Flash Player 4 - add, and, break, case, continue, default, do, else, eq, ge, gt, if, le, lt, ne, not, or, switch, tellTarget, while

Flash Player 5 - delete, for, function, in, new, onClipEvent, return, this, typeof, var, void, with

Flash Player 6 - instanceof

ActionScript 2.0 - Flash Player 6 - class, dynamic, extends, get, implements, import, interface, intrinsic, private, public, set, static

Flash Player 7 - catch, finally, throw, try

สำหรับเวอร์ชันหลังๆ ขึ้นอยู่กับคำสั่งต่างๆที่ถูกเพิ่มขึ้นมาในภายหลัง

1.3 การใช้ตัวเล็ก ตัวใหญ่ (Case Sensitive)

ใน flash เวอร์ชัน 5 นั้น ถ้าเราตั้งชื่อตัวแปรว่า myvariable หรือ myVariable หรือ myVRiable แต่ละตัวแปรนั้น โปรแกรมจะมองว่าเป็นตัวแปรเดียวกันทั้งหมด ซึ่งไม่สนใจตัวเล็กตัวใหญ่ แต่สำหรับ flash mx หรือเวอร์ชัน 6 เป็นต้นมานั้น จะเริ่มสนใจตัวอักษรเล็กใหญ่ ฉะนั้นถ้าสะกดตัวเล็กใหญ่ไม่เหมือนกัน จะถือว่าเป็นคนละตัวแปร จึงควรระวัง ณ จุดนี้ บางครั้งเราสะกดอีกอย่าง แต่เรียกใช้อีกอย่าง ก็จะทำให้อ้างอิงค่าตัวแปรนั้นไม่ได้

2. ประเภทของตัวแปร (Data Type)

ในตอนที่ยังเป็น flash 5 นั้น การกำหนดประเภทของตัวแปรยังไม่มี นั่นคือตัวแปรนั้นสามารถเก็บค่าอะไรก็ได้ อาทิเช่น

Code:

```
x=5;
```

หรือCode:

```
x="hello";
```

แต่หลังจากเวอร์ชัน MX เวอร์ชันถัดมา ได้มีการเพิ่มประเภทของตัวแปรขึ้นมา โดยที่เราสามารถกำหนดได้ว่าตัวแปรนั้นต้องการเก็บข้อมูลประเภทไหน

ตัวอย่างเช่น

Code:

```
var a:Number;
```

```
a = 5;
```

ซึ่งเป็นการกำหนดให้ตัวแปรเก็บค่าเป็นตัวเลข (Number) เท่านั้น

1. สำหรับการกำหนดค่าตัวแปรจะต้องมีการประกาศ var นำหน้าเสมอไม่งั้นโปรแกรมจะเตือน error ว่า syntax error
2. กรณีที่มีการกำหนดค่าซึ่งมีประเภทของข้อมูลไม่ตรงกับประเภทของตัวแปร โปรแกรมจะเตือน error เช่นกัน เช่น

Code:

```
var a:Number = "this is String";
```

โปรแกรมจะเตือน error ว่า Type mismatch in assignment statement:found String where Number is required. ซึ่งเตือนว่าประเภทข้อมูลที่กำหนดไม่ถูกต้อง และค่าที่ต้องการคือตัวเลข

- ตัวแปรประเภทต่างๆ

ในที่นี้จะขอกล่าวประเภทหรือชนิดของตัวแปรพื้นฐาน โดยทั่วไปก่อน ซึ่งเราจะได้เห็นบ่อยในภาษาอื่นๆด้วย ซึ่งจะมีดังต่อไปนี้

1. Number เป็นชนิดตัวแปรประเภทตัวเลข ซึ่งเราสามารถนำมาใช้ในการคำนวณได้ตามเครื่องหมายตามคณิตศาสตร์คือ บวก (+) ลบ (-) คูณ (*)หาร (/) และ modulus หรือย่อๆว่า mod (%) ซึ่งเป็นการหารเอาเฉพาะค่าเศษ เท่านั้น

นอกจากนี้ ตัวเลขยังแบ่งเป็น อีก 2 ค่าคือ

1.1 int ย่อจาก integer เก็บค่าเฉพาะตัวเลขจำนวนเต็ม

1.2 float ย่อมาจาก floating-point เป็นชนิดตัวแปรที่เก็บตัวเลขประเภทมีค่าทศนิยม

2. String เป็นชนิดตัวแปรประเภทข้อความหรือชุดอักษร โดยจะใช้ ' หรือ " เป็นตัวครอบ เช่น

Code:

```
my_str = "my name is bubbleball";
```

นอกจากนี้เรายังสามารถนำข้อความมาต่อกันด้วยการใช้เครื่องหมาย +

Code:

```
my_str = "my name is " + "bubbleball";
```

สำหรับกรณีที่เราต้องการใช้เครื่องหมาย ' หรือ " เราสามารถทำได้ 2 วิธีคือ

1. การใช้ ' ครอบ " หรือ " ครอบ ' เช่น my_str = "my name is 'bubbleball'"; หรือ my_str = 'my name is "bubbleball"';

2. การใช้เครื่องหมาย \ คั่นหน้าสัญลักษณ์นั้น เช่น my_str = "my name is \"bubbleball\""; ซึ่งจะมีค่าเท่ากับ my name is "bubbleball"

นอกจากนี้ กรณีที่นำ Number มาบวกกับ String จะทำให้ Number นั้นถูกแปลงเป็น String โดยอัตโนมัติ เช่น

```
my_str = "my number is " + 5 จะมีค่าเท่ากับ my_str = "my number is 5";
```

3. Boolean สำหรับชนิดตัวแปรนี้ จะเกี่ยวข้องกับตรรกะ จริง หรือเท็จ โดยค่าที่เก็บจะมีแค่ true หรือ false เท่านั้น ซึ่งส่วนใหญ่จะถูกใช้สำหรับการเปรียบเทียบ หรือการตรวจสอบเงื่อนไขต่างๆ เช่น

Code:

```
rainNow = true;

if (rainNow) {

    trace("ฝนกำลังตก");

}
```

```
else {
    trace("ฝนไม่ได้ตก");
}
```

สำหรับการสลับค่าตัวแปรชนิดนี้จาก true เป็น false มักจะใช้เครื่องหมาย ! ในการนำหน้าตัวแปร ซึ่งจะทำให้กลายเป็นค่าตรงกันข้าม บางครั้ง ค่า false นั้นสามารถแทนได้ด้วยตัวเลข 0 ส่วนค่า true นั้นจะแทนด้วยค่าอื่นๆนอกจาก 0 รวมถึงค่า String ด้วยเช่นกัน

4. undefined เป็นชนิดตัวแปร ที่ว่าตัวแปรนั้นไม่มีการประกาศอยู่ หรือแม้แต่ประกาศแล้ว ก็ยังไม่มีค่าที่กำหนดค่าใดๆ ลงไปทั้งสิ้น จึงเท่ากับมีค่าเป็น undefined ซึ่งสามารถใช้ตรวจสอบในเงื่อนไขได้ว่า ถ้ามีค่าตัวแปรนี้อยู่ควรจะทำอะไร และถ้าไม่มีควรจะทำอะไร

5. null เป็นการเก็บค่าเปล่าๆที่ไม่ได้ถูกนำมาใช้อะไร แทนที่จะให้ตัวแปรนั้นเป็นค่า undefined ง่ายๆ อย่างเช่น

Code:

```
my_variable = null;
```

สำหรับการตรวจสอบชนิดของตัวแปรจะใช้ฟังก์ชัน typeof ในการตรวจสอบ ซึ่งจะช่วยให้เรารู้ได้ว่า ประเภทตัวแปรถูกต้องหรือไม่ ซึ่งบางครั้งจะช่วยให้เรารู้ได้ว่า เงื่อนไขที่เราได้ตรวจสอบถูกต้องตามชนิดของตัวแปรนั้นๆหรือไม่ เพราะบางครั้งเราอาจจะคิดว่าตัวแปรนั้นเป็น Number แต่อันที่จริงอาจจะเป็นตัวแปรชนิด String ก็ได้ ซึ่งอาจจะเกิดขึ้นได้ ในกรณีที่มีการโหลดค่าต่างๆมาจากภายนอก flash เป็นต้น

- ตัวแปรประเภทเดียวกับ class

นอกจากชนิดตัวแปร แบบพื้นฐานที่กล่าวไปแล้ว เรายังสามารถกำหนดชนิดของตัวแปรตามค่า class นั้นได้อีก อย่างเช่น

```
class sound : var mySound:Sound = new Sound();

class array : var myArray:Array = new Array();
```

เป็นต้น ซึ่ง เราจะกำหนดชนิดของตัวแปรตามประเภท class ที่เราสร้างขึ้น

- การตั้งชื่อตัวแปร ตามชนิดของ class เพื่อแสดง code hinting

ในหน้าต่าง ActionScript นั้น บางครั้งเมื่อเราพิมพ์อยู่ จะปรากฏคำสั่งขึ้นมาให้เราเลือกใช้ โดยอัตโนมัติ ซึ่งเป็นการใช้งาน code hinting หรือบอกไว้ว่ามีคำสั่งใดๆ ให้ใช้บ้าง ซึ่งเงื่อนไขที่จะทำให้ code hinting ปรากฏก็คือ

1. กำหนดชนิดของตัวแปรตามหัวข้อด้านบน
2. ตั้งชื่อท้าย หรือ suffix เพื่อบ่งบอกให้ flash ทราบว่าตัวแปรนั้นเป็นตัวแปรประเภทไหน โดยหลักการคือใช้ _ แล้วลงท้ายด้วยอักษรย่อแทนชนิดตัวแปรนั้นๆ อย่างเช่น

MovieClip จะมีการตั้งชื่อตัวแปร ลงท้ายว่า _mc

ซึ่งทุกครั้งที่เราพิมพ์ _mc แล้วจะปรากฏ code hinting ขึ้นมาโดยอัตโนมัติ ดังนั้นโดยส่วนใหญ่เวลาเราตั้งชื่อ instance name ให้กับรูปวีคลิป หรือ ปุ่ม เรามักจะตั้งลงท้ายด้วย _mc เช่น my_mc หรือ _btn เช่น my_btn เป็นต้น

ตารางที่ 2.1 ตัวแปรชนิดต่างๆ (สามารถเปิด help แล้วค้นหาด้วยคำว่า code hint)

<u>Object type</u>	<u>Variable suffix</u>
Array	_array
Button	_btn
Camera	_cam
Color	_color
ContextMenu	_cm

<u>Object type</u>	<u>Variable suffix</u>
ContextMenuItem	_cmi
Date	_date
Error	_err
LoadVars	_lv
LocalConnection	_lc
Microphone	_mic
MovieClip	_mc
MovieClipLoader	_mcl
PrintJob	_pj
NetConnection	_nc
NetStream	_ns
SharedObject	_so
Sound	_sound
String	_str
TextField	_txt
TextFormat	_fmt
Video	_video
XML	_xml
XMLNode	_xmlnode

- การเปลี่ยนชนิดของตัวแปร

สำหรับการเปลี่ยนชนิดของตัวแปร สามารถเปลี่ยนได้ง่ายๆด้วยการนำฟังก์ชันตามชนิดของตัวแปรนั้นครอบตัวแปรหรือค่าที่เราต้องการเช่น

- การแปลงจาก ตัวเลข เป็น ข้อความ

Code:

```
my_number = 5;

my_str = Number(my_number);

trace(typeof(my_str)); //จะได้ string
```

- การแปลงจาก ข้อความ เป็น ตัวเลข

Code:

```
my_number = Number("5");

trace(typeof(my_number)); //จะได้ number
```

สำหรับการแปลงข้อมูล นั้นไม่ว่าเราจะสามารถแปลงจากประเภทหนึ่งไปยังอีกประเภทหนึ่งได้เสมอ ขึ้นอยู่กับเงื่อนไขที่ flash กำหนดไว้ได้ว่าสามารถแปลงจากคลาสไหนเป็นคลาสไหนได้บ้าง

ในตอนนี้ เราได้เรียนรู้การใช้งานตัวแปร เบื้องต้นเรียบร้อยแล้ว หลังจากนั้นเราจะทำการเรียนรู้การใช้งาน ฟังก์ชันต่อไป ซึ่งจะมีส่วนที่เกี่ยวข้องกับตัวแปรเพิ่มเติมอีกด้วย

- ฟังก์ชัน (Function)

ในส่วนนี้เราจะได้เรียนรู้เกี่ยวกับการใช้งานฟังก์ชัน ซึ่งถือว่าเป็นพื้นฐานที่สำคัญอย่างมากในการเขียนโปรแกรมทุกภาษา โดยฟังก์ชันนั้น เป็นการรวมกลุ่มคำสั่งหลายๆคำสั่งเข้าไว้ด้วยกัน

- ประโยชน์ของฟังก์ชัน

ฟังก์ชันนั้นมีประโยชน์ในการแบ่งคำสั่งต่างๆให้ทำงานเฉพาะอย่าง แบ่งเป็นหน้าที่ต่างๆที่ต้องการ ซึ่งการเขียนคำสั่งไว้ในฟังก์ชันนั้นจะช่วยให้สามารถนำคำสั่งนั้นไปใช้ต่อได้ง่ายขึ้น รวมถึง การใช้งานซ้ำๆ ภายในโปรแกรมได้ไม่จำกัดจำนวนครั้ง นอกจากนี้จะช่วยให้ผู้เขียนโปรแกรมสามารถอ่านโค้ดได้ง่ายมากขึ้น เนื่องจากโค้ดจะดูเป็นระเบียบเรียบร้อย

- รูปแบบของฟังก์ชัน และการเรียกใช้

รูปแบบของฟังก์ชัน จะสามารถเขียนได้ดังต่อไปนี้

1. ฟังก์ชันที่ไม่มีการส่งค่าตัวแปรพารามิเตอร์ (parameter)

Code:

```
function name() {  
    //statement  
}
```

ซึ่งจะประกาศด้วยคำสั่งว่า function หลังจากนั้นตามด้วยชื่อฟังก์ชันที่ต้องการ (name) หลังจากนั้นจะทำการเริ่มต้นการเขียนคำสั่งโดยการเปิดด้วย { และจบฟังก์ชันด้วยการเขียน } ปิดท้าย ซึ่งการเปิดปิดด้วย { } แบบนี้เราจะเรียกว่า block หรือ body ซึ่งจะต้องเปิดปิดให้ครบ เสมอ ไม่งั้นโปรแกรมจะเตือน error

สำหรับวิธีการเรียกใช้ ทำได้ง่ายๆด้วยการระบุ name() นั่นก็คือระบุชื่อฟังก์ชันแล้วตามด้วยเครื่องหมาย () นั่นเอง

ตัวอย่างเช่น

Code:

```
function helloWorld() {  
  
    trace("hello world");  
  
}  
  
helloWorld(); //เรียกใช้ ซึ่งหน้าต่าง output จะแสดง hellow world
```

จากตัวอย่างเป็นวิธีการเขียนฟังก์ชันและวิธีการเรียกใช้ฟังก์ชันแบบง่ายๆ ซึ่งจะเห็นว่ารูปแบบการประกาศและเรียกใช้ที่ไม่ยาก สำหรับตอนนี้เราเพียงแค่รู้รูปแบบการประกาศ และวิธีการเรียกใช้แบบง่ายๆพอเข้าใจก่อน ซึ่งถึงเวลาการใช้งานจริง เราจะเริ่มคุ้นเคยกับรูปแบบฟังก์ชันที่ซ้ำซ้อนมากขึ้นเอง

2. ฟังก์ชันที่มีการส่งตัวแปรพารามิเตอร์ (parameter)

ตัวแปรพารามิเตอร์ นั้นมีหน้าที่ในการรับค่าใดๆเข้าสู่ฟังก์ชันเพื่อนำไปใช้ต่อภายในฟังก์ชันนั้นๆ ซึ่งจะช่วยให้ฟังก์ชันสามารถทำงานได้หลายหลายมากยิ่งขึ้น โดยรูปแบบการกำหนดค่าพารามิเตอร์ให้กับฟังก์ชันเป็นดังต่อไปนี้

Code:

```
function name(parameter1,parameter2,...,parameterN) {  
  
    //statement  
  
}
```

นอกจากนี้เรายังสามารถตรวจสอบค่าที่ส่งผ่านพารามิเตอร์เข้ามาทั้งหมด หรือค่า argument จากพรอพเพอร์ตี้ของฟังก์ชันที่ชื่อว่า arguments อีกด้วย โดยค่า arguments นี้จะเป็นอาเรย์ที่เก็บค่าของพารามิเตอร์แต่ละตัว (หากไม่เข้าใจว่าอาเรย์คืออะไรให้ศึกษาได้ในหัวข้อการใช้งาน array) โดยวิธีการใช้มีดังนี้

Code:

```
function testArguments(param1,param2,param3) {

    trace(arguments);

    trace(arguments.length);

}

testArguments(1,2,3); //จะแสดงผล 1,2,3 และ 3
```

ซึ่งค่าที่จะแสดงผลออกมาก็คือ 1,2,3 ซึ่งเป็นอาร์เรย์ที่เก็บค่า 1,2,3 เอาไว้ในแต่ละ index และ ค่า 3 ที่แสดงนั้นก็ได้จากพรอปเพอร์ตี้ .length ของอาร์เรย์ซึ่งแสดงจำนวนที่อาร์เรย์เก็บค่าเอาไว้ทั้งหมดนั่นเอง

สำหรับ arguments นั้นจะมีประโยชน์ในการตรวจสอบว่า ค่าพารามิเตอร์ที่ถูกส่งเข้ามาทั้งหมดมีจำนวนกี่ตัว และเลือกที่จะทำงานได้ตามจำนวนค่าพารามิเตอร์ที่ถูกส่งเข้ามาได้ เนื่องจากฟังก์ชันใน flash มีความยืดหยุ่น โดยที่เราสามารถกำหนดพารามิเตอร์จำนวนหนึ่ง แล้วส่งค่าเข้ามาในฟังก์ชัน ตามจำนวนพารามิเตอร์ที่กำหนดเอาไว้หรือไม่ก็ได้ โดยอาจจะน้อยกว่าหรือมากกว่าที่ยอมได้

ข้อควรรู้ในการเรียกใช้งานฟังก์ชันอย่างหนึ่งก็คือ เราสามารถเรียกใช้งานฟังก์ชันนั้นก่อนที่จะทำการประกาศฟังก์ชันก็ได้ เนื่องจากว่า โปรแกรมจะทำการคอมไพล์คำสั่งภายในตำแหน่งที่เขียน actionscript จนจบก่อน ซึ่งจะทำให้สามารถเรียกใช้ฟังก์ชันก่อนประกาศค่าได้เลย

ตัวอย่างเช่น

Code:

```
test(); //output แสดง test

function test() {

    trace("test");}
```


- การส่งค่าคืนของฟังก์ชัน (Function Return Value)

จากตัวอย่างของฟังก์ชันที่ผ่านมา จะเห็นว่า เมื่อมีการกำหนดค่าตัวแปรใดๆ หลังจากจบการทำงาน จะใช้วิธีกำหนดตัวแปรขึ้นมาโดยตรง ซึ่งเป็นการสร้างตัวแปรขึ้นมาใหม่ แล้วจึงมีการเรียกใช้ตัวแปรดังกล่าวภายหลัง (จะทำความเข้าใจมากขึ้นในหัวข้อขอบเขตฟังก์ชันด้านล่าง)

แต่นอกจากวิธีดังกล่าวแล้ว เรายังสามารถให้ฟังก์ชันนั้นคืนค่าผลลัพธ์ที่ต้องการและนำไปใช้ได้ ซึ่งเราเรียกวิธีนี้ว่าการส่งค่าคืนของฟังก์ชัน โดยวิธีดังกล่าวเราจะใช้คำสั่งชื่อว่า `return`

Code:

```
function helloWorld() {
    return "Hello World";
}

trace(helloWorld); //output แสดง Hello World
```

หรือเราจะนำตัวแปรมารับค่าจากฟังก์ชันก็ได้ เช่น

Code:

```
function helloWorld() {
    return "Hello World";
}

myWord = helloWorld();

trace(myWord); //output แสดง Hello World
```

สำหรับฟังก์ชันที่มีการส่งค่าพารามิเตอร์ ก็มีวิธีการใช้ในรูปแบบเดียวกัน

การใช้งานคำสั่ง `return` จะเป็นการจบการทำงานของฟังก์ชันนั้นในทันที ซึ่งจะทำให้คำสั่งหลังจากนั้นไม่ถูกทำงาน บางครั้งเราอาจจะ `return ""`; เพื่อจบการทำงานของฟังก์ชัน เมื่อค่าพารามิเตอร์ไม่ตรงเงื่อนไขที่ต้องการก็ย่อมได้

การกำหนดชนิดของพารามิเตอร์และฟังก์ชัน

เนื่องจากตัวแปรนั้นสามารถกำหนดชนิดของตัวแปรได้ สำหรับพารามิเตอร์และฟังก์ชัน ก็สามารถระบุชนิดได้เช่นเดียวกัน ตัวอย่างเช่น

Code:

```
function testDataType(param1:String,param2:Number):Void {

    trace(param1 + ":" + param2);

}testDataType("My number is ",9); //output แสดงผล My number is 9
```

โดย Void หมายความว่า ไม่มีการ return ค่าใดๆให้กับฟังก์ชัน แต่ถ้ามีการ return ข้อมูลชนิดใด ก็ให้ระบุชนิดของข้อมูลเช่นนั้น

กรณีที่มีการส่งชนิดข้อมูลของพารามิเตอร์ไม่ตรงกับชนิดที่ถูกกำหนดไว้ โปรแกรมจะเตือน Type mismatch (แต่ไม่บังคับว่าต้องส่งค่าพารามิเตอร์ตัวนั้นหรือไม่)

- ฟังก์ชันแบบเรียกตัวเอง (Recursive Function)

รีเคอร์ซีฟฟังก์ชันนั้นเป็นฟังก์ชันที่เรียกใช้งานตัวเอง ซึ่งปกติ เรามักจะเรียกใช้งานฟังก์ชันอื่นๆ ภายในฟังก์ชันใดๆมากกว่าอย่างเช่น

Code:

```
function first() {

    trace("first");

}

function second() {

    first();

    trace("second"); } second(); //จะได้ output first และ second คนละบรรทัด
```

แต่เราก็สามารถที่จะให้ฟังก์ชันสามารถเรียกใช้งานตัวเองได้ โดยการอ้างถึงชื่อฟังก์ชันนั้นๆเอง แต่กรณีแบบนี้จะทำให้เกิดปัญหาอย่างหนึ่งคือ การทำงานของฟังก์ชันนั้นจะทำงานแบบไม่รู้จบ ฉะนั้นข้อสำคัญในการใช้ recursive function ก็คือ ฟังก์ชันนั้นจะต้องมีเงื่อนไขในการจบและยกเลิกการเรียกตัวเองเสมอ

ตัวอย่าง recursive function ที่มีปัญหา

Code:

```
function nonStop() {  
    trace("nonStop");  
    nonStop();  
}
```

```
nonStop();
```

ซึ่งจะทำให้ flash แสดง output "nonStop" จำนวน 256 ครั้งแล้วเตือน error ว่า

อ้างถึง

256 levels of recursion were exceeded in one action list.

This is probably an infinite loop.

Further execution of actions has been disabled in this movie.

สำหรับวิธีแก้เราจะใช้วิธีการตรวจสอบเงื่อนไขเพื่อออกจาก function อาทิเช่น

Code:

```
num=0;
```

```
function count() {

    num++;

    trace(num);

    if (num==100) {

        return "";

    }

    else {

        count();

    }

}

count();
```

ซึ่งเมื่อกดทดสอบ หน้าต่าง output จะแสดงเลข 1 ถึง 100 เนื่องจากว่าเราเขียนเงื่อนไขเช็คเอาไว้ว่าถ้าค่า num ถึง 100 ก็ให้ return "" เพื่อทำการออกจากฟังก์ชันนั่นเอง

การทำงานของ recursive function จะเป็นการทำงานแบบจบในรอบเดียว ซึ่งก็คือจะไม่มีเรื่องของความต่างของเวลาในการเรียกใช้ฟังก์ชันแต่ละครั้งมาเกี่ยวข้อง หรือพูดง่าย ๆ ก็คือ ฟังก์ชันจะถูกทำงานและจบในทันที

- การใช้ recursive function แบบส่งค่า parameter

เนื่องจากเราทราบว่า function ภายใน flash นั้นสามารถที่จะส่งค่าหรือไม่ส่งก็ได้ ถึงแม้จะประกาศตัวแปรพารามิเตอร์เอาไว้ก็ตาม ฉะนั้นเราสามารถนำเงื่อนไขนี้มาสร้างฟังก์ชัน ที่รับค่าจากครั้งเก่ามาดำเนินการต่อไป หรือ สร้างฟังก์ชันที่ทำงานแตกต่างกันในแต่ละครั้งของการทำงาน หรือ แตกต่างกันระหว่างครั้งแรก และครั้งถัดไปได้

ตัวอย่างฟังก์ชันที่มีการส่งค่าจากครั้งเก่ามาใช้ในครั้งถัดๆไป เช่น การหาค่า factorial (เช่น factorial ของ 5 ก็จะเท่ากับ $5 \times 4 \times 3 \times 2 \times 1$)

Code:

```
function factorial(n) {
    if (n==0) {
        return 1;
    }
    else {
        return n*factorial(n-1);
    }
}

trace(factorial(2)); //output 2 (ได้จาก 2 x 1)
trace(factorial(3)); //output 6 (ได้จาก 3 x 2 x 1)
}
```

จาก ตัวอย่าง เมื่อส่งค่าเข้าไปในฟังก์ชัน ด้วยค่า n ซึ่งกรณีที่ค่า n ยังไม่เท่ากับ 0 ก็จะมีการเอาค่า n ไปคูณกับค่า $n-1$ ซึ่งจะถูกลวนเรียกตัวเองไปเรื่อยๆจนกว่าจะถึงค่า n เท่ากับ 0 จึงส่งค่า 1 กลับมาและถือเป็นการจบการเรียกฟังก์ชันซ้ำเนื่องจากไม่ได้มีการเรียกใช้ฟังก์ชัน factorial อีก

ตัวอย่างการทำงานต่างกันครั้งแรกและครั้งถัดๆไป

Code:

```
function doDifferent(num,firstTimes) {
    trace(num);
}
```

```

if (num>100) {return ""};

if (firstTimes) {

    trace("this is first times");

    num = num+2;

    doDifferent(num);

} else {

    trace("this is not first times");

    num = num+1;

    doDifferent(num);

}

} doDifferent(1,true);

```

จากตัวอย่างเป็นการสร้างฟังก์ชันบวกค่าง่ายๆ เพียงแต่มีการตรวจสอบค่าพารามิเตอร์ชื่อว่า firstTimes ว่ามีค่าหรือไม่ ซึ่งในครั้งแรกสุดเราเรียกใช้งานฟังก์ชันแบบกำหนดค่า ก็จะทำให้มีการเข้าเงื่อนไข if (firstTimes) แต่หลังจากที่มีการเรียกฟังก์ชันตัวเองซ้ำ และไม่มีการส่งค่า true ในพารามิเตอร์ที่ 2 ก็จะทำให้ฟังก์ชันนั้นทำงานโดยเข้าเงื่อนไข else ไปตลอดทุกครั้งนั่นเอง (ส่วนใหญ่ผมมักสร้างฟังก์ชันประเภทนี้สำหรับกรณีที่ต้องมีการเรียกใช้งานแบบเรียกฟังก์ชันซ้ำหลายๆ วินาทีที่กำหนด โดยฟังก์ชันจะเริ่มทำงานครั้งแรกหลังจากวินาทีที่กำหนดเช่นเดียวกัน เพียงแต่ครั้งแรกสุดผมต้องการให้ทำงานทันทีก็จะใช้วิธีกำหนดเงื่อนไขลักษณะนี้)

- ขอบเขตของฟังก์ชัน และตัวแปร (Function Scope)

เนื่องจากฟังก์ชันเป็นการเขียนคำสั่งภายใน block หรือ { } ซึ่งใช้ในการเปิดและปิดฟังก์ชัน ซึ่งจะทำให้เกิดขอบเขตขึ้นมา ถ้าให้อธิบายมากขึ้นก็คือ ขอบเขตจะเป็นตัวกำหนดการใช้งานของตัวแปรใดๆว่าอ้างอิงหรือใช้งานได้ไม่ได้นั่นเอง

โดยปกติแล้ว ในการเขียนโปรแกรมทั่วไป เรื่องของขอบเขตจะเกี่ยวกับตัวแปรเป็นหลัก โดยตัวแปรนั้นจะมีการแบ่งออกเป็น 2 แบบก็คือ

1. ตัวแปรแบบ global เป็นตัวแปรประเภทที่ใช้งานทุกที่
2. ตัวแปรแบบ local เป็นตัวแปรประเภทใช้งานเฉพาะที่

ในเบื้องต้นนี้เราจะทำความรู้จักตัวแปรทั้งสองแบบก่อน โดยยึดจากการประกาศตัวแปรภายในฟังก์ชันเป็นหลัก

1. ตัวแปรแบบ global เป็นตัวแปรประเภทที่ใช้งานทุกที่

ตัวแปรประเภทนี้ส่วนใหญ่ มักจะมีการถูกใช้งานหลายครั้ง โดยที่อาจจะประกาศค่าเอาไว้ก่อนหรือภายหลังก็ได้ โดยสามารถประกาศจากภายในฟังก์ชันก็สามารถทำได้

ตัวอย่างตัวแปรแบบ global

Code:

```
globalVariable = 5;

function globalTest() {
    globalVariable = 10;
}

trace(globalVariable); //output แสดงผล 5

globalTest();

trace(globalVariable); //output แสดงผล 10
```

จากตัวอย่าง เราประกาศตัวแปรเริ่มต้นเอาไว้ที่ 5 หลังจากเมื่อเรียกฟังก์ชัน globalTest() ซึ่งมีการกำหนดค่าใหม่ให้กับตัวแปร globalTest() จะทำให้ค่า globalVariable เปลี่ยนค่าจาก 5 กลายเป็น 10 ดังนั้นเมื่อเราทดสอบแสดงค่า globalVariable อีกครั้ง จึงได้ค่าเป็น 10 นั่นเองเท่ากับว่าตัว-

แปร globalVariable เป็นตัวแปรเดียวกับตัวแปรภายในฟังก์ชัน ซึ่งถือว่าเป็น ตัวแปรประเภท global

ตัวอย่างตัวแปรแบบ global

Code:

```
function globalTest() {
    globalVariable = 10;
}

trace(globalVariable); //output แสดงผล undefined

globalTest();

trace(globalVariable); //output แสดงผล 10
```

จากตัวอย่างแสดงให้เห็นว่า ถึงแม้ไม่ได้กำหนดตัวแปรเอาไว้ที่นอกฟังก์ชัน แต่เมื่อมีการประกาศตัวแปรภายในฟังก์ชัน ก็จะเท่ากับการประกาศตัวแปรเอาไว้ภายนอกฟังก์ชันได้เช่นกัน

2. ตัวแปรแบบ local เป็นตัวแปรประเภทใช้งานเฉพาะที่ส่วนใหญ่ตัวแปรประเภทนี้ มักจะถูกใช้แบบชั่วคราวเท่านั้น

ตัวอย่างตัวแปรแบบ local

Code:

```
globalVariable = 5;

function globalTest() {
    var globalVariable = 10;
}

trace(globalVariable); //output แสดงผล 5
```



```
globalTest();

trace(globalVariable); //output แสดงผล 5
```

จากตัวอย่าง จะเห็นว่าเราเพียงแค่ประกาศ var นำหน้าตัวแปรภายในฟังก์ชันเพิ่มขึ้นมา ซึ่งเป็นการระบุว่าตัวแปรดังกล่าวเป็นตัวแปรแบบ local ซึ่งใช้เฉพาะภายในฟังก์ชันเท่านั้น เมื่อจบการทำงานของฟังก์ชัน ก็จะถูกครีเอตออกจากเมมโมรี ดังนั้น ค่าที่กำหนดให้กับตัวแปรภายในฟังก์ชัน จึงไม่มีผลใดๆกับตัวแปรที่อยู่ภายนอกฟังก์ชัน หรือสรุปง่ายๆว่าเป็นตัวแปรคนละตัวกันนั่นเอง

รูปแบบการประกาศตัวแปรสามารถใช้ var นำหน้าได้ ซึ่งกรณีที่เราประกาศ var นำหน้าตัวแปรที่อยู่นอกฟังก์ชัน ก็ยังถือว่าเป็นตัวแปรแบบ global อยู่แต่เนื่องจากโปรแกรม flash นั้นมีเรื่องของไทม์ไลน์มาเกี่ยวข้องรวมถึงการซ้อนกันของไทม์ไลน์ หรือมูฟวี่คลิ๊ป ฉะนั้น เราจะถือว่าตัวแปรที่อยู่นอกฟังก์ชันนั้น จะเป็นตัวแปรแบบ local ของระดับไทม์ไลน์นั้นเท่านั้น ส่วนกรณีการจะใช้งานต่างไทม์ไลน์ เราจะได้เรียนรู้ในหัวข้อ target path ภายหลัง รวมถึงการกำหนดตัวแปรแบบ global ในระดับ swf

- ความสัมพันธ์ระหว่างตัวแปรหรือฟังก์ชัน กับ เลเยอร์ และ เฟรม และ ต่างระดับไทม์ไลน์

ในหัวข้อนี้เราจะเรียนรู้ว่าการเขียนโค้ดในแต่ละจุดจะมีผลอย่างไรบ้าง ซึ่งจะเกี่ยวข้องกับตำแหน่งที่ประกาศรวมถึงลำดับในการคอมไพล์คำสั่งของโปรแกรมและการเรียกใช้งานต่อไป โดยจะแบ่งเป็นการเขียนในระดับเลเยอร์ และเฟรม และไทม์ไลน์

1. การเขียนโค้ดต่างเลเยอร์
2. การเขียนโค้ดต่างเฟรม
3. การเขียนโค้ดต่างไทม์ไลน์

สำหรับแต่ละหัวข้อมีคำอธิบายดังนี้

1. การเขียนโค้ดต่างเลเยอร์

การเขียนโค้ดต่างเลเยอร์ แต่อยู่บนไทม์ไลน์เดียวกัน โปรแกรมจะคอมไพล์คำสั่งจาก เลเยอร์บนลงล่าง ฉะนั้นกรณีที่เรามีการประกาศฟังก์ชันเอาไว้ในเลเยอร์ที่อยู่ต่ำกว่า เราจะไม่สามารถ

เรียกใช้งานฟังก์ชันนั้นได้ในตอนที่ swf เริ่มทำงาน แต่สามารถเรียกใช้งานได้ปกติ ณ ช่วงเวลาอื่นๆ (run time)

2. การเขียนโค้ดต่างเฟรม

การเขียนโค้ดต่างเฟรมนั้น เรื่องของลำดับถือว่าไม่สำคัญขอเพียงแต่ต้องมีการประกาศตัวแปรและฟังก์ชันขึ้นมาก่อน ซึ่งเขียนโค้ดเอาไว้ในเฟรมไหนก็ต้องเล่นเฟรมนั้นก่อน หลังจากนั้นก็จะคงอยู่ตลอดไปและเรียกใช้ได้ตลอดทุกเฟรม ในระดับไทม์ไลน์เดียวกัน

3. การเขียนโค้ดต่างไทม์ไลน์

การเขียนโค้ดต่างไทม์ไลน์นั้น จะแบ่งออกเป็น 2 กรณี คือ

3.1 การเขียนโค้ดภายใต้ symbol คนละตัวในระดับไทม์ไลน์เดียวกัน

กรณีที่มีการสร้าง symbol ขึ้นมาหลายตัว ในระดับไทม์ไลน์เดียวกัน และมีการประกาศตัวแปรหรือฟังก์ชันเอาไว้ภายใน symbol นั้น โปรแกรมจะเลือกทำงานจากระดับเลเวลความลึกของ symbol ตัวนั้น (วัตถุที่ถูกสร้างล่าสุด หรือ cut แล้ว paste บน stage ใหม่ จะมีระดับเลเวลหรือการซ้อนทับที่สูงกว่า) โดยพิจารณาจากค่า load order ที่เราเซตเอาไว้ใน public setting ในแถบ flash

ซึ่งกรณีที่เรเลือก Bottom up ก็จะเลือกทำงานจากระดับเลเวลต่ำขึ้นไประดับเลเวลสูงกว่า แต่ถ้าเลือก Top down ก็จะทำงานจากระดับเลเวลสูงขึ้นไปยังเลเวลต่ำกว่า ฉะนั้น กรณีที่จะมีการเรียกใช้คำสั่งข้าม symbol กัน ในตอนที่ swf ทำงานเริ่มต้น จะต้องพิจารณาเรื่องลำดับการทำงานให้ดีว่าตัวแปรและฟังก์ชันที่อ้างอิงถึงนั้นมีการประกาศแล้วหรือยัง

3.2 การเขียนโค้ดภายใต้ symbol ซึ่งอยู่ภายใต้ symbol สูงสุดตัวเดียวกัน

สำหรับกรณีที่มีการซ้อนไทม์ไลน์หรือซ้อนมูฟวี่คลิ๊ป นั้น ลำดับการทำงานของ actionscript จะเริ่มทำงานจากไทม์ไลน์ชั้นนอกสุดแล้วไล่ลำดับเข้าไปภายในไทม์ไลน์ที่อยู่ข้างใน ฉะนั้น กรณีที่มี symbol ชื่อ symbolIn อยู่ภายใน symbolOut และทั้งคู่มีการเขียน actionscript เอาไว้ การทำงานนั้นจะเริ่มทำที่ symbolOut ก่อน แล้วจึงทำคำสั่งใน symbolIn เป็นลำดับถัดไป สำหรับไทม์ไลน์หลักที่วาง symbol เอาไว้ ถือว่าเป็นระดับไทม์ไลน์สูงสุด จึงทำงานก่อน symbol ที่ถูกเอาไว้เช่นเดียวกัน

- รูปแบบการส่งค่าของฟังก์ชัน

การส่งค่าเข้าฟังก์ชันนั้น จะมีรายละเอียดที่ควรรู้เพิ่มเติมกรณีที่เราส่งเป็นตัวแปรเข้าไป ซึ่งประเภทหรือชนิดของตัวแปรนั้นจะมีความแตกต่างกันด้วย ซึ่งจะต่างกันในเรื่องของการนำค่าไปใช้ หรือการนำตำแหน่งอ้างอิงตัวแปรนั้นไปใช้ ดังต่อไปนี้

1. pass by value

การส่งค่าเข้าฟังก์ชันแบบนี้ จะเป็นการส่งตัวแปรประเภทดั้งเดิม (primitive) อาทิเช่น เก็บค่า String, Number, Boolean เป็นต้น โดยการส่งตัวแปรนั้น จะถูกนำไปใช้เพียงค่าที่ตัวแปรนั้นเก็บเอาไว้ ซึ่งเมื่อมีการกำหนดค่าให้กับตัวแปรพารามิเตอร์ที่รับค่าเข้ามา จะไม่เกิดผลกระทบใดๆกับตัวแปรที่ถูกส่งเข้ามา ตัวอย่างเช่น

Code:

```
my_variable = 5;

function passByValue(param) {
    param = 10;
}

trace(my_variable); //output 5

passByValue(my_variable);

trace(my_variable); //output 5
```

จะเห็นว่า เรามีการส่งตัวแปรเข้าไปในฟังก์ชัน ซึ่งภายในฟังก์ชันมีการเปลี่ยนค่าให้กับพารามิเตอร์ที่รับค่าตัวแปรนั้นมา แต่เมื่อเราทดสอบแสดงค่าตัวแปรดังกล่าวอีกรอบ จะเห็นว่าค่าตัวแปรนั้นมีค่าเท่าเดิม เนื่องจากการนำค่าภายในตัวแปรไปใช้เท่านั้น

2. pass by reference

การส่งค่าแบบนี้ เป็นการส่งค่าโดยส่งตำแหน่งอ้างอิงของตัวแปรที่ถูกส่ง ฉะนั้นเมื่อมีการแก้ไขใดๆกับพารามิเตอร์ที่รับค่าตัวแปรนั้นมา จะส่งผลให้ค่าตัวแปรนั้นถูกเปลี่ยนแปลงไปด้วย โดยส่วนใหญ่จะเป็นตัวแปรที่มีชนิดเป็น object ทั้งหลาย รวมถึง array ด้วย (อันที่จริงแล้ว array ก็ถูกสร้างจากคลาส Array จึงถือว่าเป็น object เช่นกัน)

ตัวอย่างการส่งค่าอาร์เรย์ เข้าฟังก์ชัน

Code:

```
my_array = [1]; //เป็นการสร้างอาร์เรย์แบบเปล่าๆ โดยเก็บค่า 1 เอาไว้

function passByReference(param) {

    param.push(2); //เป็นการเพิ่มค่าเก็บใส่อาร์เรย์

}

trace(my_array); //output 1

passByReference(my_array); //ส่งอาร์เรย์เข้าฟังก์ชัน

trace(my_array); //output 1,2
```

จากตัวอย่างเป็นการสร้างอาร์เรย์ ซึ่งเอาไว้เก็บค่าต่างๆในแต่ละลำดับ โดยจากตัวอย่างเราเก็บค่า 1 เอาไว้ หลังจากนั้น เราได้ทำการส่งอาร์เรย์ดังกล่าวเข้าสู่ฟังก์ชัน แล้วทดสอบแสดงค่าอาร์เรย์อีกรอบ ปรากฏว่า ได้ค่า 2 เพิ่มขึ้นมาด้วย แปลว่า ตัวแปร param นั้นเก็บตำแหน่งสำหรับอ้างอิงตัวแปร my_array เอาไว้ ฉะนั้นเมื่อทำอะไรก็ตามกับ param ก็จะเท่ากับทำกับ my_array นั่นเอง

ตัวอย่างการส่ง object เข้าฟังก์ชัน

Code:

```
my_obj = {no:1};

function passByReference(param) {

    delete param.no;

}

trace(my_obj.no); //ต้องการแสดงตัวแปร no ใน object ชื่อ my_obj output ได้ 1

passByReference(my_obj); trace(my_obj.no); //undefined
```

จากตัวอย่างนั้น เริ่มต้นเราได้ทำการสร้าง object ซึ่งมีตัวแปรชื่อว่า no ซึ่งเก็บค่า 1 เอาไว้ เมื่อแสดงค่า no ของตัวแปรของ my_obj จึงได้ค่า 1 แสดงออกมา แต่เมื่อทำการเรียกฟังก์ชันโดยส่งตัวแปร my_obj เข้าไป แล้วทำการแสดงค่าอีกรอบ ปรากฏว่าค่าดังกล่าวหายไป ได้ค่าเป็น undefined เนื่องจากตัวแปรดังกล่าวถูกลบจากในตัวฟังก์ชันเรียบร้อยแล้ว



บทที่ 3 แผนงานการปฏิบัติงานและขั้นตอนการดำเนินงาน

3.1 แผนงานปฏิบัติงาน

ตารางที่ 3.1 แผนงานปฏิบัติงาน

หัวข้องาน	มิถุนายน	กรกฎาคม	สิงหาคม	กันยายน
ศึกษาทฤษฎีขั้นพื้นฐานและการนำไปประยุกต์ใช้ - ศึกษาการออกแบบเกม และการทำงานของโปรแกรม - เขียนเกมง่ายๆ เพื่อเป็นการศึกษาโครงสร้างและการทำงาน - เรียนรู้ทฤษฎีของ โค้ด และ ฟังก์ชัน ต่างๆ				
ศึกษา Program และ การประยุกต์เพื่อการใช้งาน - ศึกษาโปรแกรม Adobe Flash ขั้นพื้นฐานและการนำไปใช้ - ทดลองเขียนโปรแกรม เพื่อศึกษา ฟังก์ชัน และ โค้ด - ตรวจสอบการ ทดสอบ และ ข้อผิดพลาด				
ออกแบบ และ สร้างเกม - กำหนดรูปแบบการเล่นและ ต่างที่ใช้แสดงผลต่างๆของเกมของเกม - ออกแบบ ฟังก์ชัน และ กำหนดค่า ลอจิก - เขียนโปรแกรม และ ต่างที่ใช้แสดงผลต่างๆของเกม ในแต่ละ ฟังก์ชัน - ทดสอบ และ ตรวจสอบข้อผิดพลาด				
พัฒนา และ ต่อยอด				
เขียน รายงาน				

3.2 รายละเอียดงานที่นักศึกษาปฏิบัติในงานสหกิจศึกษา หรือรายละเอียดโครงการที่ได้รับมอบหมาย

3.2.1 หลักการ และเหตุผล

บริษัท ดิจิคราฟต์ จำกัด เป็นบริษัทที่ให้บริการด้าน Digital Content เน้นการออกแบบ และพัฒนา สื่อออนไลน์ต่างๆ เช่น เกม 2 มิติ, เกม 3 มิติ, เว็บไซต์ และอื่นๆอีกมากมาย โดยบริษัทได้มีการแบ่งแผนกการจัดทำสื่อต่างๆเป็นแผนกอย่างชัดเจน ซึ่งผู้ศึกษาได้ปฏิบัติงานสหกิจในตำแหน่ง Programmer โดยงานที่ได้รับมอบหมายคือการศึกษา จัดทำและพัฒนาเกม 2 มิติจากโปรแกรม Adobe Flash และ Action Script 2.0 โดยมีจุดประสงค์เพื่อการเรียนรู้ และสามารถนำการศึกษาในครั้งนี้ไปพัฒนาและต่อยอดองค์ความรู้ต่างๆที่เคยศึกษา

3.3 ขั้นตอนการดำเนินงานที่นักศึกษาปฏิบัติงานหรือโครงการ

3.3.1 ติดตั้ง Software

Action Script เป็นภาษาด้าน Programming ที่ใช้ในโปรแกรม Flash โดยเป็นการเขียนสคริปต์เพื่อควบคุมการทำงาน ด้านต่างๆ ของชิ้นงานที่สร้างขึ้น Action Script เป็นเหมือนตัวเชื่อมระหว่างสิ่งที่ผู้ใช้เข้าใจกับสิ่งที่ Flash เข้าใจ การทำงานจะขึ้นอยู่กับเหตุการณ์ (Event) ที่เกิดขึ้น หากเหตุการณ์นั้นไม่เกิดขึ้นก็ไม่มีการทำงาน จึงทำให้งานที่สร้างจากโปรแกรม Flash และการเขียน Action Script ควบคุมจะไม่ใช้ Animation ธรรมดาอีกต่อไป แต่จะเป็นงานที่สามารถโต้ตอบกับผู้ใช้กับได้อย่างเต็มรูปแบบ

3.3.1.1 Adobe Flash CS3

เป็นโปรแกรมที่ใช้ในการเขียนสื่อมัลติมีเดียที่เอาไว้ใช้สร้างเนื้อหาเกี่ยวกับ Flash ซึ่งตัว Flash Player พัฒนาและเผยแพร่โดย อะโดบีซิสเต็มส์ (เริ่มต้นพัฒนาโดยบริษัท ฟิวเจอร์แวร์ ตอนหลังเปลี่ยนเป็น แมโครมีเดีย ซึ่งภายหลังถูกควบรวมกิจการเข้ากับ อะโดบี) ซึ่งเป็นโปรแกรมที่ทำให้ เว็บเบราว์เซอร์ สามารถแสดงตัวมันได้ ซึ่งมันมีความสามารถในการรองรับ ภาพแบบเวกเตอร์ และ ภาพแบบแรสเตอร์ และมีภาษาสคริปต์ที่เอาไว้ใช้เขียนโดยเฉพาะเรียกว่า แอ็กชันสคริปต์ (Action Script) และยังสามารถเล่นเสียงและวิดีโอ แบบสเตริโอได้

แต่ในความหมายจริงๆ แล้ว แฟลช คือโปรแกรมแบบ Integrated development environment (IDE) และ Flash Player คือ virtual machine ที่ใช้ในการทำงานงานของไฟล์ แฟลชซึ่ง

ในภาษาพูดเราจะเรียกทั้งสองคำนี้ในความหมายเดียวกัน: "แฟลช" ยังสามารถความความถึงโปรแกรมเครื่องมือต่างๆตัวแสดงไฟล์หรือ ไฟล์โปรแกรม

เทคโนโลยีแฟลชได้กลายมาเป็นที่นิยมในการเสนอ แอนิเมชัน และ อินเตอร์แอคทีฟ ในเว็บเพจ และในโปรแกรมหลายๆ โปรแกรมระบบ และ เครื่องมือต่างๆ ที่มีความสามารถในการแสดง แฟลชได้ และ แฟลชยังเป็นที่นิยมในการใช้สร้าง แคมพิวเตอร์แอนิเมชันโฆษณาออกแบบส่วนต่างๆ ของเว็บเพจใส่วิดีโอบนเว็บ และอื่นๆ อีกมากมาย

ไฟล์ Flash ในบางครั้งอาจเรียกว่า "flash movies" โดยทั่วไปกับไฟล์ที่มีนามสกุล .swf และ .flv

3.3.1.2 Adobe Photoshop CS3

โปรแกรม Photoshop เป็นโปรแกรมในตระกูล Adobe ที่ใช้สำหรับตกแต่งภาพถ่ายและภาพกราฟิก ได้อย่างมีประสิทธิภาพ ไม่ว่าจะเป็นงานด้านสิ่งพิมพ์ นิตยสาร และงานด้านมัลติมีเดีย อีกทั้งยังสามารถ retouching ตกแต่งภาพและการสร้างภาพ ซึ่งกำลังเป็นที่นิยมสูงมากในขณะนี้ เราสามารถใช้โปรแกรม Photoshop ในการตกแต่งภาพ การใส่ Effect ต่าง ๆ ให้กับภาพ และตัวหนังสือ การทำภาพขาวดำ การทำภาพถ่ายเป็นภาพเขียน การนำภาพมารวมกัน การ Retouch ตกแต่งภาพต่าง

3.3.2 ศึกษาทฤษฎีขั้นพื้นฐานของการเขียนเกม

โดยเริ่มศึกษาประเภทของเกมว่ามีอะไร และแต่ละประเภทยังมีความแตกต่างกันอย่างไรเพื่อหาข้อมูลมาทำโปรเจกต์ ซึ่งเป็นเกม 2 มิติ

ซึ่งเริ่มจากการออกแบบการทำงานข้างต้นของเกม และต่างที่ใช้แสดงผลต่างๆของเกม ภาพรวมเพื่อนำไปศึกษาและหาจุดบกพร่องเพื่อนำไปต่อยอดมาเป็นเกม ซึ่งในช่วงแรกๆได้มีการเปลี่ยนแปลงเกมต่างๆเมื่อเจอจุดบกพร่อง และต่อมาได้นำจุดบกพร่องมาพัฒนาแนวความคิดของผู้ปฏิบัติสหกิจใหม่ และคิดแนวเกมที่เหมาะสมต่อไป

โดยขั้นต้นนั้นได้มีการศึกษาเรื่องของโปรแกรม Adobe Flash และ Action script 2.0 ขั้นต้นด้วยการฝึกเขียนเกมง่ายๆ เช่น เกมเป่ายิงฉุบ

โค้ด:

```
_root.ran = ร่ม (3) +1;
```

```
Trace (_root.ran);
```

```
trace(_root.a);
```



```

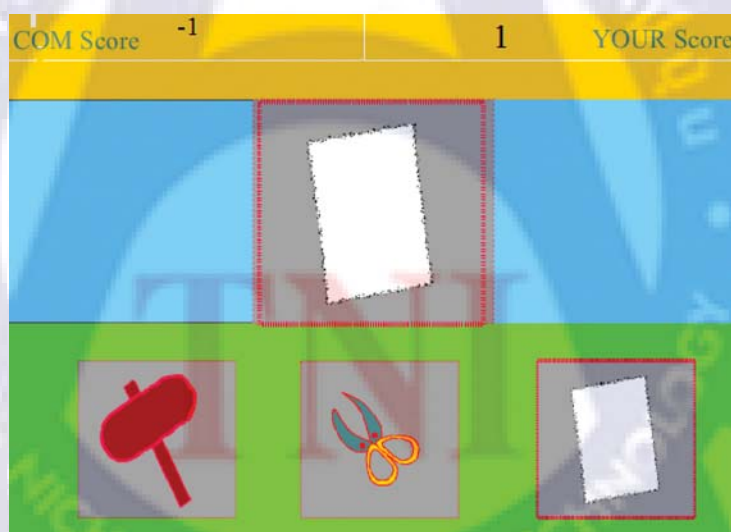
if(_root.ran == 1) {
    gotoAndStop(_root.ran+1);

} else if(_root.ran == 2) {
    gotoAndStop(_root.ran+1);

} else if(_root.ran == 3) {
    gotoAndStop(_root.ran+1);
}

```

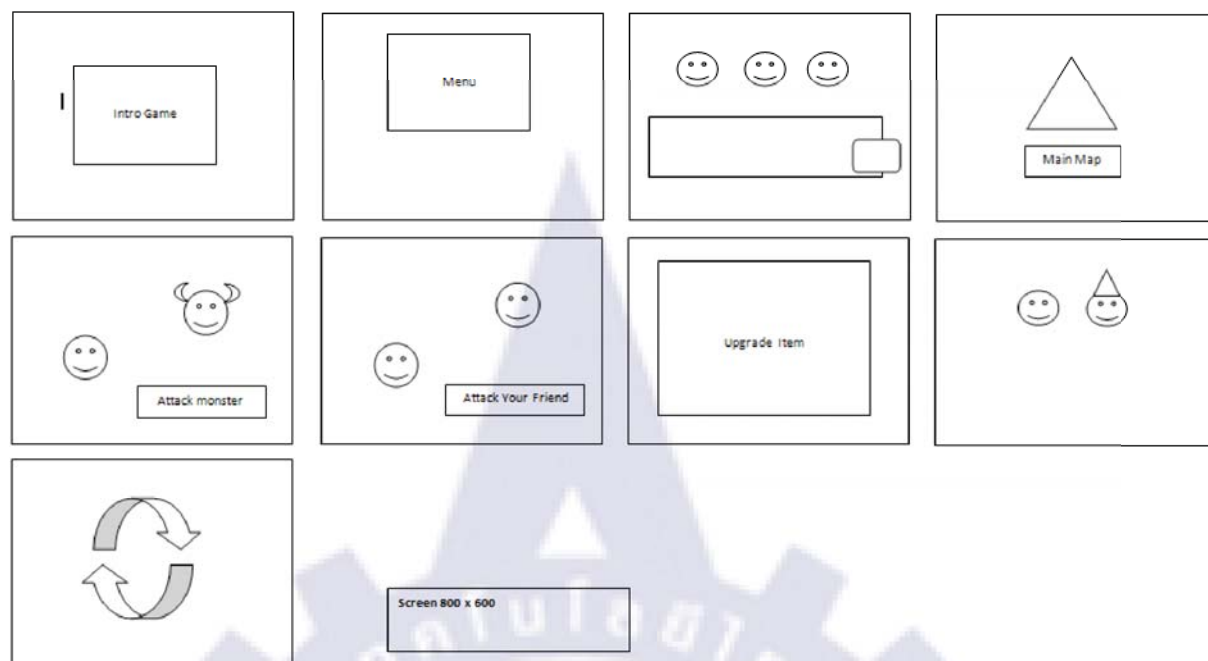
โค้ดข้างต้นคือการสุ่มค่าออกมาเพื่อนำไปใช้ในการสุ่มเกมว่าจะออกก้อน กรรไกร หรือ กระดาษ โดยจากการศึกษาเรื่องการสุ่มนี้ได้นำไปใช้ในการพัฒนาเกมต่อไปด้วย



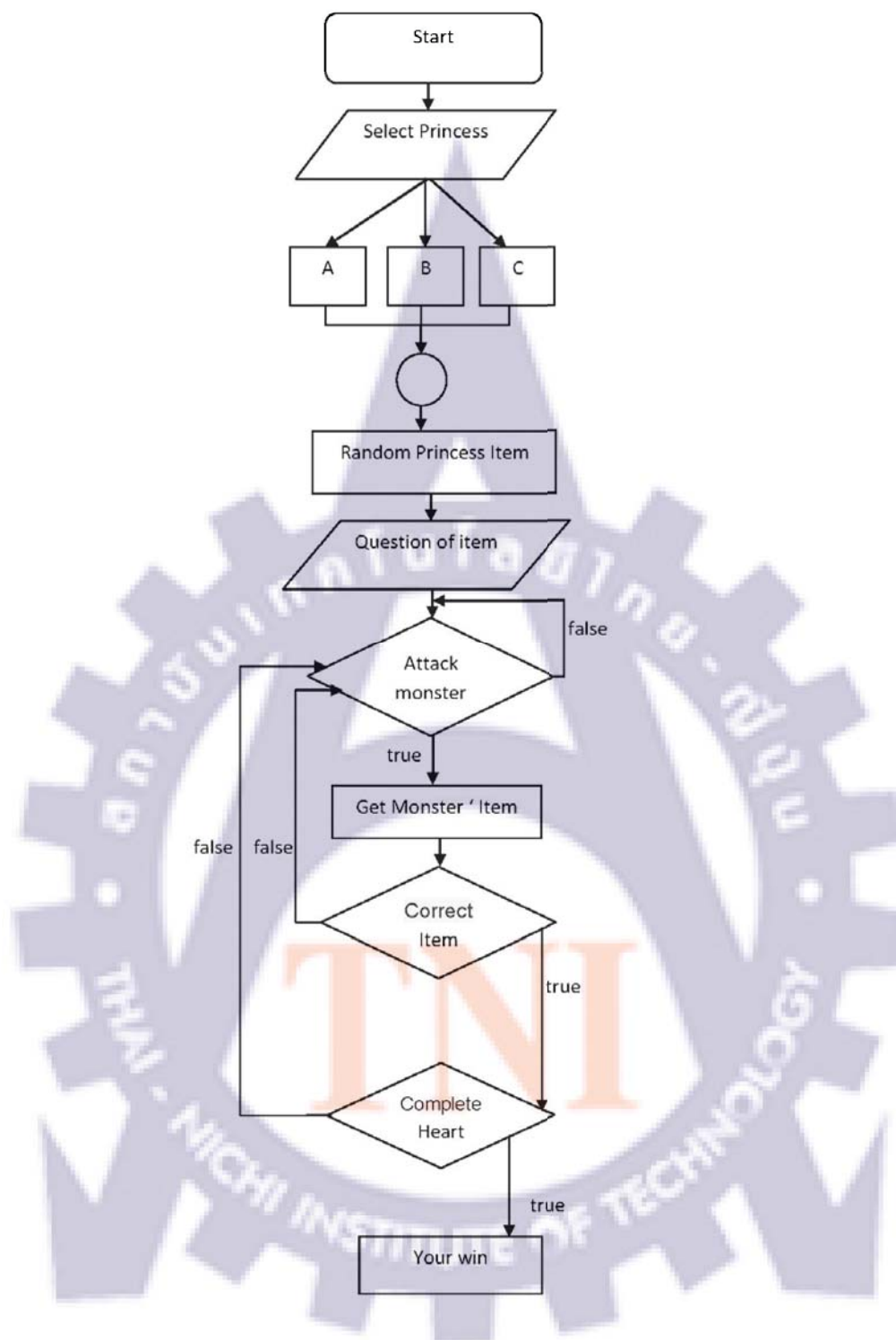
ภาพที่3.1 หน้าต่างที่ใช้แสดงผลต่างๆของเกม เป้ายิงฉุบ

3.3.2.1 ขั้นตอนการพัฒนาเกมขั้นต้น

1. คิดการทำงานของเกมโดยรวมทั้งหมด ว่าในเกมนั้นมีเนื้อเรื่องอะไร ตัวละครที่ตัวทำอะไรได้บ้าง การดำเนินของตัวละครเป็นอย่างไร เป็นต้น



ภาพที่3.3 หน้าต่างที่ใช้แสดงผลต่างๆของเกม แบบ Pop Up Screen โดยรวมทั้งหมดของเกมที่ได้ศึกษา



ภาพที่ 3.4 Flow Chart การทำงานของเกมที่ได้ศึกษา

ซึ่งหลังจากศึกษาเกี่ยวกับ Action script พอสมควรก็ได้มีการออกแบบมินิเกม 1 เกม โดยได้นำความรู้ข้างต้นทั้งหมดที่ได้ศึกษามาประยุกต์ใช้ออกมาเป็นเกม Sudoku ซึ่งเป็นเกมเกี่ยวกับการเรียงตัวเลข

3.3.3 การออกแบบ พัฒนาเกม 2 มิติ

Sudoku คือ เกมปริศนาตัวเลข ที่ผู้เล่นต้องเลือกใส่ หมายเลขตั้งแต่ เลข 1 ถึงเลข 9 โดยมีเงื่อนไขว่าในแต่ละแถวและแต่ละหลักตัวเลขต้องไม่ซ้ำกัน ตารางซูโดะกุจะมี 9×9 ช่อง ซึ่งประกอบจากตารางย่อย 9 ตาราง ในลักษณะ 3×3 แบ่งแยกกันโดยเส้นหนา และในแต่ละตารางย่อยจะต้องมีตัวเลข 1 ถึง 9 เช่นเดียวกัน เมื่อเริ่มเกมจะมีตัวเลขบางส่วนให้มาเป็นคำใบ้ และผู้เล่นจะต้องใส่ทุกช่องที่เหลือให้ครบ โดยตามเงื่อนไขว่าแต่ละตัวเลขในแต่ละแถวและหลักจะใช้ได้ครั้งเดียว รวมถึงในแต่ละขอบเขตตารางย่อย การเล่นเกมนี้จำเป็นต้องใช้ความสามารถในด้าน ตรรกะ และความอดทนรวมถึงสมาธิ

3.3.3.1 ขั้นตอนการพัฒนาเกม Sudoku

1. ศึกษาการทำงานของเกมที่คล้ายๆกัน

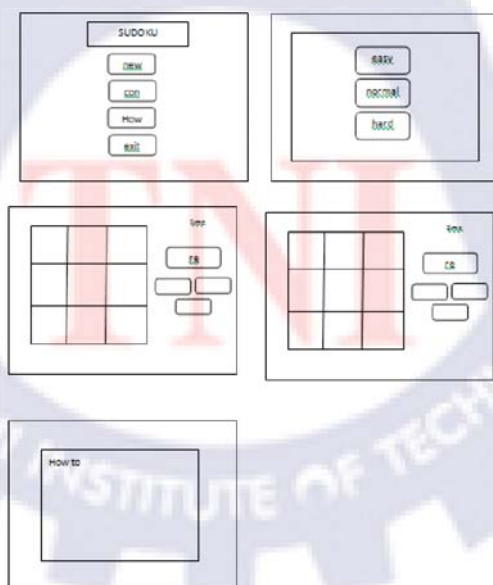


ภาพที่ 3.5 หน้าต่างที่ใช้แสดงผลต่างๆของเกม Sudoku ต้นแบบที่ศึกษา

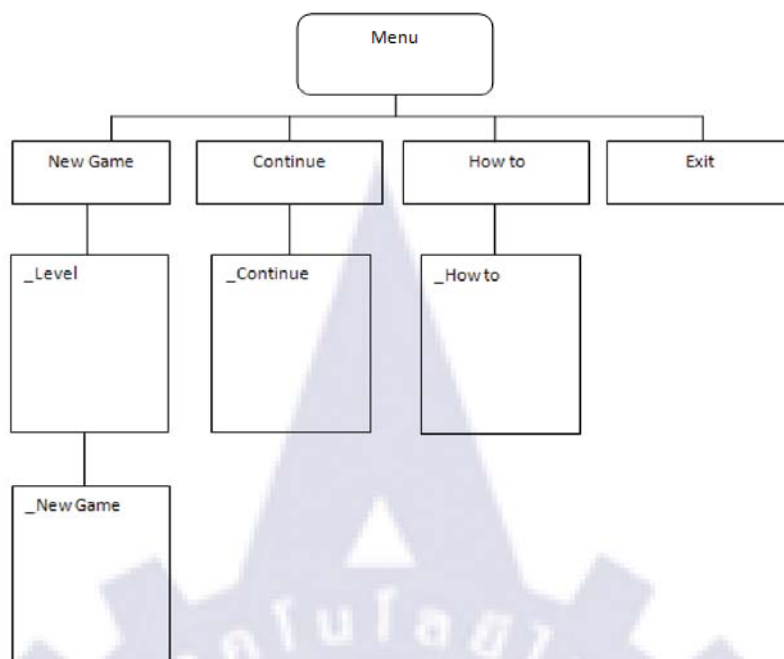


ภาพที่3.6 หน้าต่างที่ใช้แสดงผลต่างๆของเกม Sudoku ที่แสดงวิธีการเล่น

2. การออกแบบ ต่างที่ใช้แสดงผลต่างๆของเกม และ ขั้นตอนการทำงานของเกม ซึ่งได้แบ่งออกเป็นภาพรวมทั้งหมดของเกม

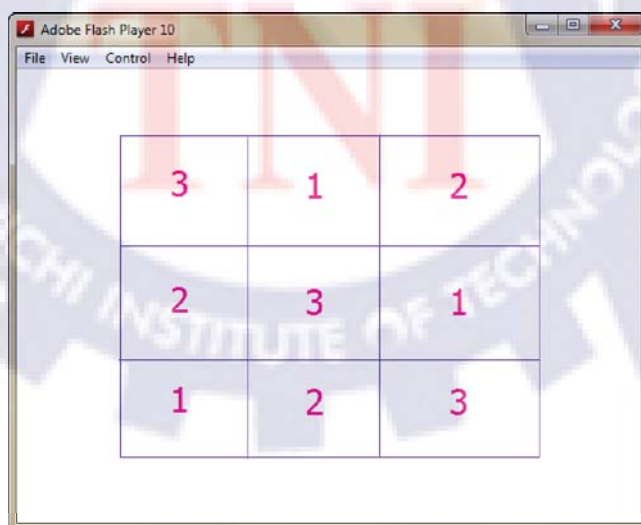


ภาพที่3.7 หน้าต่างที่ใช้แสดงผลต่างๆของเกม Sudoku ที่ออกแบบ



ภาพที่3.8 ลำดับการทำงานทั้งหมดของเกม Sudoku

3. ศึกษา ลอจิกของเกมที่คล้ายกัน โดยได้เลือกเป็นเกม Sudoku เช่นเดียวกัน แต่เป็นตาราง 3x3 การศึกษาเกม Sudoku 3x3 ขั้นต้นนั้นมีการคิดแบบลอจิกการคำนวณขึ้นมาโดยสามารถนำไปประยุกต์ให้กับตาราง 9x9 ได้เช่นกัน



ภาพที่3.9 ตาราง Sudoku 3x3 ที่ได้ศึกษา

โค้ด:

```
var numArray:Array = new Array("1", "2", "3");
var splNum:Array = new Array();
var pushNum:Array = new Array();

for (var i = 1; i<=3; i++) {
    var ran = สุ่ม(numArray.length);
    trace("ran :"+ran);
    trace("numarray :"+numArray[ran]);
    splNum = numArray.splice(ran, 1);
    trace("splNum : "+splNum);
    trace(" ");
    pushNum.push(splNum);
    trace("pushNum: "+pushNum);
    trace(" ");
}
for (var j = 0; j<3; j++) {
    this["abc_txt_"+(j)].abc_txt.text = pushNum[j];
}
```

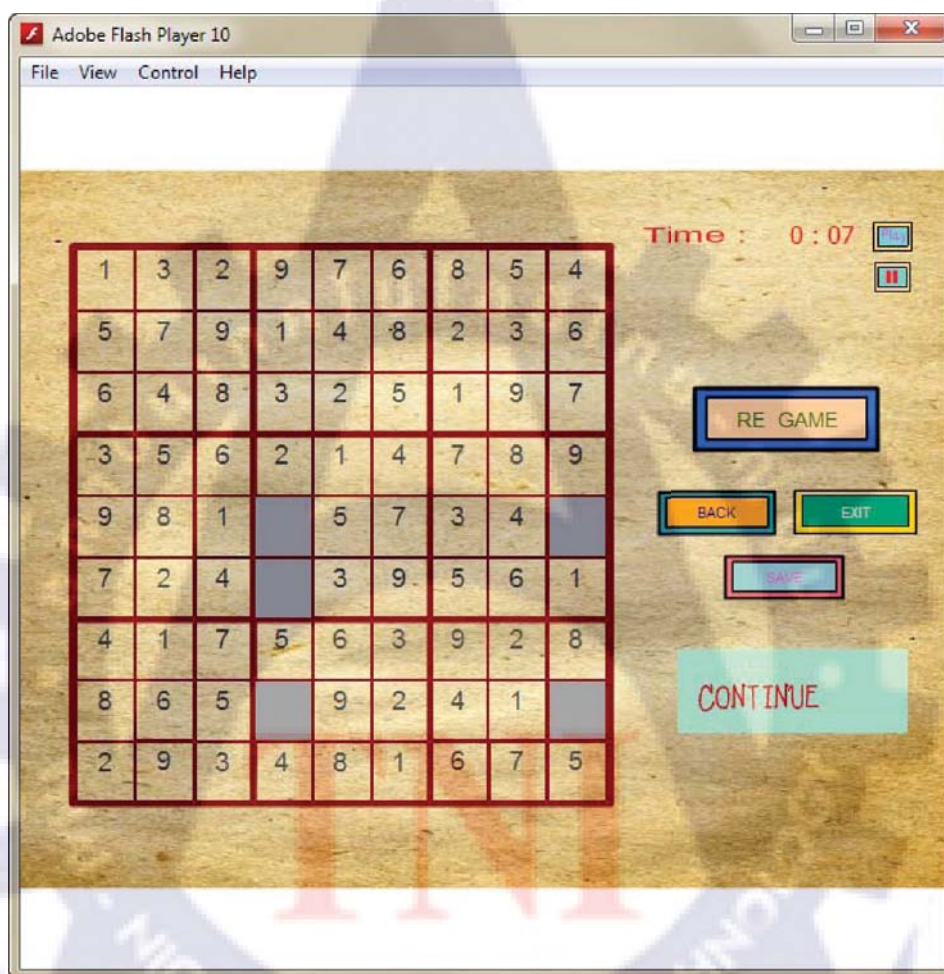
ข้างต้นเป็นตัวอย่างโค้ด Sudoku 3x3 ที่ได้นำความรู้เรื่องการ สุ่ม มาพัฒนาและสามารถนำความรู้เรื่องนี้ไปต่อยอดเป็นเกมต่างๆได้มากมาย เช่น Sudoku 9x9 โดยสามารถนำลอจิกนี้มาพัฒนาต่อไป

4. นำความรู้จาก Sudoku 3x3 มาพัฒนาลอจิกเป็น Sudoku 9x9

ขั้นตอนการออกแบบ Sudoku 9x9

1. ศึกษาการทำงานทั้งหมดของเกม โดยเขียนออกมาเป็น ว่าทำอะไร ที่ไหน และเมื่อไร
2. ออกแบบลอจิกของเกมทั้งหมดโดยเน้นไปที่ ฟังก์ชันหลักของเกม
3. ออกแบบหน้าต่างที่ใช้แสดงผลต่างๆของเกม

4. ออกแบบฟังก์ชัน และค่าตัวแปรต่างๆที่จำเป็น
5. แบ่งฟังก์ชันต่างๆออกมาเพื่อเขียนโค้ดทดสอบแต่ละฟังก์ชัน
6. นำฟังก์ชันมารวมกัน และทดสอบ
7. วาดหน้าตาที่ใช้แสดงผลต่างๆของเกมทั้งหมดตามที่ออกแบบ
8. ทดสอบเกม และพัฒนาในบางจุดที่ไม่สมบูรณ์



ภาพที่3.10 หน้าตาที่ใช้แสดงผลต่างๆของเกม เกม Sudoku 9x9 ในช่วงทดสอบ

โค้ด :

```
ฟังก์ชัน ตรวจสอบWin() {
    var winArray:Array = new Array();
    for (var i = 0; i<9; i++) {
```

```

        for (var j = 0; j<9; j++) {
            if (_root["_"+(i)+"_"+(j)]._currentframe != 2 && tmpArray[i][j] != ""
&& tmpArray[i][j] != NaN) {

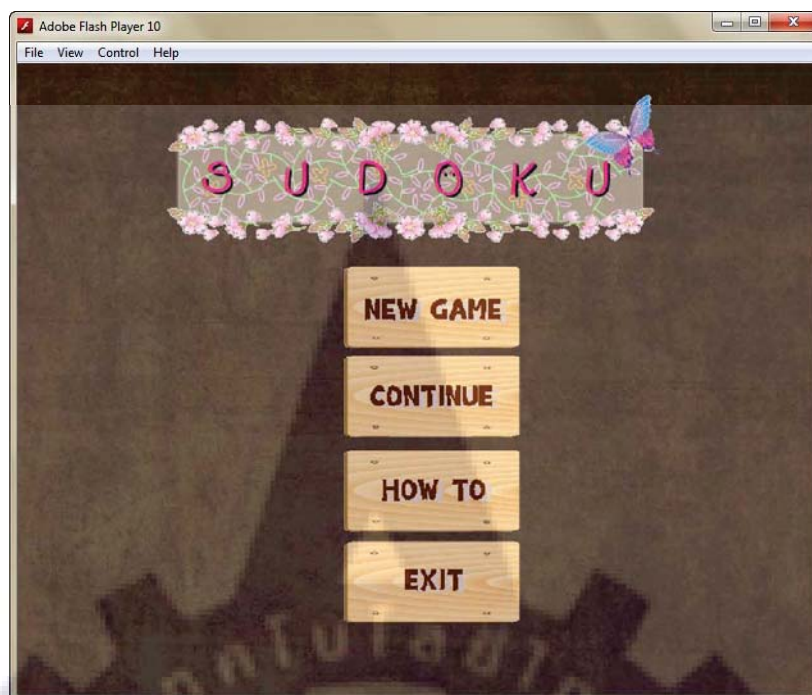
                var win:Number = 0;
                winArray.push(win);
                if (winArray.length == 81) {

                    trace("win");
                    youWin._visible = true;
                    delete onEnterFrame;
                    youWin.onPress = ฟังก์ชัน() {
                        genSudoku();
                        onEnterFrame = onCountTime;
                    };
                }
            }
        }
    }
}

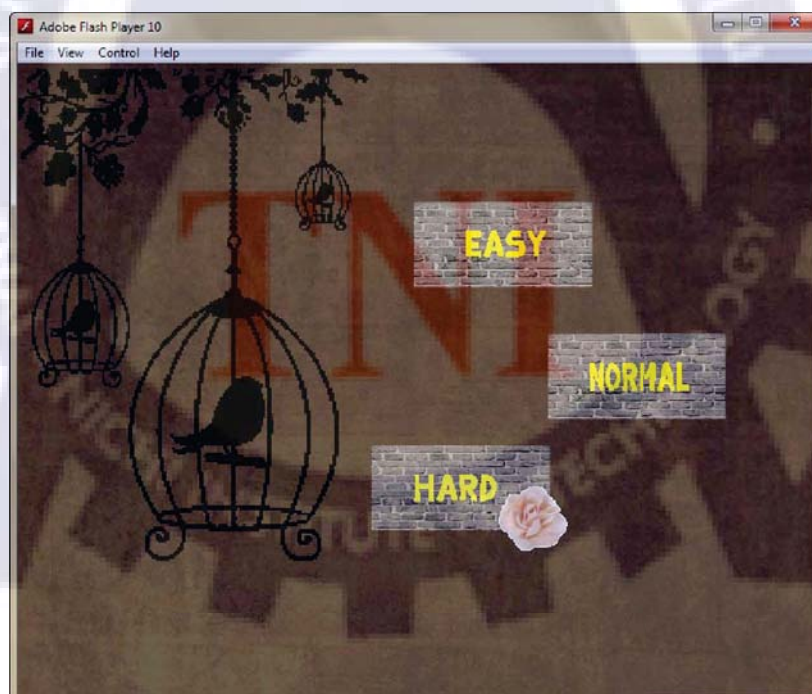
```

ตัวอย่าง โค้ดฟังก์ชันตรวจสอบ เกม Sudoku ว่าชนะหรือแพ้ โดยข้างต้นตรวจสอบจากค่าของตำแหน่งเฟรมที่เป็นช่องว่างและค่า NaN

5. พัฒนาและแก้ไขรูปแบบของหน้าต่างที่ใช้แสดงผลต่างๆของเกม ขึ้นต้นใหม่เพื่อให้ดูน่าสนใจและน่าเล่นมากยิ่งขึ้น



ภาพที่3.11 หน้าต่างที่ใช้แสดงผลต่างๆของเกม เกม Sudoku หน้าเมนู



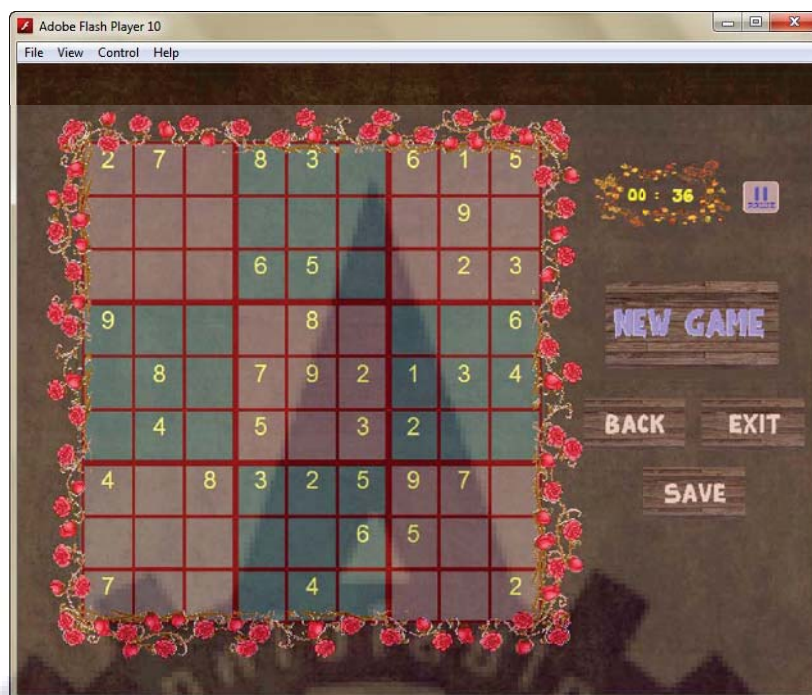
ภาพที่3.12 หน้าต่างที่ใช้แสดงผลต่างๆของเกม เกม Sudoku หน้า Level



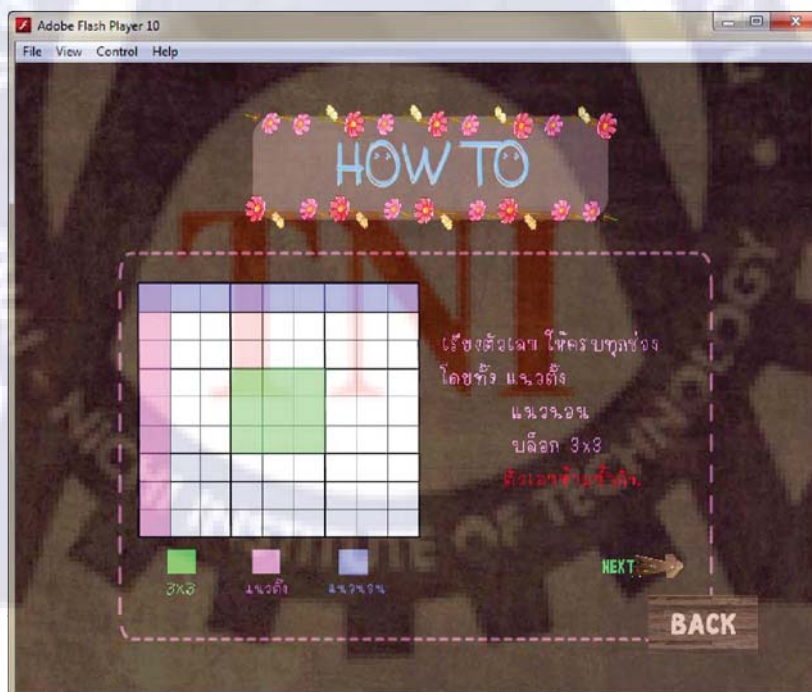
ภาพที่3.13 หน้าต่างที่ใช้แสดงผลต่างๆของเกม เกม Sudoku หน้า Game เมื่อเลือก Level Normal



ภาพที่3.14 หน้าต่างที่ใช้แสดงผลต่างๆของเกม เกม Sudoku หน้า Game เมื่อต้องการใส่ตัวเลข



ภาพที่3.15 หน้าต่างที่ใช้แสดงผลต่างๆของเกม Sudoku หน้า Continue เมื่อเลือก Save



ภาพที่3.16 หน้าต่างที่ใช้แสดงผลต่างๆของเกม เกม Sudoku หน้า How to

บทที่ 4 ผลการดำเนินงาน การวิเคราะห์และสรุปผลต่างๆ

4.1 ขั้นตอนและผลการดำเนินงาน

โดยแบ่งขั้นตอนการดำเนินงานออกเป็น 4 ช่วงทั้งหมดตามระยะเวลา

4.1.1 ขั้นตอนการศึกษาทฤษฎีขั้นพื้นฐานและการนำไปประยุกต์ใช้

4.1.2 ขั้นตอนการศึกษาโปรแกรม และ การประยุกต์เพื่อการใช้งาน

4.1.3 ขั้นตอนการออกแบบ และ สร้างเกม

4.1.4 ขั้นตอนการพัฒนาและต่อยอด

4.1.1 ขั้นตอนการศึกษาทฤษฎีขั้นพื้นฐานและการนำไปประยุกต์ใช้

โดยได้มีการเริ่มเรียนรู้เกี่ยวกับการเขียนเกม เช่นการทำงาน การออกแบบ ระบบฟังก์ชัน และการประมวลผล ซึ่งรวมไปถึงความรู้พื้นฐานด้านการเขียนโปรแกรม เพื่อคิดค้นหาตัวอย่างเกม มาเพื่อเขียนเกมต้นแบบขึ้นมาหนึ่งเกม โดยได้มีการคิดระบบเกมขึ้นมาใหม่ให้มีความน่าสนใจและแปลกใหม่อีกขึ้น

ซึ่งในขั้นตอนนี้ได้มีการฝึกเขียนเกมเป่ายังอุบขึ้นมาเพื่อเป็นการ ทดสอบความรู้พื้นฐาน และกระบวนการความคิดในเรื่องการเขียนโปรแกรมนั่นเอง

ผลจากการดำเนินงานข้างต้น ทำให้เรามีความรู้เรียนการออกแบบและเขียนเกมต่างๆมากขึ้น รวมทั้งการเขียนโปรแกรมขั้นพื้นฐานอีกด้วย

4.1.2 ขั้นตอนการศึกษา โปรแกรม และ การประยุกต์เพื่อการใช้งาน

ได้มีการเรียนรู้เกี่ยวกับการใช้งานโปรแกรม Adobe Flash มากขึ้น ทั้งฟังก์ชัน ชุดคำสั่ง สำเร็จรูปรวมถึงกระบวนการทำงานของโปรแกรมขั้นพื้นฐาน เช่นการเรียกใช้งาน แทบต่างๆ เมนู และอื่น ๆ

ซึ่งขั้นตอนนี้ได้รวมไปถึงการใช้งานโปรแกรมอื่นๆที่เกี่ยวข้อง เช่น Adobe Photoshop เป็นต้น โดยเราสามารถทำการประยุกต์โปรแกรมต่างๆเข้าด้วยกันเพื่อความสะดวกในการเขียนโปรแกรมของเรายิ่งขึ้น

ผลจากการดำเนินงานข้างต้น ทำให้เราสามารถให้โปรแกรม Adobe Flash และ โปรแกรมอื่นๆ ได้มากขึ้น ทั้งยังสามารถนำความรู้เรื่องการใช้โปรแกรมนี้ไปประยุกต์เพื่อใช้เขียนเกมต่อไป

4.1.3 ขั้นตอนการออกแบบ และ สร้างเกม

ขั้นตอนนี้เป็นการนำความรู้ข้างต้นทั้งหมดมาประยุกต์ให้จริง คือการออกแบบเกม 2 มิติ หนึ่งเกมจากความรู้ที่ได้ศึกษามาทั้งหมด โดยเริ่มตั้งแต่การออกแบบเกม การเขียนฟังก์ชัน การประมวลผล เป็นต้น โดยหลังจากที่เราได้ต้นแบบของการสร้างเกมแล้ว ก็สามารถนำมาเขียนโปรแกรมได้ง่ายยิ่งขึ้น โดยการออกแบบในที่นี้คือการสร้างต้นแบบเกม Sudoku ซึ่งเป็นเกมที่มีการคิดคำนวณลอจิกที่ค่อนข้างเยอะ ดังนั้นการคิดแบบการประมวลผลจึงค่อนข้างละเอียดตามไปด้วย

ผลจากการดำเนินงานข้างต้น ทำให้เรารู้ข้อดีข้อเสียด้านความคิดของเรามากขึ้น คือถ้าเมื่อเกิดปัญหาเราจะมีวิธีแก้อย่างไร การคิดลอจิกที่ต้องละเอียด เพื่อให้ขั้นตอนการเขียนโปรแกรมที่ไม่ติดขัดนั่นเอง

4.1.4 ขั้นตอนการพัฒนาและต่อยอด

ขั้นตอนนี้เป็นขั้นตอนสุดท้ายในการพัฒนาเกมทั้งหมด โดยเมื่อเราได้ต้นแบบที่สมบูรณ์แล้ว เราสามารถนำต้นแบบนั้นมาพัฒนา ปรับปรุงทั้งตัวโค้ด และ หน้าต่างที่ใช้แสดงผลต่างๆ ของเกมนั้นๆ ได้ ซึ่งการพัฒนานี้ขึ้นอยู่กับความเหมาะสมและองค์ประกอบต่างๆ ในการเขียนโปรแกรมขั้นต้นอีกด้วย

ผลจากการดำเนินงานข้างต้น ณ จุดนี้เราจะมองเห็นข้อผิดพลาดในการเขียนโปรแกรมของเรามากขึ้น ทั้งการเขียนฟังก์ชัน การตกแต่ง หรือแม้การออกแบบทั้งหมดในข้างต้นก็ตาม ซึ่งเราสามารถปรับปรุง Version ของเกมได้ให้ดีที่สุด在这一ขั้นตอน

4.2 ผลการวิเคราะห์ข้อมูล

จากผลที่ได้จากการศึกษาและการปฏิบัติงานสหกิจตลอด 4 เดือน ซึ่งผลที่ได้เป็นที่น่าพึงพอใจ คือ ตัวผู้ปฏิบัตินั้นได้รับความรู้เกี่ยวกับโปรแกรม Adobe Flash ขึ้นพื้นฐานมากขึ้น และยังรวมถึงการเขียนโปรแกรมด้วย Action script 2.0 อีกด้วย ซึ่งจากการปฏิบัติงานครั้งนี้ทำให้ตัวผู้ปฏิบัติได้รับประสบการณ์การทำงานจริง ระบบการทำงานต่างๆ หน้าที่ความรับผิดชอบที่ได้รับ กระบวนการคิด การประยุกต์ใช้งาน การแก้ปัญหาเฉพาะหน้าต่างๆ และการเรียนรู้ระบบการทำงานจริงๆ ภายใตบริษัท ซึ่งถือว่าเป็นประสบการณ์ที่ดีมากอีกด้วย

4.3 วิเคราะห์และวิจารณ์ข้อมูลโดยเปรียบเทียบผลที่ได้รับกับวัตถุประสงค์และ จุดมุ่งหมายในการปฏิบัติงานหรือการจัดทำโครงการ

ตารางที่ 4.1 เปรียบเทียบผลที่ได้รับกับวัตถุประสงค์และจุดมุ่งหมายในการปฏิบัติงานหรือการผล
จัดทำโครงการ

วัตถุประสงค์	ผลที่ได้รับ
เพื่อศึกษาการใช้งาน โปรแกรมAdobe Flash และAction Script 2.0 ขั้นพื้นฐาน	ได้รับความรู้เกี่ยวกับการใช้งาน Adobe Flash มากขึ้น และสามารถนำAction Script 2.0 มาใช้ในการพัฒนาเกม 2 มิติได้ แต่ยังไม่สมบูรณ์ 100 %
เพื่อให้ได้ประสบการณ์ในการทำงานจริง	ได้รับประสบการณ์การทำงานจริงและประสบการณ์ที่ดีๆต่างมากมาย จากพี่ๆ และบริษัท
เพื่อให้ได้ความรู้และเทคนิคการทำงานต่างๆ ด้านเทคโนโลยีสารสนเทศมากขึ้น	ซึ่งผลที่ได้รับถือว่าเป็นความรู้ใหม่ และสามารถนำไปศึกษาต่อเพื่อให้สามารถพัฒนาเกม หรือโปรแกรมต่างๆที่จัดทำขึ้นจากโปรแกรม Adobe Flash และ Action Script 2.0
เพื่อสร้างลักษณะนิสัยในการทำงานการสื่อสาร และการอยู่ร่วมกันในสังคม	ได้รับการต้อนรับ การดูแลเอาใจใส่เป็นอย่างดี และประทับใจในการปฏิบัติสหกิจศึกษาในครั้งนี้เป็นอย่างมาก

บทที่ 5 บทสรุปและข้อเสนอแนะ

5.1 สรุปผลการดำเนินงาน

จากการทำงานครั้งนี้ ถือว่าประสบความสำเร็จในระดับดี เนื่องจากได้ประสบการณ์การทำงานจริง ได้ความรู้ในเรื่องต่างๆมากมาย อาทิเช่น การใช้งานโปรแกรม Adobe Flash, Action script 2.0, Adobe Photoshop และอื่นๆ และยังได้ฝึกฝนการทำงานจริง การประสานงาน การทำงานร่วมกับผู้อื่น เป็นต้น

ในด้านของโครงการที่จัดทำถือว่าอยู่ในเกณฑ์ดี เนื่องจากมีทั้งข้อดีและข้อเสีย ข้อดีคือได้ทำการพัฒนา ออกแบบ และสร้าง โปรแกรมด้วยตัวเอง ตั้งแต่การวิเคราะห์ความต้องการ ออกแบบ จนถึงเขียนโปรแกรมต่างๆด้วยตัวเอง ซึ่งข้อเสียหรือข้อที่ควรปรับปรุงคือ โปรแกรมยังไม่สมบูรณ์ และยังขาดความสามารถบางอย่างที่ผู้ใช้งานต้องการ ไม่สามารถพัฒนาแล้วเสร็จในบางฟังก์ชันดังนั้นจึงควรต้องมีการปรับปรุงต่อไป

5.2 แนวทางการแก้ไขปัญหา

5.2.1 ปัญหาและอุปสรรค

5.2.1.1 มีความรู้เกี่ยวกับ Adobe Flash และ Action Script 2.0 ไม่เพียงพอ

แนวทางการแก้ไขปัญหาคือ มีการหาความรู้เกี่ยวกับเรื่อง Adobe Flash และ Action Script 2.0 ให้มากขึ้นทั้งจากทางอินเทอร์เน็ต หรือสอบถามพี่ที่ปรึกษา

5.2.1.2 การวิเคราะห์ ออกแบบลอจิกของเกมไม่ชัดเจน และมักเกิดปัญหามั้ย่อยๆ จึงทำให้มีการเปลี่ยนแปลงเกมบ่อย

แนวทางการแก้ไขปัญหาคือ มีการคิดและออกแบบมากขึ้น โดยหาแนวทางการแก้ปัญหา ณ จุดที่เราผิดพลาดบ่อยๆ ถ้าคิด ณ จุดนั้นจริงๆ ก็จะมีการสอบถามข้อมูลจากพี่ เพื่อให้ได้ข้อเสนอแนะที่ดีๆ

5.2.1.3 ขาดประสบการณ์การทำงานจริงจึงทำให้มองภาพรวมของโปรแกรมไม่ชัดเจน ทำให้เกิดปัญหาในการออกแบบ

แนวทางการแก้ไขปัญหาคือ มั่นศึกษาโปรแกรม และหาความรู้ใหม่ๆเพื่อนำมาประยุกต์ใช้กับงานจริง

5.2.1.4 ขาดความสามารถในการตัดสินใจอันเด็ดขาดในการคิดและแก้ไขปัญหามากกว่าที่เกิดขึ้นในการทำงาน

แนวทางการแก้ไขปัญหา พยายามคิดหรือแสดงการตัดสินใจของตนเองให้ชัดเจน

5.2.1.5 มีความเกี่วข้องแก่ใจในการสอบถามข้อมูลต่างๆเพื่อแสวงหาความรู้ใหม่ๆมากขึ้นไป จนทำให้การทำงานล้าช้า

แนวทางการแก้ไขปัญห พยายามแสวงหาความรู้ใหม่ๆด้วยตนเองเพิ่มเติม และรับผิดชอบต่องานต่างๆมากขึ้น

5.3 ข้อเสนอแนะจากการดำเนินงาน

5.3.1 ควรศึกษาเกี่ยวกับ Adobe Flash และ Action Script 2.0 เพิ่มเติม และฝึกการใ้งานบ่อยๆ

5.3.2 ควรศึกษากระบวนการออกแบบ และลอจิกของเกมเพิ่มเติม

5.3.3 ฝึกฝนการเขียนโปรแกรมให้มากขึ้น มันทำแบบทดสอบหรือหาโปรแกรมใหม่ๆมาฝึกฝน

5.3.4 มันคิดค้นลอจิกหรือสูตรคำนวณต่างๆใหม่ๆอยู่เสมอๆ เพื่อความง่ายต่อการทำงานครั้งต่อไป

5.3.5 มันคิดค้นไอเดียใหม่ๆเพื่อให้โปรแกรมของเรามีอะไรแปลกใหม่อยู่เสมอ

5.3.6 ควรแสวงหาความรู้ใหม่ๆตลอดเวลา ไม่ว่าจะเป็นเรื่องเรียน หรือความรู้ทั่วไป

เอกสารอ้างอิง

1. กำพล ลีลาภรณ์ ,2551 ,Flash Actionscript ,พิมพ์ครั้งที่ 1 ,บริษัท เอชเอ็น กรัฟ จำกัด ,สำนักพิมพ์ บริษัท โปรวิชั่น จำกัด ,480
2. บุญเลิศ อรุณพิบูลย์ ,2550 ,เริ่มต้นกับ Flash ,69
3. นางสาวฤดีมาศ วงษ์พิมพ์ , 2552 , สื่อการเรียนรู้ด้วยตนเองผ่านระบบอินเทอร์เน็ต เรื่องการใช้โปรแกรม Microsoft Office 2007 , บริหารธุรกิจบัณฑิต สาขาวิชาคอมพิวเตอร์ธุรกิจ มหาวิทยาลัยมหาสารคาม
4. bubbleball, 2011, <http://flashinw.com/> [online],Available:URL[2011, August 20]



ภาคผนวก





ภาคผนวก ก.

การใช้โปรแกรม Flash

การใช้โปรแกรม Flash

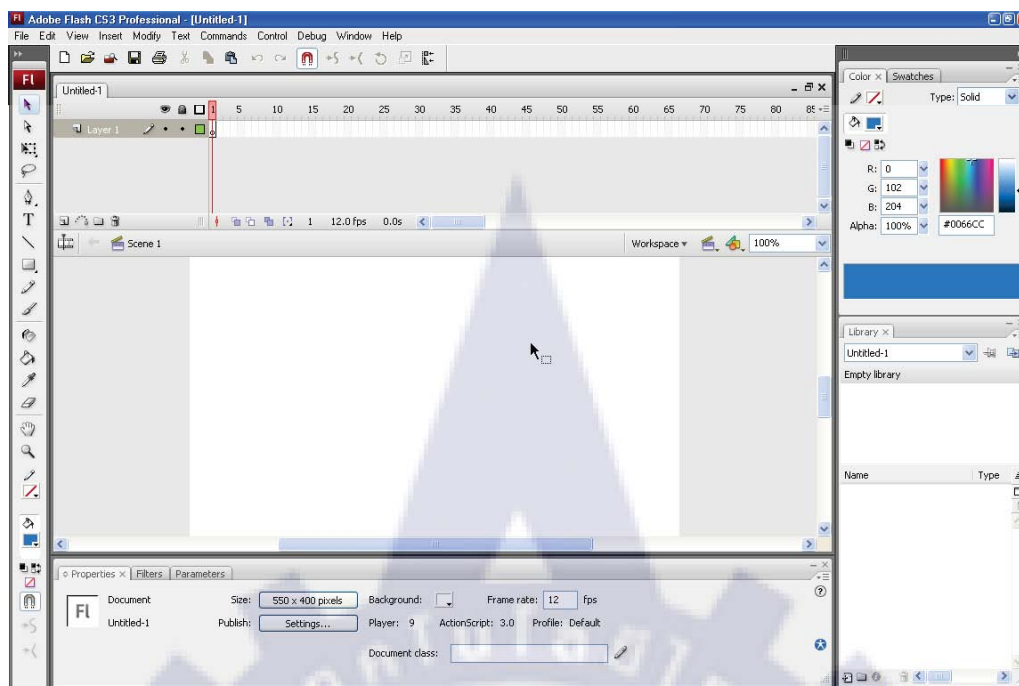
การเรียกใช้งานโปรแกรม Flash มีหลักการคล้ายๆ กับการเรียกโปรแกรมทั่วไปของระบบปฏิบัติการวินโดวส์ โดยเริ่มจากการคลิกปุ่ม Start จากนั้นเลื่อนไปคลิกที่รายการ Program, Adobe Flash CS3 Professional รอสักครู่จะปรากฏหน้าต่างการทำงาน ซึ่งมีโหมดการทำงานให้เลือกได้หลายลักษณะได้แก่

- การเปิดไฟล์จากคำสั่ง Open a Recent Item
- การสร้างงานผลงานจากรายการ Create New
- การสร้างผลงานแม่แบบ Create from Template



ภาพที่ ก.1 หน้าต่างโปรแกรม Adobe Flash CS3

เมื่อคลิกเลือกการสร้างผลงานใหม่ของ Flash จากรายการ Create New Flash File จะปรากฏส่วนประกอบจอภาพการทำงานดังนี้



ภาพที่ ก.2 หน้าการทำงานหลักของโปรแกรม

แถบเครื่องมือหลัก (Main Toolbar)

แถบเครื่องมือควบคุมการทำงานหลักของโปรแกรม เช่น การสร้างไฟล์ใหม่, การเปิดไฟล์, การคัดลอกข้อมูลเป็นต้น โดยสามารถควบคุมให้แสดง หรือไม่ต้องแสดงโดยคลิกเลือกคำสั่ง Window, Toolbars, Main



ภาพที่ ก.3 แถบเครื่องมือหลัก

แถบเครื่องมือ (Toolbox)

กลุ่มเครื่องมือสร้างงานและจัดการวัตถุ ประกอบด้วยปุ่มเครื่องมือย่อยต่างๆ สามารถเปิด/

ปิดด้วยคำสั่ง Window, Tools โดยสามารถแบ่งเครื่องมือเป็นหมวดๆ ได้ 5 หมวด คือ เครื่องมือ
หมวดเลือกวัตถุ (Selection) เครื่องมือหมวดวาดภาพ (Drawing) เครื่องมือจัดแต่งวัตถุ (Modify)
เครื่องมือควบคุมมุมมอง (View) และเครื่องมือควบคุมสี (Color)



ภาพที่ ก.4 แถบเครื่องมือ (Toolbox)

Document Tab

ส่วนควบคุมเอกสาร สามารถคลิกเพื่อสลับเปลี่ยนจอภาพเอกสาร



ภาพที่ ก.5 Document Tab

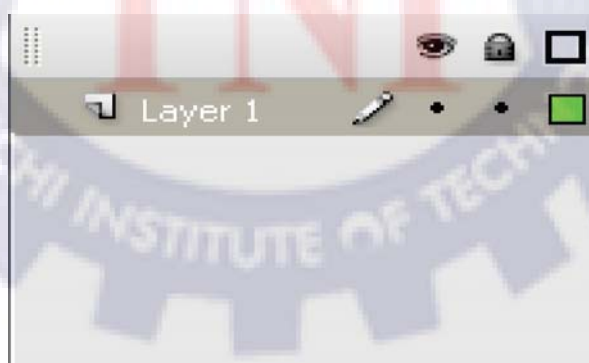
Timeline & Layer

Timeline เป็นส่วนสำคัญที่ทำหน้าที่ควบคุมการนำเสนอผลงาน สามารถเปิด/ปิดด้วยคำสั่ง Window, Timeline



ภาพที่ ก.6 Timeline

Layer ส่วนควบคุมการสร้างชั้นวัตถุ เพื่อให้การควบคุมวัตถุแต่ละชั้น มีอิสระ และสะดวกต่อการแก้ไข ปรับแต่ง



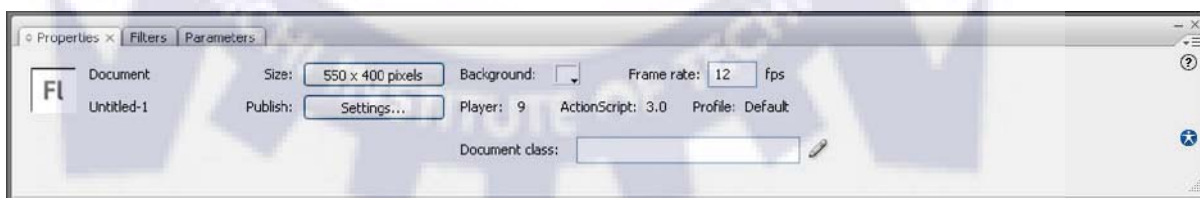
ภาพที่ ก.7 Layer

Stage & Workspace



ภาพที่ ก.8 ภาพ stage และการวางวัตถุบน Stage

จากรูปตัวอย่างนี้ เมื่อสั่งนำเสนอผลงาน แสดงว่าจะกำหนดให้เรื่องไม่ต้องแสดงผลทันทีบนจอภาพ เพราะเรื้อถูกนำไปวางไว้ในพื้นที่สีเทา เมื่อนำมาทำเป็น Movie ให้เรื้อวิ่งผ่านจอภาพไปอีกด้านหนึ่ง ก็จะปรากฏเรื้อวิ่งผ่านจอจากด้านซ้ายไปด้านขวาของจอ ตามช่วงเวลาที่กำหนด Panel หน้าต่างเล็กๆ ที่ทำหน้าที่แสดงคำสั่งควบคุมย่อยต่างๆ ของโปรแกรม โดยจะปรากฏรายการคำสั่งในเมนู Window



ภาพที่ ก.9 หน้าต่าง Properties



ภาพที่ ก.10 รายการคำสั่งในเมนู Window

ควบคุม Panel

Panel เป็นจอภาพเล็กๆ ที่แสดงฟังก์ชันการทำงานเฉพาะอย่าง เช่น การทำงานเกี่ยวกับสีจะควบคุมด้วย Color Mixer หรือ Color Swatches การจัดตำแหน่งวัตถุต่างๆ ควบคุมด้วย Align Panel เป็นต้น การเรียกใช้หรือเปิด/ปิด Panel จะใช้คำสั่ง Window แล้วตามด้วยชื่อ Panel นั้นๆ ทำงานกับไฟล์ Flash

โปรแกรม Flash สามารถสร้างผลงานได้ทั้งภาพนิ่ง ที่เรียกว่า (Still Image) และภาพเคลื่อนไหว ซึ่งจะเรียกว่า Movie ทั้งภาพนิ่ง และภาพเคลื่อนไหว จำเป็นต้องเริ่มต้นจากไฟล์ Flash ต้นฉบับ ที่มีส่วนขยายเป็น .fla จากนั้นจึงบันทึกเป็นไฟล์ภาพใช้งาน สำหรับภาพนิ่ง สามารถเลือกบันทึกได้หลายฟอร์แมต เช่น

ตารางที่ ก.1 นามสกุลของไฟล์ภาพนิ่ง

ประเภทไฟล์	ส่วนขยาย
Adobe Illustrator	.ai
GIF Image	.gif
Bitmap	.bmp
AutoCAD DXF Image	.dxf
Enhanced Metafile	.emf
EPS 3.0	.eps
JPEG Image	.jpg
PICT	.pct
PNG Image	.png
Windows Metafile	.wmf

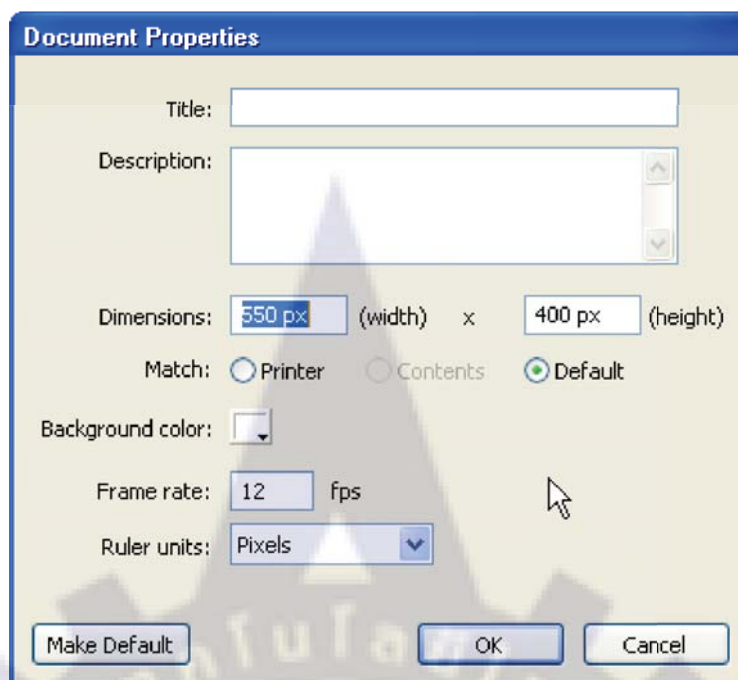
ภาพเคลื่อนไหว หรือ Flash Movie สามารถบันทึกในฟอร์แมตที่พร้อมใช้งาน ได้ดังนี้

ตารางที่ ก.2 นามสกุลของไฟล์ภาพเคลื่อนไหว

ประเภทไฟล์	ส่วนขยาย
Flash Movie	.swf
Animation GIF Image	.gif
QuickTime	.mov
Windows AVI	.avi
Execute File	.exe

สร้างไฟล์ใหม่

ไฟล์ Flash ที่สร้างใหม่ทุกครั้ง ควรกำหนดคุณสมบัติให้เหมาะสม ด้วยคำสั่ง Modify Document เช่น ความกว้าง/ความสูงของ Stage ลักษณะสีพื้นของ Stage เป็นต้น



ภาพที่ ก.11 Document Properties

เปิดไฟล์

การเปิดไฟล์ภาพใช้คำสั่ง File, Open... หรือคลิกปุ่ม Open  จาก Main Toolbar นอกจากนี้ยังสามารถเลือกไฟล์ที่เคยเปิดได้จากเมนูคำสั่ง File, Open Recent การคืนสู่สภาพเดิม (Revert) ไฟล์ที่กำลังแก้ไข ถ้าต้องการคืนกลับสู่สภาพก่อนการแก้ไข ใช้คำสั่ง File, Revert ปิดไฟล์

ไฟล์ที่สร้าง หรือเปิดอยู่ หากต้องการปิดไฟล์ สามารถใช้คำสั่ง File, Close หรือ File, Close All ทั้งนี้ไฟล์ที่ยังไม่ได้ผ่านการบันทึก โปรแกรมจะแสดงกรอบเตือน ถ้าต้องการบันทึกไฟล์ก่อนปิด ก็คลิกปุ่ม Yes เพื่อเข้าสู่โหมดการบันทึกไฟล์ แต่ถ้าต้องการปิดไฟล์โดยไม่บันทึกก็คลิกปุ่ม No หรือคลิกปุ่ม Cancel เพื่อยกเลิกการปิดไฟล์ กลับสู่จอภาพสร้างงานตามปกติ

บันทึกไฟล์

ภาพที่วาดที่สร้างเสร็จแล้ว หรือปรับแต่งแก้ไขแล้ว ควรบันทึกไฟล์เก็บไว้ทุกครั้ง โดยไฟล์ต้นฉบับจะได้ส่วนขยายเป็น .fla การบันทึกไฟล์สามารถใช้คำสั่ง File, Save... หรือ File, Save

As... จุดสังเกตว่าไฟล์ได้ผ่านการบันทึกแล้วหรือไม่ ก็ดูได้จากชื่อไฟล์ใน Title Bar หากมีเครื่องหมาย * แสดงว่ายังไม่ผ่านการบันทึก



ภาพที่ ก.12 หัวข้อ โปรแกรมที่ยังไม่มีการ Save

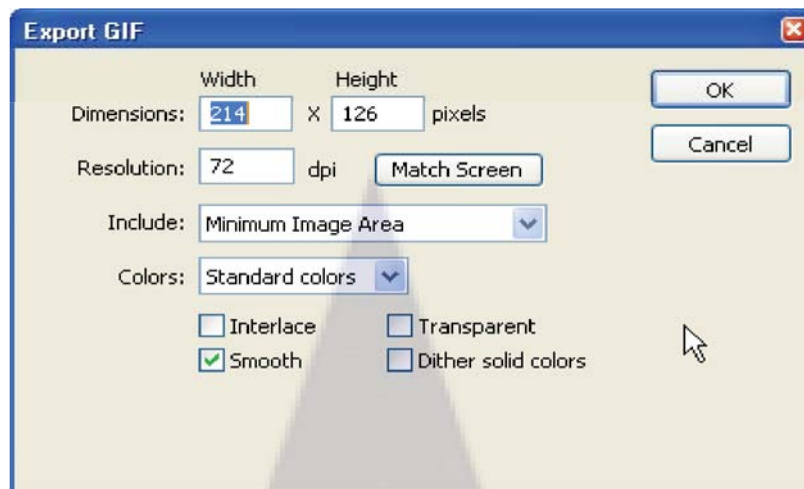
การบันทึกเป็นภาพนิ่งใช้งาน

เนื่องจากไฟล์ .fla เป็นไฟล์ต้นฉบับ ไม่สามารถนำไปใช้งานได้ ก่อนนำไฟล์ภาพที่สร้างด้วย Flash ไปใช้งาน จำเป็นต้องบันทึกในฟอร์แมตที่เหมาะสม สำหรับภาพนิ่ง ให้เลือกคำสั่ง File, Export, Export Image... ฟอร์แมตของภาพนิ่ง ก็เป็นรายการที่ควรนำมาพิจารณาประกอบ เช่น ถ้าต้องการเป็นภาพประกอบสื่อสิ่งพิมพ์ แนะนำให้เลือกเป็น .bmp หรือ .ai แต่ถ้าต้องการใช้บนเว็บไซต์ ก็ควรเลือกเป็น .gif, .jpg หรือ .png โดยมีหลักในการพิจารณา ดังนี้

- ภาพโครงร่าง หรือภาพที่มีการใช้สีแบบ Solid ให้เลือกเป็น GIF Format
- ภาพที่มีการใช้สีแบบไล่โทน หรือมีการใช้สีจำนวนมาก ให้เลือกเป็น JPEG Format

การบันทึกในฟอร์แมต GIF

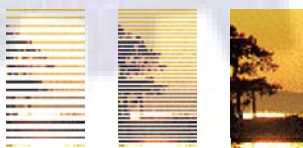
การบันทึกภาพวาดในฟอร์แมต GIF ทำได้โดยเลือกคำสั่ง File, Export, Export Image... เลือกรายการ Save as Type เป็น GIF Image (*.GIF)



ภาพที่ ก.13 การบันทึกในฟอร์แมต GIF

รายการเลือกของ GIF Format ได้แก่

- Dimension กำหนดขนาดของภาพ
- Resolution กำหนดความละเอียด มีค่าเท่ากับ 72 dpi
- Include เลือกรูปแบบการบันทึกพื้นที่รอบภาพ กรณีที่ต้องการบันทึกเฉพาะพื้นที่ที่มีภาพเท่านั้น ให้เลือกเป็น Minimum Image Area โปรแกรมจะไม่นำพื้นที่รอบภาพมาบันทึกด้วย แต่ถ้าเลือกเป็นรายการ Full Document Size จะเป็นการบันทึกเท่ากับขนาดที่ระบุจริงในรายการ Dimension
- Colors เลือกจำนวนค่าสีที่เหมาะสมกับภาพ ดังนั้นหากบางภาพมีการใช้สีน้อย ก็สามารถระบุจำนวนสีที่เหมาะสมได้
- Interlace เลือกเมื่อภาพที่วาดมีขนาดโตกว่า 200 pixel เพื่อกำหนดให้ภาพแสดงผลแบบโครงร่างก่อน แล้วค่อยๆ ชัดขึ้นเมื่อเวลาผ่านไป

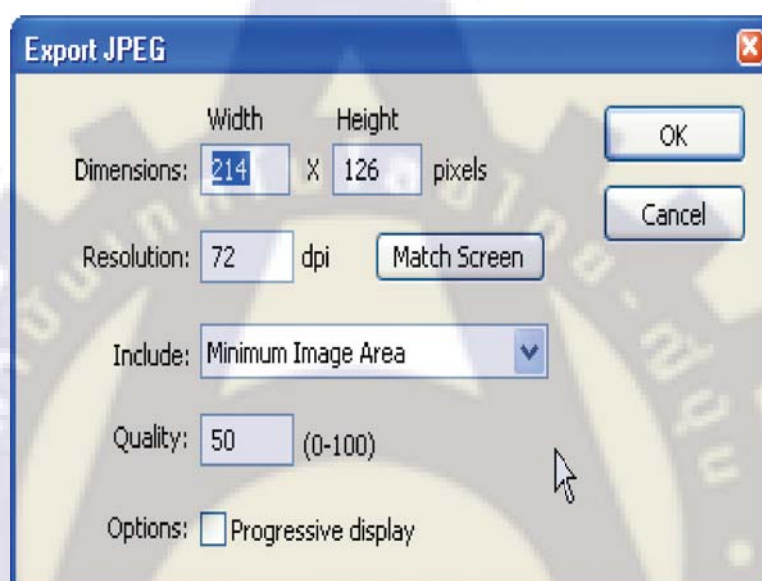


ภาพที่ ก.14 ตัวอย่างการทำงานแบบ Interlace

- Transparent เลือกเพื่อกำหนดให้ภาพมีลักษณะของพื้นแบบโปร่งใส
- Smooth เลือกให้ภาพมีลักษณะขอบกระด้าง หรือขอบมน
- Dither solid colors เลือกลักษณะการเกลี่ยสีที่มีลักษณะใกล้เคียงกัน

การบันทึกในฟอร์แมต JPEG

การบันทึกภาพวาดในฟอร์แมต JPEG ทำได้โดยเลือกคำสั่ง File, Export, Export Image... เมื่อเลือกไดร์ฟ/โฟลเดอร์ และตั้งชื่อไฟล์ภาพ ให้เลือกรายการ Save as Type เป็น JPEG Image (*.jpg) แล้วคลิกปุ่ม Save จะปรากฏรายการเลือกค่าควบคุม ดังนี้

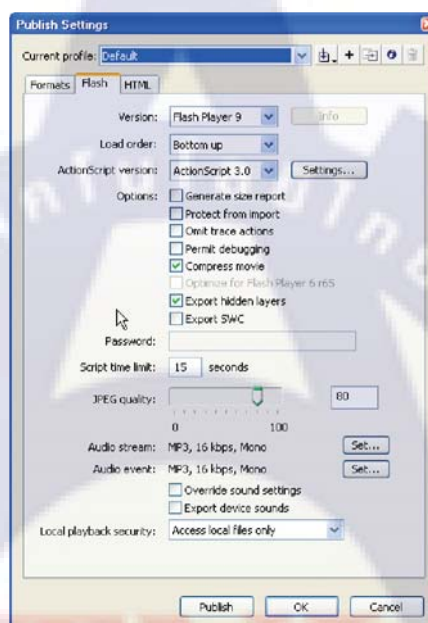


ภาพที่ ก.15 การบันทึกภาพวาดในฟอร์แมต JPEG

- Dimension กำหนดขนาดของภาพ
- Resolution กำหนดความละเอียด มีค่าเท่ากับ 72 dpi
- Include เลือกรูปแบบการบันทึกพื้นที่รอบภาพ กรณีที่ต้องการบันทึกเฉพาะพื้นที่ที่มีภาพเท่านั้น ให้เลือกเป็น Minimum Image Area โปรแกรมจะไม่นำพื้นที่รอบภาพมาบันทึกด้วย แต่ถ้าเลือกเป็นรายการ Full Document Size จะเป็นการบันทึกเท่ากับ ขนาดที่ระบุจริงในรายการ Dimension
- Quality คุณภาพของภาพ กรณีที่นำไปใช้กับเอกสารเว็บ ควรกำหนดไว้ที่ 60 – 90 แต่ถ้าต้องการบันทึกเป็นภาพต้นฉบับเพื่อไปตกแต่งด้วยโปรแกรมอื่นต่อไป ควรกำหนดเป็น 100

- Progressive เลือกเมื่อภาพที่วาดมีขนาดโตกว่า 200 pixel เพื่อกำหนดให้ภาพแสดงผลแบบโครงร่างก่อน แล้วค่อยๆ ชัดขึ้นเมื่อเวลาผ่านไป คล้ายๆ กับคุณสมบัติ Interlace ของ GIF การบันทึกเป็นภาพเคลื่อนไหว

การบันทึกผลงานของ Flash เป็นภาพเคลื่อนไหว หรือ Flash Movie สามารถเลือกได้สองคำสั่ง คือ File, Export, Export Movie... โดยมีหน้าต่างการบันทึก และการเลือกฟอร์แมตคล้ายกับการบันทึกภาพนิ่งที่แนะนำไปก่อน นอกจากนี้ยังสามารถเลือกได้จากคำสั่ง File, Publish Settings... ซึ่งเป็นคำสั่งที่นิยมเลือกใช้มากกว่า กรณีที่เป็นภาพเคลื่อนไหว โดยจะปรากฏหน้าต่างทำงาน ดังนี้



ภาพที่ ก.16 การบันทึกผลงานของ Flash เป็นภาพเคลื่อนไหว
เลือกฟอร์แมตที่ต้องการใช้งาน

- ใช้งานในเครือข่ายอินเทอร์เน็ต ให้คลิกเลือกรายการ Flash และ HTML
- สร้าง Movie ในฟอร์แมต QuickTime ให้เลือกรายการ QuickTime ซึ่งจะได้ไฟล์ Movie ที่มีส่วนขยายเป็น .mov
- สร้าง Movie ที่สามารถนำเสนอได้ทันที โดยไม่ต้องอาศัย Plug-Ins ใดๆ ให้เลือกรายการ Windows Projector ซึ่งจะได้ไฟล์ที่มีส่วนขยาย .exe หรือเลือกรายการ Macintosh Projector สำหรับการนำเสนอบนเครื่องคอมพิวเตอร์ Macintosh

เมื่อเลือกรูปแบบไฟล์ที่ต้องการแล้ว ให้คลิกปุ่ม Publish โปรแกรมจะแปลงงานบน Stage เป็น Movie ตามฟอร์แมตที่เลือก โดยใช้ชื่อไฟล์เดียวกับไฟล์ Flash ต้นฉบับ

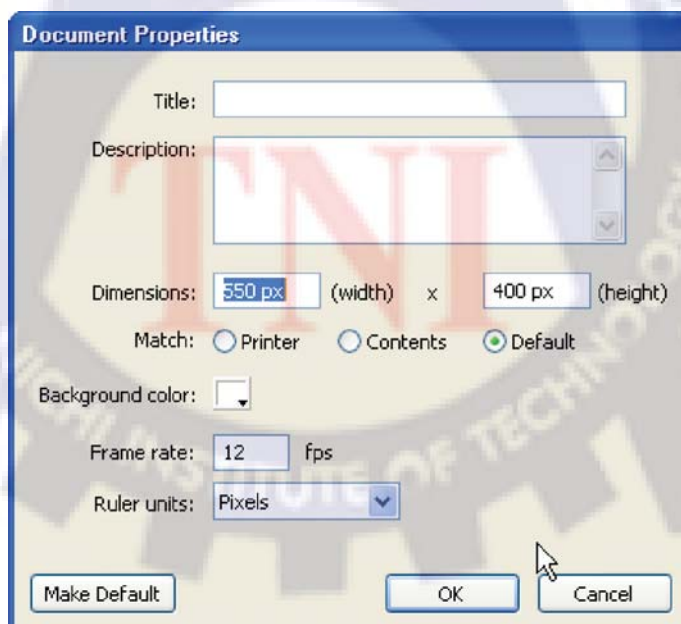
ทำงานกับ Stage

Stage เป็นชื่อเรียกพื้นที่สร้างภาพกราฟิกของ Flash นับเป็นพื้นที่สำคัญในการสร้างสรรค์งานกราฟิก หรือสร้าง Movie มีลักษณะเป็นพื้นที่สี่เหลี่ยมสีขาวล้อมด้วยกรอบสีเทา ภาพกราฟิกวัตถุใดๆ สามารถวางได้บนพื้นที่สีขาว และสีเทา แต่เมื่อสั่งนำเสนอผลงาน เฉพาะภาพกราฟิกหรือวัตถุที่วางบนพื้นที่สีขาวเท่านั้น ที่จะแสดงผล

ขนาดของ Stage

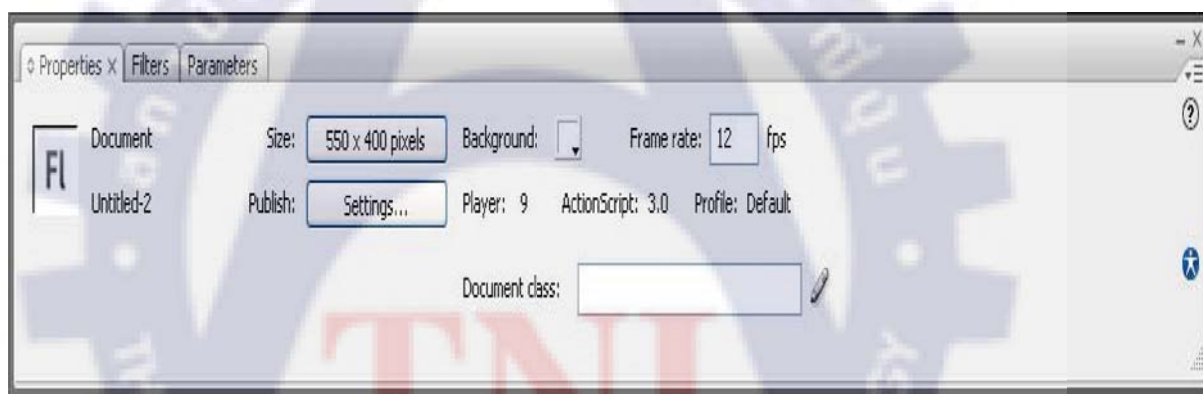
ขนาดของ Stage จะหมายถึงพื้นที่การแสดงผลของ Movie นั้นเอง ดังนั้นก่อนสร้างงานควรกำหนดขนาดของ Stage ให้เหมาะสมและตรงกับการใช้งานจริง การกำหนดขนาดของ Stageกระทำได้โดย

- เลือกคำสั่ง File, New... เพื่อเข้าสู่โหมดการสร้าง Movie
- คลิกเลือกคำสั่ง Modify, Document...



ภาพที่ ก.17 การปรับขนาดของ Stage

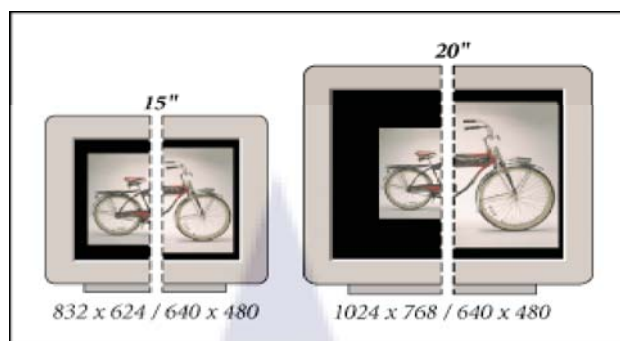
- กำหนดค่าความกว้าง ความสูงของ Stage (หน่วยปกติจะเป็น pixel) จากรายการ Dimensions:
 - o กรณีที่มีข้อมูลบน Stage สามารถคลิกปุ่ม Match: Printer หรือ Contents เพื่อให้ Flash ปรับขนาดให้เหมาะสมกับข้อมูลโดยอัตโนมัติ
- เลือกสีพื้นของ Stage จากตัวเลือก Background color:
- รายการ Frame rate: เป็นหน่วยวัดการสร้างภาพเคลื่อนไหว หมายถึงในเวลา 1 วินาทีจะต้องมีภาพกี่เฟรม จึงจะทำให้ภาพเคลื่อนไหวมีความสมจริง ถ้ากำหนดน้อยเกินไปก็จะทำให้มีอาการกระตุก และหากเร็วเกินไปก็จะทำให้สูญเสียรายละเอียด การนำเสนอบนเว็บ ค่า 12 เฟรมต่อวินาทีเป็นค่าที่เหมาะสมที่สุด แต่การสร้างภาพยนตร์เช่น QuickTime หรือ AVI จะต้องกำหนดไว้ที่ 24 เฟรมต่อวินาที
- Ruler units: หน่วยวัดของไม้บรรทัด และหน่วยวัดการสร้างวัตถุต่างๆ แนะนำให้ใช้หน่วยเป็น pixels การกำหนดขนาดและคุณสมบัติอื่นๆ ของ Stage ยังสามารถเลือกได้จาก Properties Panel โดยต้องอยู่ในโหมดการใช้เครื่องมือ Selection 



ภาพที่ ก.18 Properties Panel

ข้อแนะนำการกำหนดขนาดของ Stage

วัตถุประสงค์หลักของการพัฒนา Flash Movie ก็คือการนำไปใช้เป็นองค์ประกอบของเว็บไซต์ ในปัจจุบันหน้าเว็บไซต์ที่ถือว่าเป็นมาตรฐานจะมีขนาดที่เหมาะสมกับการแสดงบนจอภาพขนาด 1024 x 768 pixels



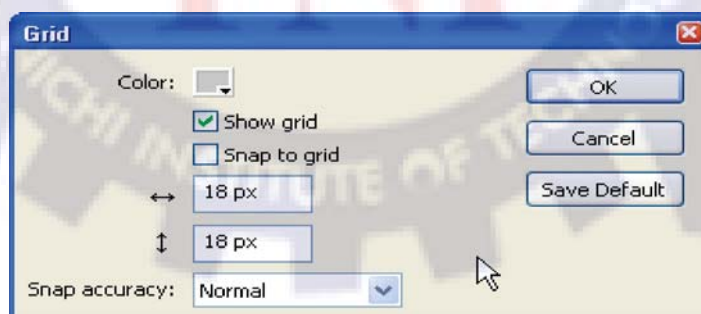
ภาพที่ ก.19 ภาพตัวอย่างการปรับขนาด Stage ให้เหมาะสมกับหน้าจอ

ดังนั้นขนาดของ Stage ควรกำหนดให้มีความกว้าง ความสูงสัมพันธ์กับเลย์เอาต์ของหน้าเว็บ

Ruler, Grid, Guides

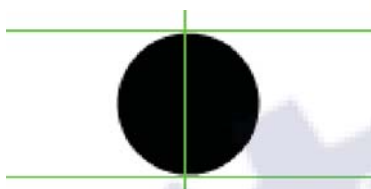
Ruler, Grid, Guides เครื่องมือช่วยกำหนดขอบเขต และวางตำแหน่งการสร้างกราฟิกบน Stage โดย

- Ruler แถบไม้บรรทัดจะปรากฏที่ขอบด้านซ้าย และด้านบนของ Stage สามารถ เปิด/ปิด ได้จากคำสั่ง View, Rulers
- Grid มีลักษณะเป็นตารางหมากรุก ที่แบ่งเป็นช่องเล็ก ช่วยในการกำหนดตำแหน่งในการสร้างย่อ/ขยาย หรือเคลื่อนย้ายวัตถุบน Stage สามารถเปิด/ปิด ได้จากคำสั่ง View, Grids, Show Grids โดยปกติขนาดของช่องตารางจะมีค่าเท่ากับ 18×18 pixels ซึ่งปรับแต่งได้จากคำสั่ง View, Grids, Edit Grid...



ภาพที่ ก.20 การแสดงข้อมูล , ปรับขนาด Grid

Guide มีลักษณะเป็นเส้นตรงที่ผู้ใช้สามารถกำหนดตำแหน่ง เพื่อช่วยในการกระยะต่างๆ ช่วยในการวาดภาพ เคลื่อนย้ายตำแหน่งลักษณะเดียวกับกริด แต่มีความอิสระมากกว่า โดยการทำงานจะต้องอยู่ในสถานะการเปิดใช้งาน Ruler ก่อนเสมอ จากนั้นนำเมาส์ไปชี้ในแถบไม้บรรทัด (ด้านใดก็ได้) กดปุ่มเมาส์ค้างไว้ เมื่อลากเมาส์จะปรากฏเส้นตรงสีเขียววาง ณ ตำแหน่งที่ปล่อยเมาส์ การปรับย้ายตำแหน่งเส้นไกด์ กระทำได้โดยคลิกเลือกเครื่องมือ Move  แล้วนำเมาส์ไปชี้ที่เส้นไกด์ เพื่อปรับย้ายตำแหน่ง 

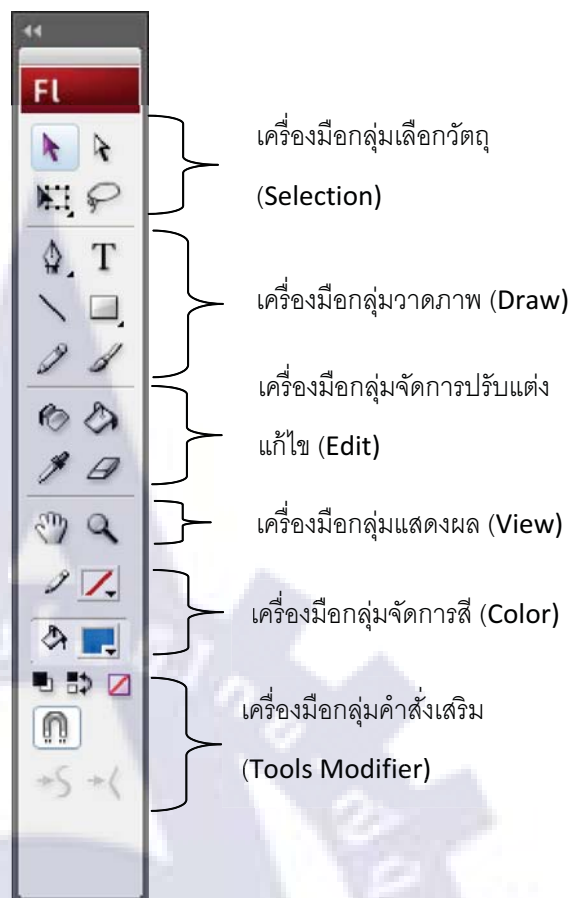


คำสั่งที่เกี่ยวข้องกับแถบไม้บรรทัด, Grids และไกด์ คือคำสั่ง Snap to... ซึ่งมีหลายคำสั่ง เช่น

- Snap to Grids ช่วยให้การวาด/สร้างวัตถุ, การย่อขยาย หรือย้ายตำแหน่งอิงเส้นกริดที่กำหนดไว้
- Snap to Guides ช่วยให้การวาด/สร้างวัตถุ, การย่อขยาย หรือย้ายตำแหน่งอิงเส้นไกด์ที่กำหนดไว้
- Snap to Objects ช่วยให้การวาด/สร้างวัตถุ, การย่อขยาย หรือย้ายตำแหน่งอิงจุดกึ่งกลาง (Center Point) ของวัตถุ

การใช้เครื่องมือของ Flash

การสร้างภาพกราฟิกต่างๆ สามารถใช้เครื่องมือกราฟิกจากชุดเครื่องมือ Toolbox โดยมีเครื่องมือหลายกลุ่ม และเปิด/ปิด Toolbox ได้จากเมนูคำสั่ง Window, Tools



ภาพที่ ก.21 เครื่องมือของ Flash

Tools Modifier

Tools Modifier เป็นส่วนขยายคำสั่งของเครื่องมือปกติ โดยรายการในส่วนนี้จะปรับเปลี่ยนไปตามเครื่องมือที่เลือก ดังนั้นการเลือกเครื่องมือใดๆ ควรตรวจสอบคำสั่งย่อยของเครื่องมือนั้นๆ จากส่วนควบคุมนี้ด้วยเสมอ



Tool modidier ของเครื่องมือ



Tool modidier ของเครื่องมือ

มุมมองจอภาพ

Stage เป็นพื้นที่หลักของการสร้างงาน ดังนั้นเครื่องมือชุดแรกที่ควรรทราบ ก็คือเครื่องมือในกลุ่ม View ซึ่งจะใช้ในการควบคุม Stage เป็นหลัก เช่น การย่อ/ขยาย Stage การเลื่อน Stage เป็นต้น Hand tool  เป็นเครื่องมือที่ใช้เลื่อนและปรับขนาดของ Stage

o Drag & Drop เพื่อเลื่อน Stage

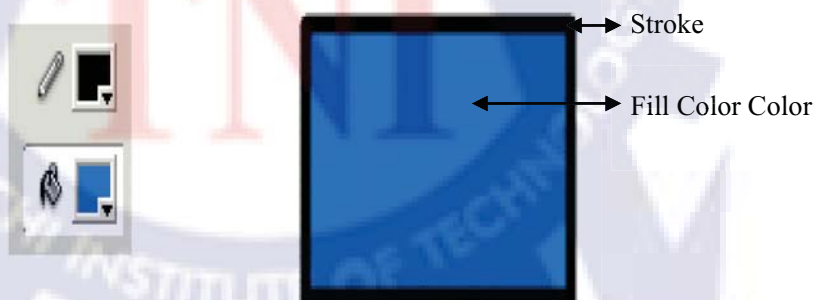
o ดับเบิลคลิกที่  กำหนดขนาดของ Stage ให้มีขนาดพอดีกับความกว้างของจอภาพ (Fit on screen) Zoom tool  เป็นเครื่องมือปรับขนาดของ Stage คลิกที่  จะปรากฏรายการเลือกย่อยที่ Modifier คลิกเลือกรูปแบบการย่อ  หรือขยาย  จากนั้นนำมาสัมผัสคลิกบน Stage

o ดับเบิลคลิกที่  เพื่อกำหนดให้ Stage มีขนาดเป็น 100% อย่างรวดเร็ว

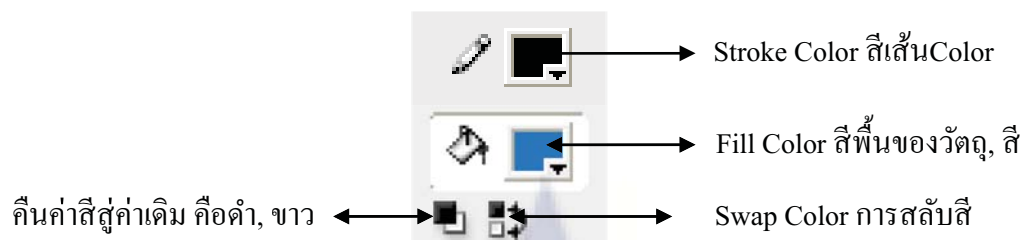
การควบคุม Stage ยังสามารถใช้ปุ่ม Zoom  ที่ปรากฏอยู่มุมบนขวาของ Stage หรือเลือกจากเมนูคำสั่ง View, Zoom in/Zoom out/Magnifier ได้เช่นเดียวกัน

ทำงานกับสี (Color)

วัตถุใน Flash จะประกอบด้วยส่วนประกอบอย่างน้อยๆ 2 ส่วน ได้แก่ พื้นของวัตถุ (Background หรือ Fill) และเส้นขอบวัตถุ (Stroke) แต่ละส่วนสามารถแสดงผลด้วยสีที่แตกต่างกันได้ เช่น รูปสี่เหลี่ยมที่มีเส้นขอบสีดำ และพื้นข้างในเป็นสีน้ำเงิน ดังนั้นการทำงานเกี่ยวกับสี จึงเป็นการทำงานที่จะต้องเกี่ยวข้องตลอดเวลา โดยอาศัยชุดเครื่องมือเลือกสี จาก Toolbox

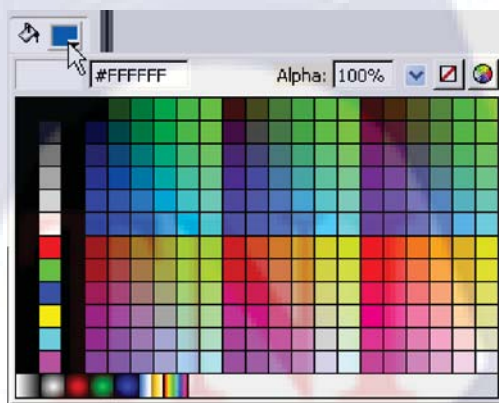


ภาพที่ ก.22 รายละเอียดเกี่ยวกับส่วนควบคุมสีใน Toolbox



ภาพที่ ก.23 รายละเอียดเกี่ยวกับส่วนควบคุมสีใน Toolbox

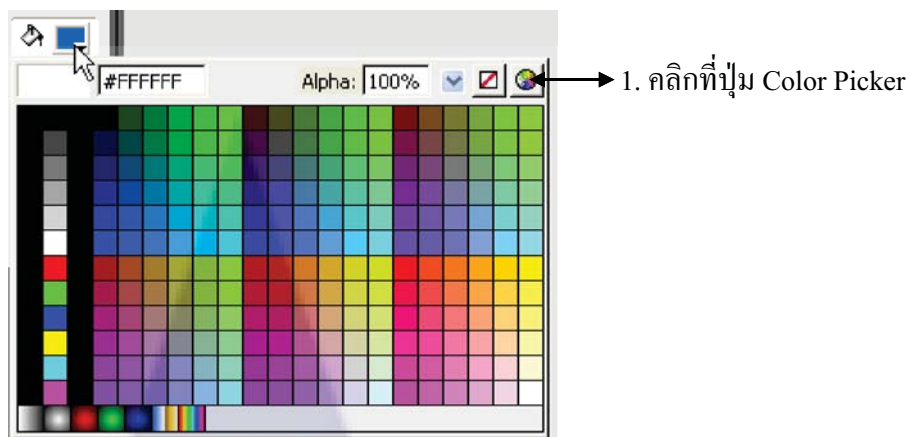
รายการเลือก No Color จะแสดงผลเมื่อคลิกเลือกวาดสี่เหลี่ยม หรือวาดวงกลม ดังนั้นก่อนเลือกสีให้กับการวาดสี่เหลี่ยม, วงกลม ควรคลิกเลือกเครื่องมือวาดสี่เหลี่ยม หรือวาดวงกลมก่อนที่จะคลิกปุ่มเลือกสี ซึ่ฟ เป็นรายการสำคัญมาก และมักจะเป็นรายการที่เข้าใจผิด เช่น ถ้าต้องการวาดวงกลมไม่มีสีพื้น หลายๆ ท่าน จะใช้วิธีการเลือก Fill Color ให้กับสีของ Background เช่น ถ้า Background เป็นสีขาว ก็จะเลือก Fill Color เป็นสีขาว ซึ่งมีความหมายที่ต่างไป



ภาพที่ ก.24 หน้าต่างสี

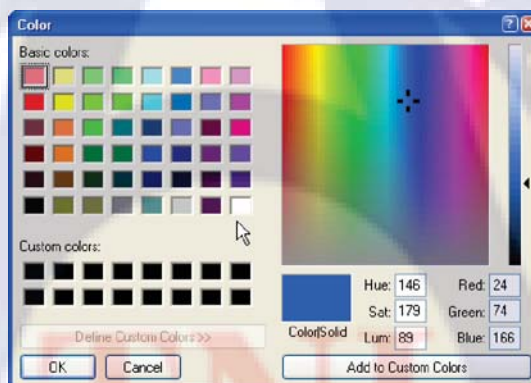
เพิ่มสี

การเพิ่มรายการสี สามารถทำได้โดยคลิกที่เครื่องมือเลือกสี (จะเป็น Fill หรือ Stroke) ก็ได้ แล้วคลิกที่ปุ่ม Color Picker ซึ่งปรากฏที่มุมบนขวาของจอภาพแสดงค่าสี



ภาพที่ ก.25 แสดงขั้นตอนการเลือกสี

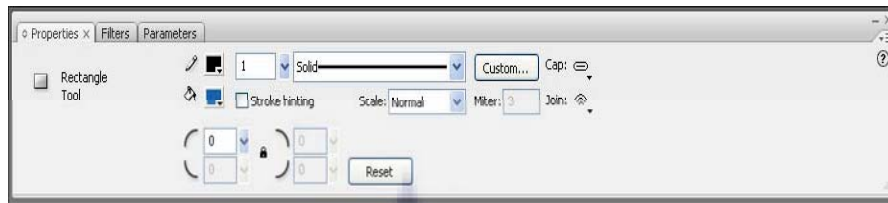
2.ปรากฏจอภาพผสมสีใหม่



ภาพที่ ก.26 แสดงขั้นตอนการผสมสีใหม่

3. คลิกในช่อง Custom colors
4. ผสมสีตามต้องการ
5. คลิกปุ่ม Add to Custom Colors

การเลือกสีให้กับกราฟิกต่างๆ ที่วาดด้วยเครื่องมือของ Flash นอกจากจะใช้ส่วนควบคุมสีที่กล่าวไปแล้ว ก็จะมีรายการเลือกสีใน Properties Panel ของเครื่องมือนั้นๆ

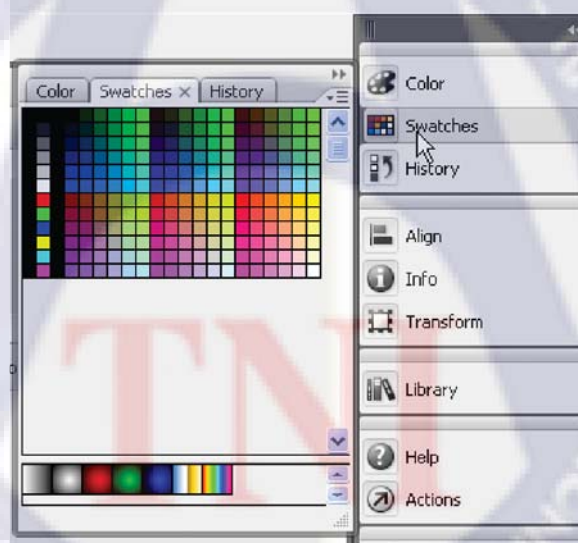


ภาพที่ ก.27 การใช้ Properties Panel ในการเลือกสี

แผงควบคุมสี (Color Panel)

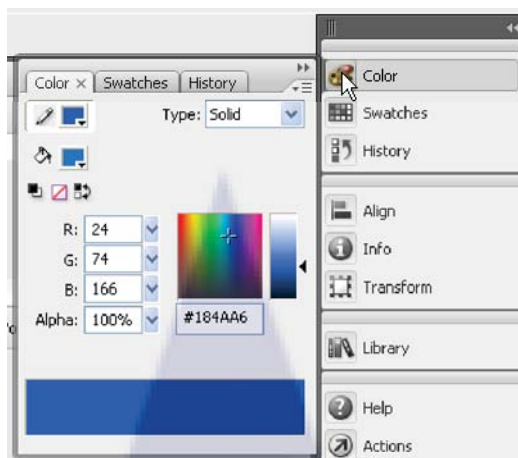
Color Panel เป็นการเพิ่มประสิทธิภาพของการทำงานเกี่ยวกับสี โดยเฉพาะในส่วนที่เป็น การไล่โทนสี (Gradient) เนื่องจากการสร้างชุดสีการไล่โทน ไม่สามารถทำได้จากส่วนควบคุมสี ปกติ Flash เตรียม Panel เกี่ยวกับสีไว้ 2 ชุดคือ

- Swatches ซึ่งมีการทำงาน/ใช้งานลักษณะเดียวกับ Toolbox



ภาพที่ ก.28 Swatches

Color มีส่วนเพิ่มเติมการใช้สีมากกว่าปกติ เช่น การทำสีแบบไล่โทนลักษณะต่างๆ, การใช้ ภาพกราฟิกมาเป็นพื้นของกราฟิก (Texture) รวมทั้งการปรับค่าความโปร่งใสของสี (Alpha)



ภาพที่ ก.29 ตัวอย่างการเลือกสีแบบ Fill Type

การเลือกรายการจาก Color มีรายการที่น่าสนใจ คือ Fill Type ซึ่งจะช่วยให้ผู้ใช้สามารถเลือกรูปแบบของสีได้หลากหลายลักษณะ เช่น สีพื้น (Solid Color), สีไล่โทนแบบเส้นตรง (Linear Gradient), สีไล่โทนแบบรัศมี (Radial Gradient) และการนำภาพจากภายนอกมาเป็นพื้นของวัตถุ (Bitmap Background)

ชุดสีแบบไล่โทน

- คลิกเลือกรายการ Fill Type เป็น Linear (ไล่โทนในแนวระนาบ) หรือ Radial (ไล่โทนในแนวรัศมี)



Linear



Radial

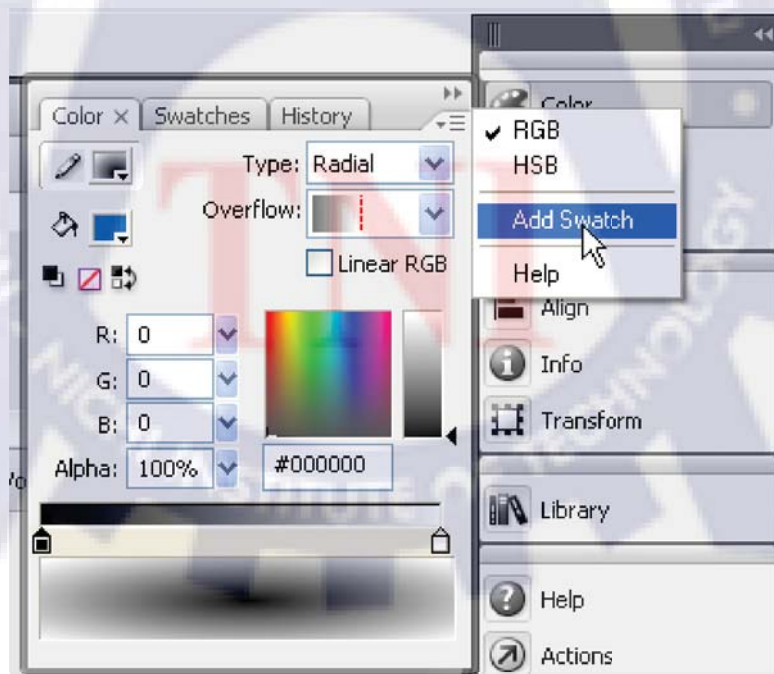
ภาพที่ ก.30 ตัวอย่างการเลือกสีแบบ Fill Type Linear หรือ Radial

- นำเมาส์ไปคลิกใต้ Gradient definition bar จะปรากฏ Gradient  Pointer กำหนดจำนวน Gradient Pointer  ตามต้องการ



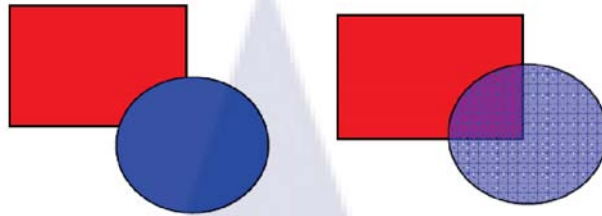
ภาพที่ ก.31 การกำหนดจำนวน Gradient Pointer

- ถ้าต้องการลบ Gradient Pointer ให้นำเมาส์ไปชี้ ณ Gradient Pointer ที่ต้องการลบ แล้วลากออก จาก Gradient definition bar
- กำหนดสีให้กับ Gradient Pointer โดยคลิกที่ Gradient Pointer ขึ้นที่ต้องการ จากนั้นคลิกเลือกสี จาก Current Color ทำซ้ำกับ Gradient Pointer ตำแหน่งอื่น
- สามารถเลื่อนปรับตำแหน่งของ Gradient Pointer โดยใช้หลัก Drag & Drop
- คลิกปุ่ม Color Mixer Option Menu แล้วเลือกคำสั่ง Add Swatch เพื่อเพิ่มสีที่กำหนดให้กับ โปรแกรม



ภาพที่ ก.32 การ Add Swatch

ความโปร่งใสของสีวัตถุ (Alpha)

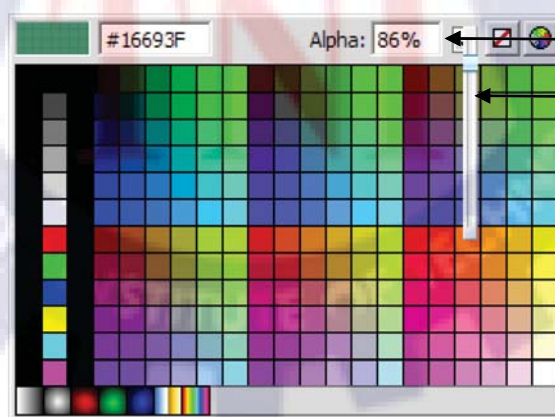


ภาพที่ ก.33 ตัวอย่างความโปร่งใสของวัตถุ

รูปวงกลมไม่ได้กำหนดค่าความโปร่งใส ก็จะซ้อนทับสีเหลี่ยมแบบไม่เห็นภาพด้านหลังแต่ ถ้ากำหนดวงกลมให้มีค่าโปร่งใส ก็จะทำให้สีของวงกลมมีลักษณะจาง และมองเห็นรูปด้านหลังได้

วิธีการปรับค่าความโปร่งใสของสีวัตถุ (Alpha)

ซึ่งเราสามารถเลือกปรับค่า ความโปร่งใสของสีวัตถุ (Alpha) ได้ใน แถบสี โดยค่า ความโปร่งใสของสีวัตถุ (Alpha) จะอยู่ที่ 0 -100 % ตามความทึบ-ใสของวัตถุ ซึ่งถ้าค่า ความโปร่งใสของสีวัตถุ (Alpha) ยิ่งน้อย ภาพก็จะยิ่งโปร่งใส



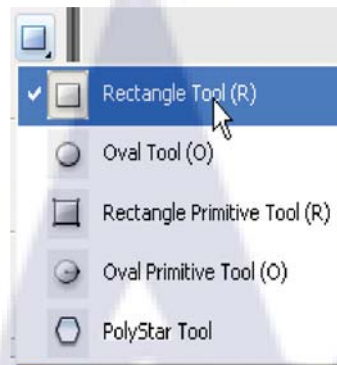
ปรับค่า Alpha ได้
ณ 2ตำแหน่งนี้

ภาพที่ ก.34 วิธีการปรับค่าความโปร่งใสของสีวัตถุ (Alpha)

เครื่องมือวาดภาพ

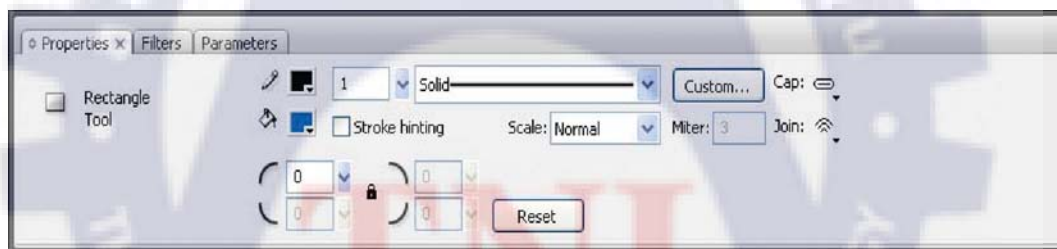
สี่เหลี่ยม, วงกลม, วงรี

- คลิกเลือกเครื่องมือ Rectangle  สามารถคลิกค้างไว้เพื่อเลือกรูปแบบการวาด



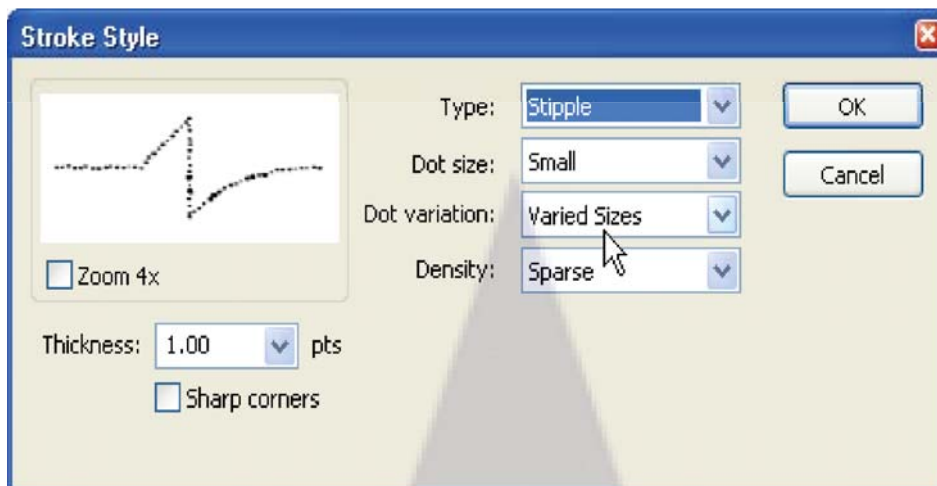
ภาพที่ ก.35 เครื่องมือวาดภาพ

- กำหนดสีพื้น, สีเส้นขอบ และลักษณะของเส้นขอบจาก Properties



ภาพที่ ก.36 สามารถกำหนดสีพื้น, สีเส้นขอบ และลักษณะของเส้นขอบจาก Properties

- o กำหนดลักษณะของเส้นขอบวงรี วงกลมได้โดยคลิกปุ่ม  แล้วปรับแต่งลักษณะของเส้นขอบได้จาก Stroke Style Dialog Box ดังภาพ



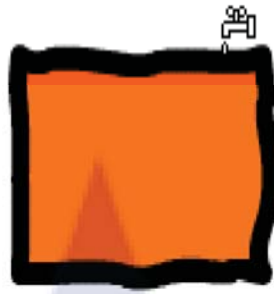
ภาพที่ ก.37 ปรับแต่งลักษณะของเส้นขอบได้จาก Stroke Style Dialog Box

- นำเมาส์มาคลิก ณ ตำแหน่งที่ต้องการวาดรูป กดปุ่มเมาส์ค้างไว้ แล้วลากเมาส์ เมื่อได้ขนาดและรูปทรงที่ต้องการ จึงปล่อยนิ้วจากเมาส์
- ถ้าต้องการภาพที่สมบูรณ์ ควรกดปุ่ม <Shift> ค้างไว้ ขณะลากเมาส์

ลบวัตถุด้วย Eraser Tool

วัตถุต่างๆ ที่วาดไว้แล้ว สามารถลบได้ 3 วิธี คือ

- ลบวัตถุทั้งหมดโดยดับเบิลคลิกที่ Eraser Tool 
- ลบเฉพาะ Fill หรือ Stroke ของวัตถุ โดย
 - o คลิกเลือก Eraser Tool
 - o คลิกที่ปุ่ม Faucet 
 - o นำเมาส์ไปคลิก ณ ตำแหน่งที่ต้องการ
 - ถ้าคลิก ณ ตำแหน่ง Stroke โปรแกรมจะลบเส้นขอบของวัตถุออกทั้งหมด
 - ถ้าคลิกในพื้นที่วัตถุ โปรแกรมก็จะลบพื้นที่วัตถุออกอย่างรวดเร็ว



ภาพที่ ก.38 ลบวัตถุด้วย Eraser Tool

• ลบวัตถุขึ้นที่ต้องการ โดย

o คลิกเลือก Eraser Tool

o เลือกโหมดการลบโดยคลิกที่ 

- Erase Normal

ลบในสภาวะปกติ คือ ลบทั้ง Stroke และ Fill

- Erase Fills

ลบเฉพาะส่วนที่เป็น Fill

- Erase Lines

ลบเฉพาะเส้นขอบ Stroke

- Erase Selected Fills

ลบเฉพาะส่วนที่เลือกไว้

- Erase Inside

ลบเฉพาะส่วนที่เป็น Fill แต่ถ้ามมีการลากผ่านตำแหน่งที่

เป็น Fill ว่างๆ จะไม่ลบให้

o เลือกลักษณะหรือขนาดของ Eraser โดยคลิกที่ 

o นำเมาส์มาคลิก หรือลากผ่าน ณ ตำแหน่งที่ต้องการลบ



Erase Normal

Fill

Stroke

Selection

Inside

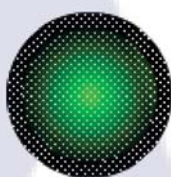
ภาพที่ ก.39 ลบวัตถุด้วย Eraser Tool

การเลือกวัตถุ (Selection)

วัตถุในความหมายนี้ ก็คือ รูปทรง รูปภาพ ภาพกราฟิกที่วาด หรือนำเข้ามาใช้งานใน Flash นั่นเอง การเลือกวัตถุ เป็นขั้นตอนสำคัญในการปรับเปลี่ยน แก้ไข แปลงวัตถุ โดยโปรแกรมเตรียมเครื่องมือเลือกวัตถุดังนี้



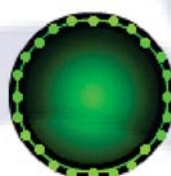
Selection Tool สำหรับเลือกวัตถุในสภาวะปกติ



ภาพที่ ก.40 ตัวอย่างการเลือกวัตถุในสภาวะปกติ



Subselection Tool สำหรับการเลือกวัตถุในโหมดจุดเชื่อมต่อ



ภาพที่ ก.41 ตัวอย่างการเลือกวัตถุในโหมดจุดเชื่อมต่อ



Lasso Tool สำหรับการเลือกวัตถุที่มีรูปทรงอิสระ หรือกำหนดขอบเขตการเลือกอิสระรวมทั้งการเลือกโดยใช้ค่าสีที่มีค่าเดียวกันหรือใกล้เคียงกัน



ภาพที่ ก.42 ตัวอย่างการเลือกวัตถุที่มีรูปทรงอิสระ

สิ่งสำคัญที่สุดในการเลือกวัตถุใน Flash ก็คือ อย่าลืมว่าวัตถุทุกชิ้นเกิดจาก “จุด” หลายๆ จุดมาประกอบรวมกัน และแต่ละวัตถุจะประกอบด้วยโครงสร้างอย่างน้อย 2 ส่วนคือ ส่วนที่เป็น “Fill” และส่วนที่เป็น “Stroke” ดังนี้



ภาพที่ ก.43 โครงสร้างของวัตถุที่ประกอบด้วยโครงสร้างย่อย 2 ส่วน



Stroke + Fill = Shape

ภาพที่ ก.44 รูปแสดง “จุด” อันเกิดจากการเลือกบางส่วนของวงกลม

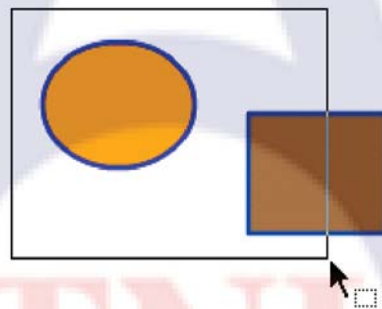
การเลือกวัตถุ หรือกลุ่มวัตถุด้วย Selection Tool

- คลิกเลือกเครื่องมือ Selection Tool
- เลือกวัตถุ โดยยึดหลักดังนี้
 - o เลือกเส้นขอบของวัตถุนำเมาส์ไปชี้ที่เส้นขอบวัตถุ แล้วคลิกหรือ ดับเบิลคลิก
 - o เลือกพื้นวัตถุนำเมาส์ไปชี้ที่พื้นวัตถุ แล้วคลิกหรือ ดับเบิลคลิก



ภาพที่ ก.45 การเลือกพื้นที่วัตถุ

- o เลือกวัตถุทั้งชิ้นนำมาใส่ไปชี้ที่วัตถุ แล้วดับเบิลคลิก
- o เลือกวัตถุทั้งชิ้นนำมาใส่ไปชี้ ณ ตำแหน่งมุมของวัตถุ แล้วลากกรอบสี่เหลี่ยมคลุมวัตถุ
- o เลือกวัตถุหลายๆ ชิ้นนำมาใส่ไปชี้ ณ ตำแหน่งมุมของวัตถุ แล้วลากกรอบสี่เหลี่ยมคลุมวัตถุ



ภาพที่ ก.46 การเลือกวัตถุโดยการลากคลุมพื้นที่

- o เลือกวัตถุหลายๆ ชิ้นคลิกวัตถุชิ้นที่ 1 กดปุ่ม Shift ค้างไว้ แล้วคลิกวัตถุชิ้นถัดไปเรื่อยๆ
- o เลือกวัตถุทุกชิ้นบน Workspace และ Stage กดปุ่ม Ctrl + A ยกเลิกการเลือกวัตถุ
- นำเมาส์ไปคลิกบนตำแหน่งว่างๆ ของ Stage หรือ เลือกเมนูคำสั่ง Edit, Deselect All ซ่อนการเลือกวัตถุ

บางครั้งผู้ใช้อาจจะต้องการซ่อนการเลือก (Selection) ไว้ชั่วคราวเพื่อทำงานหรือตรวจสอบผลให้ถูกต้องก่อนการใช้งานจริง ซึ่งกระทำได้โดยเลือกเมนูคำสั่ง View, Hide Edges หรือกดปุ่ม Ctrl + H

การยกเลิกคำสั่ง (Undo)

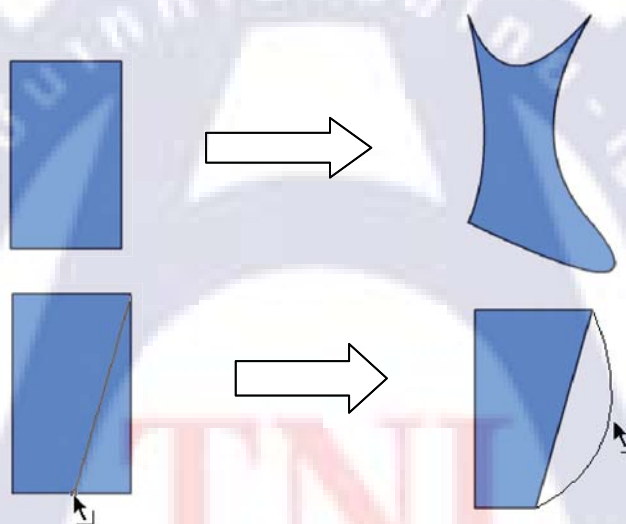
เมื่อทำงานใดๆ ผิดพลาด สามารถย้อนกลับ หรือยกเลิกคำสั่งนั้นๆ ได้โดยคลิกปุ่ม Ctrl + Z หรือ Edit, Undo... ซึ่งสามารถยกเลิกคำสั่งย้อนหลังได้มากกว่า 1 ครั้ง

ปรับแต่ง แก้ไขวัตถุ

จุดเด่นของการสร้างวัตถุด้วย Flash ก็คือวัตถุ หรือกราฟิกที่สร้างไว้แล้ว สามารถปรับแต่ง แก้ไข ปรับเปลี่ยนรูปทรง ขนาด และลักษณะได้ง่าย รวดเร็ว

เปลี่ยนรูปทรง

กราฟิกจาก Flash เกิดจากการรวมกันของ “จุด” ทำให้การปรับแต่ง เปลี่ยนรูปทรง กระทำได้ง่าย และสะดวก เพียงแต่ใช้หลักการ Drag & Drop ก็ทำให้รูปทรงพื้นฐาน เช่น วงกลม, วงรี, สี่เหลี่ยม เป็นสภาพเป็นรูปทรงอิสระอื่นๆ ได้ตามต้องการ เช่น



ภาพที่ ก.47 การเปลี่ยนรูปทรงของวัตถุ

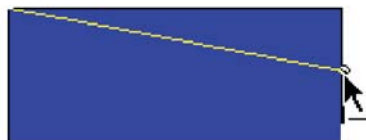
การเปลี่ยนรูปทรงของวัตถุ มีหลักการดังนี้

- วาดรูปทรงพื้นฐานที่ต้องการ จากตัวอย่างคือรูปสี่เหลี่ยม
- เลือกเครื่องมือ Selection 
- นำเมาส์ไปชี้บริเวณเส้นขอบของรูป สังเกตเมาส์จะมีรูปร่างเป็น  กดปุ่มเมาส์ค้างไว้เมื่อลากเมาส์รูปทรงจะถูกยืดหรือขยาย หรือหดตัวตามทิศทางการลากเมาส์
- นำเมาส์ไปชี้บริเวณมุมเหลี่ยมของรูป สังเกตเมาส์จะมีรูปร่างเป็น  กดปุ่มเมาส์ค้างไว้ เมื่อลากเมาส์รูปทรงจะถูกยืดหรือขยาย หรือบิดตัวตามทิศทางการลากเมาส์

ตัวอย่างการวาดจรวดแบบง่าย



1. วาดสี่เหลี่ยมผืนผ้า



2. เลือกเครื่องมือ Move เลื่อนไปชี้ที่มุมบนขวา

ของสี่เหลี่ยม ดึงเข้ามาตำแหน่งกึ่งกลางของด้านขวา ถ้าดึงแล้วสัดส่วนบิดเบี้ยวให้คลิกเพื่อตรึงตำแหน่งการบิดภาพ จากนั้น ทำซ้ำกับมุมล่างขวา



3. เลื่อนไปชี้ที่ขอบซ้าย แล้วดึงเข้ามาด้านใน ให้ได้เป็นรูปจรวดดัง ตัวอย่าง



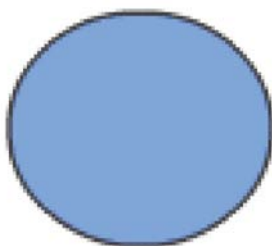
ภาพที่ ก.48 ตัวอย่างการเปลี่ยนรูปทรงวัตถุ

การย้ายวัตถุ

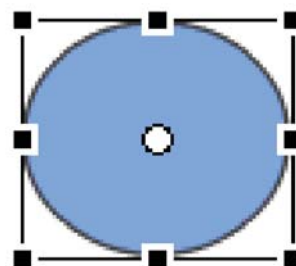
- คลิกเลือกเครื่องมือ Selection Tool
- เลือกวัตถุ แล้วลากเมาส์เพื่อย้ายวัตถุไปยังตำแหน่งใหม่
- เลื่อนวัตถุเป็นแนวเฉียง 45 องศา ให้กด Shift ด้วย
- เลื่อนวัตถุเป็นระยะทางสั้นๆ ครั้งละ 1 pixel ให้ใช้ปุ่มลูกศร
- เลื่อนวัตถุเป็นระยะทางสั้นๆ ครั้งละ 10 pixel ให้ใช้ปุ่มลูกศร พร้อมกับการกด Shift
- เลื่อนวัตถุไปยังตำแหน่งต่างๆ โดยระบุพิกัด ให้ระบุพิกัด x, y จาก Shape Properties ตำแหน่งมุมบนซ้ายของ Stage จะมีพิกัดเป็น 0, 0

การย่อ/ขยาย และปรับรูปทรงของวัตถุ

- เลือกวัตถุ
- คลิกปุ่มเครื่องมือ Free Transform Tool 



วัตถุสถานะปกติ



วัตถุในสถานะ Free Transform

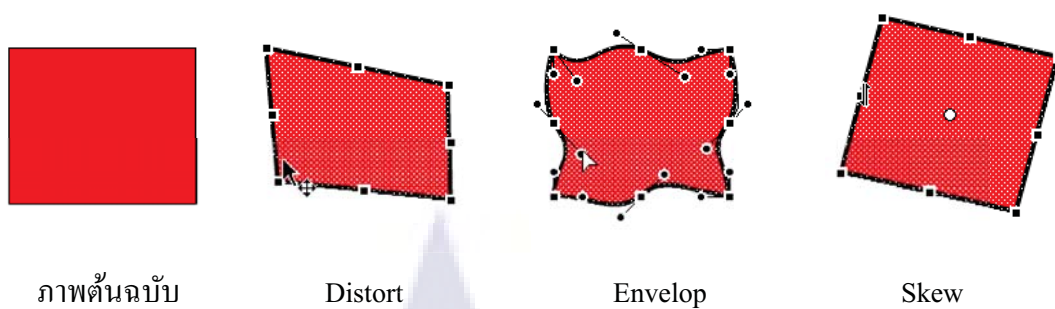
ภาพที่ ก.49 แสดงตัวอย่างวัตถุในสถานะ Free Transform

- นำเมาส์ไปชี้ที่มุม หรือขอบวัตถุ แล้ว Drag & Drop เพื่อปรับขนาด หรือรูปทรงตามต้องการ



ภาพที่ ก.50 แสดงตัวอย่างการปรับรูปทรงวัตถุ

- คลิกเลือกจากเมนูคำสั่ง Modify, Transform,... ซึ่งมีคำสั่งให้เลือกทั้งคำสั่งหมุนวัตถุ, คำสั่งกลับด้านของวัตถุ เป็นต้น



ภาพที่ ก.51 แสดงตัวอย่างการปรับรูปทรงวัตถุแบบต่างๆ

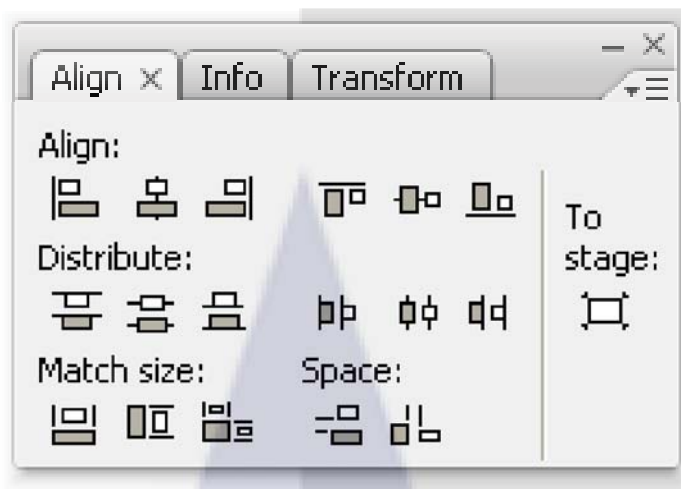
การจัดเรียงวัตถุ (Alignment)

บางครั้งวัตถุที่ต้องใช้งานจะมีมากกว่า 1 ชิ้น ซึ่งจำเป็นต้องจัดเรียงตำแหน่งให้อยู่ในแนวระดับเดียวกัน หากใช้เมาส์ลากแล้วปล่อยโอกาสที่จะตรงกัน หรือในแนวเดียวกันก็ทำได้ยาก Flash ได้เตรียมคำสั่ง Align เพื่อช่วยจัดเรียงวัตถุได้สะดวก รวดเร็ว



ภาพที่ ก.52 ตัวอย่าง Stage และวัตถุนอก Stage

1. วาดวัตถุให้อยู่นอก Stage
2. เลือกวัตถุด้วยเครื่องมือ Move
3. เปิด Align Panel



ภาพที่ ก.53 Align Panel

4. คลิก  ต่อด้วย  และ 

การคัดลอกลักษณะเส้นขอบวัตถุ

การคัดลอกลักษณะเส้นขอบวัตถุ จะช่วยให้การปรับแต่งแก้ไขวัตถุมากกว่า 1 ชิ้นทำได้ สะดวกกว่าการปรับเปลี่ยนทีละชิ้น ตัวอย่างมีวัตถุนบน Stage 3 ชิ้นลักษณะต่างๆ กัน ดังภาพ



ภาพที่ ก.54 คัดลอกลักษณะเส้นขอบวัตถุ

ต้องการให้วงกลม และสี่เหลี่ยมเส้นขอบดำ มีเส้นขอบเดียวกับสี่เหลี่ยมชั้นที่สอง ที่มีเส้นขอบเป็นจุดสีแดง หากต้องเลือกแล้วปรับเปลี่ยนทีละชิ้น ก็จะใช้เวลานานมาก วิธีการที่สะดวก คือ

คลิกเลือกเครื่องมือ Eyedropper



• นำเมาส์ซึ่งมีรูปร่างเป็น Eyedropper ไปชี้ที่เส้นขอบของสี่เหลี่ยมชั้นต้นฉบับ (สี่เหลี่ยมด้านขวาสุด) สังเกตได้ว่าเมาส์จะมีรูปร่างเป็น  แสดงว่าได้เลือกเส้นขอบวัตถุได้ถูกต้อง เมื่อคลิกเมาส์ 1 ครั้งเมาส์จะเปลี่ยนรูปร่างเป็น  แสดงว่าขณะนี้ Flash อยู่ในโหมด Ink Bottl  ซึ่งเป็นโหมดในการคัดลอกลักษณะเส้นขอบนั่นเอง

• นำเมาส์ที่เป็นรูปร่าง  ไปคลิกบนวัตถุชิ้นอื่นๆ ที่ต้องการปรับเปลี่ยนลักษณะเส้นขอบ

กลุ่มวัตถุ (Group)

ภาพกราฟิกหลายๆ ภาพ ได้จากภาพ หรือรูปทรงย่อยหลายๆ ชิ้นมารวมกัน เพื่อให้การปรับแต่งแก้ไข เคลื่อนย้ายกระทำได้สะดวก มักจะรวมภาพกราฟิก หรือรูปทรงทุกชิ้นที่เกี่ยวข้อง ให้เป็นกลุ่มเดียวกันโดยเลือกวัตถุทุกชิ้น แล้วเลือกเมนูคำสั่ง Modify, Group และเมื่อต้องการแยกกลับสู่สภาพเดิมก็เลือกคำสั่ง Modify, Ungroup



ภาพที่ ก.55 รูปการ์ตูนที่เกิดจาก Shape หลายชิ้นประกอบรวมกัน



ปัญหาจากการย้าย Shape



รูปที่ยังไม่ได้รวมกลุ่ม

ภาพที่ ก.56 รูปการ์ตูนที่เกิดจากปัญหาการย้าย Shape ต่างๆ



ภาพที่ ก.57 ภาพที่ผ่านการรวมกลุ่มแล้วจะมีเส้นขอบสีฟ้าล้อมรอบ

การแยกชิ้นส่วนของวัตถุ

วัตถุบางชิ้นที่เป็น Group หรือ Instance เมื่อจะต้องนำมาทำเป็น Movie แบบ Shape Tweening จำเป็นต้องแยกชิ้นส่วนของวัตถุให้อยู่ในสถานะ “จุด” ก่อนเสมอ ซึ่งทำได้โดยเลือกวัตถุก่อน จากนั้นจึงใช้เมนูคำสั่ง Modify, Break Apart หรือกดปุ่ม Ctrl + B



Flash MX



ภาพที่ ก.58 รูปภาพในสถานะกลุ่ม

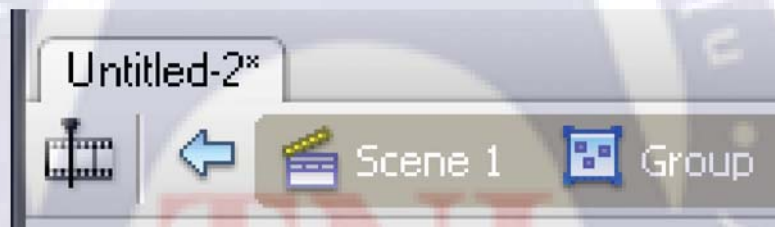


Flash MX

ภาพที่ ก.59 รูปภาพที่ผ่านการ Break Apart แล้ว

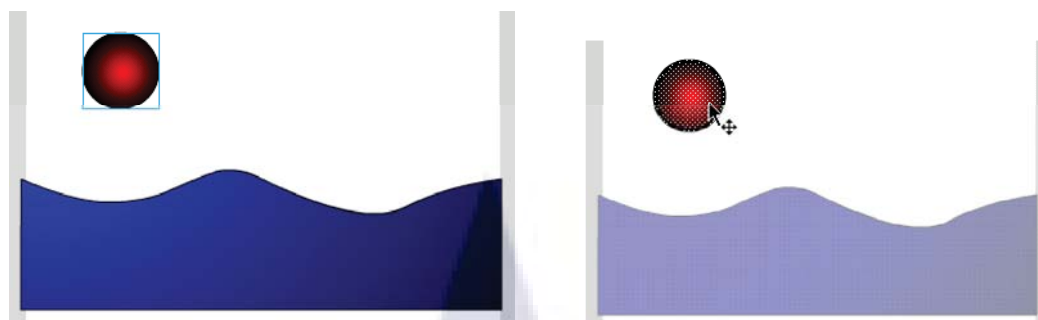
การปรับแต่งแก้ไขวัตถุที่เป็น Group

วัตถุที่อยู่ในสถานะ Group สามารถย่อ/ขยาย หมุนได้อิสระ แต่จะไม่สามารถปรับแต่งแก้ไขเกี่ยวกับสีได้ หากต้องการปรับเปลี่ยนสีของวัตถุที่เป็น Group จะต้องเข้าไปแก้ไขในโหมดจอภาพเฉพาะเกี่ยวกับ Group ซึ่งกระทำได้โดยการดับเบิลคลิกที่วัตถุนั้นๆ จอภาพจะเข้าสู่โหมดการแก้ไข Group ซึ่งสังเกตได้จาก



ภาพที่ ก.60 จอภาพเข้าสู่โหมดการแก้ไข Group

- Scene 1 หมายถึงจอภาพในโหมดหลัก
- Group หมายถึงจอภาพในโหมดแก้ไข Group จะสังเกตได้ว่าวัตถุชิ้นอื่น จะมีสีที่จางไป



จอภาพโหมดปกติ (Scene1)

โหมดแก้ไข Group จะพบว่าภาพคลื่นทะเลจะจางกว่าปกติ

ภาพที่ ก.61 ตัวอย่างการแก้ไข Group

เมื่อปรับแต่งแก้ไขวัตถุเรียบร้อยแล้ว ให้คลิกที่ Scene 1 เพื่อกลับสู่โหมดการทำงานปกติด้วยทุกครั้ง

Symbol และ Instance

การสร้าง Movie ด้วย Flash จำเป็นต้องเกี่ยวข้องกับ Symbol และ Instance ดังนั้นการศึกษาว่า Symbol และ Instance คืออะไร มีความสำคัญอย่างไร ใช้งานอย่างไร จึงเป็นสิ่งสำคัญมาก อย่างไรก็ตาม Flash มีการกำหนดประเภทของวัตถุไว้หลากหลายลักษณะ โดยสามารถแบ่งได้ดังนี้

- Dot หรือ Part เป็นส่วนย่อยที่สุดของวัตถุ มีลักษณะเป็นจุดเล็กๆ
- Shape เป็นวัตถุที่เกิดจาก Dot หลาย Dot มาผสมกัน โดยจะเรียกวัดที่สร้างด้วยเครื่องมือสร้างกราฟิกพื้นฐานว่า Shape ยกเว้น Text Tool
- Group เป็น Shape หลายๆ ชิ้นที่รวมกันเป็นชุดเดียว เกิดจากคำสั่ง Modify, Group รวมทั้งข้อความจาก Text Tool
- Symbol เป็นวัตถุที่ถูกแปลงสภาพเพื่อพร้อมสร้าง Movie เกิดจากการแปลงวัตถุต่างๆ รวมถึงการสร้าง Button, Movie Clip และการนำเข้าภาพจากแหล่งอื่นๆ สามารถตรวจสอบได้ว่าไฟล์ที่ทำงานมี Symbol อะไร ประเภทใด จาก Library Panel (เรียกด้วยคำสั่ง Window, Library)
 - o Graphic เป็น Symbol ภาพนิ่ง
 - o Button เป็น Symbol ปุ่มกดที่สามารถคลิกได้
 - o Movie Clip เป็น Symbol ภาพเคลื่อนไหว
- Instance เป็น Symbol ที่นำมาใช้งานบน Stage



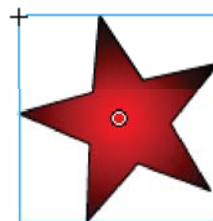
ภาพโหมด Shape/Dot

มีลักษณะเป็นจุด



โหมด Group

มีกรอบสี่ล้อมรอบ

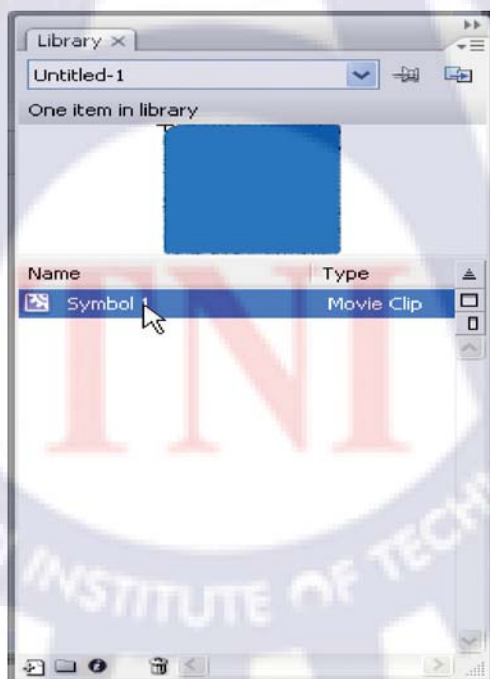


โหมด Instance/Symbol

มีจุดกึ่งกลางกลมและกากบาทกำกับ

ตรวจสอบ Symbol สำหรับไฟล์

เมื่อมีการเปิดไฟล์หรือสร้างไฟล์ใดๆ บางครั้งอาจจะไม่ทราบว่าไฟล์นั้นๆ มี Symbol ไດบ้าง โปรแกรมมีคำสั่งในการตรวจสอบ Symbol โดยเปิด Library Panel ด้วยเมนู Window, Library



ภาพที่ ก.62 Library ที่มี Symbol

แปลงวัตถุเป็น Symbol

การสร้างภาพเคลื่อนไหวแบบ Motion Tweening หรือการทำระบบโต้ตอบ จำเป็นต้องทำงานกับ Symbol ดังนั้นวัตถุหรือภาพกราฟิกใดๆ ก็ตาม จะต้องแปลงสภาพจาก Shape, Group หรือ Picture ให้เป็น Symbol ก่อนและจะต้องเลือก Behavior ของ Symbol ให้ตรงกับลักษณะการใช้งาน เช่น Symbol ที่ต้องการกำหนดให้คลิกได้เพื่อส่งงานใดๆ จะต้องกำหนดเป็น Button ภาพใดๆ ที่มีการเคลื่อนที่หรือการเคลื่อนไหว เช่น ล้อรถที่ต้องหมุน ควรกำหนดเป็น Movie Clip หรือ Symbol ที่แสดงเป็นเพียงภาพนิ่ง ก็เลือกเป็น Graphic เป็นต้น

การแปลงวัตถุให้เป็น Symbol มีหลักการดังนี้

- วาด/สร้าง หรือนำเข้าวัตถุ
- เลือกวัตถุ
- เลือกคำสั่ง Insert, Convert to Symbol... หรือกดปุ่ม <F8>



ภาพที่ ก.63 หน้าต่างการเปลี่ยนวัตถุเป็น Symbol

- ตั้งชื่อในช่อง Name แล้วเลือก Behavior ให้เหมาะสม
- คลิกปุ่ม OK เพื่อยืนยันการแปลงวัตถุเป็น Symbol

การสร้าง Symbol

นอกจากการแปลงวัตถุให้เป็น Symbol ด้วยวิธีการดังกล่าวข้างต้น ยังสามารถเข้าสู่โหมดการสร้าง Symbol ได้โดยตรง ซึ่งมีขั้นตอนดังนี้

- เลือกคำสั่ง Insert, New Symbol...
- ปรากฏจอภาพ Create New Symbol

- ตั้งชื่อ Symbol และเลือก Behavior จากนั้นคลิกปุ่ม OK ก็จะปรากฏจอภาพสร้าง Symbol ที่มีหน้าตาคล้ายกับ Stage เกือบทุกอย่าง เพียงแต่จะมีสัญลักษณ์ + อยู่กึ่งกลางจอ เป็นการระบุตำแหน่งพิกัด 0,0 เพื่อให้สะดวกต่อการวางตำแหน่ง หรือสร้างวัตถุ
- เมื่อสร้างวัตถุให้กับ Symbol เรียบร้อยแล้วก็คลิกที่ Scene 1 เพื่อกลับสู่สภาวะการทำงานปกติ



ภาพที่ ก.64 ขั้นตอนการกลับสู่ หน้าจอการทำงานปกติ

- รายละเอียดการสร้าง Movie Clip และ Button Symbol จะกล่าวในหัวข้อถัดไปการสร้าง Symbol ใดๆ ควรยึดตำแหน่งสัญลักษณ์ + เป็นจุดเริ่มต้นของวัตถุ ตำแหน่งพิกัด 0,0

Symbol และ Instance

วัตถุที่พร้อมสำหรับการสร้าง Movie ลักษณะต่างๆ รวมถึงการลงรหัส ActionScript เพื่อทำระบบโต้ตอบ (Interactive) เมื่อมีการสร้างและเก็บไว้ใน Library จะเรียกว่า Symbol แต่เมื่อนำ Symbol มาวางใช้งานบน Stage จะเรียกว่า Instance ทั้งนี้ Instance จะเปรียบเสมือนตัวแทนของ Symbol หากมีการแก้ไข Instance จะไม่มีผลกระทบต่อ Symbol แต่ถ้าแก้ไข Symbol ตัวแทนหรือ Instance ทุกตัวที่เกิดจาก Symbol นั้นๆ จะถูกแก้ไขตามไปด้วยอัตโนมัติ

การเรียกใช้ Symbol

การนำ Symbol จาก Library มาใช้งาน กระทำได้โดยการเปิด Library Panel แล้วคลิกเลือก Symbol ขึ้นที่ต้องการ ลากมาวางบน Stage และ Symbol จะเปลี่ยนสถานะเป็น Instance ทันที

เรียกใช้ Symbol สำเร็จรูปของโปรแกรม

โปรแกรมได้เตรียม Symbol สำเร็จรูปเพื่อสะดวกต่อการสร้างงาน โดยเรียกใช้ได้จากเมนูคำสั่ง Window, Other Panels, Common Libraries... ซึ่งมี Library สำเร็จรูปให้เลือกใช้ได้ 3 กลุ่ม เมื่อเลือก Symbol ที่ต้องการได้แล้ว ก็สามารถนำมาวางบน Stage โดยการนำเมาส์ชี้ที่ Symbol นั้น แล้วลากออกจาก Library มาวางบน Stage ได้เลย และ Symbol จะเปลี่ยนสถานะเป็น Instance ทันที

เรียกใช้ Symbol จากไฟล์อื่น

จุดเด่นของ Symbol คือ สามารถโอนใช้งานได้กับไฟล์อื่น หรือเรียกใช้ Symbol จากไฟล์อื่น โดยมีหลักการดังนี้

- เปิดไฟล์เอกสารที่ต้องการสร้างงาน
- เรียกใช้คำสั่ง File, Import, Open External Library...
- เลือกไฟล์ที่ต้องการนำ Library มาใช้งาน
- โปรแกรมจะเปิด Library มาให้เลือกทำงาน เมื่อเลือกใช้ Symbol ใดๆ นั้นจะถูกโอนไปยังไฟล์เอกสารปัจจุบันโดยอัตโนมัติ

แก้ไข Symbol

Symbol ที่สร้างไว้แล้ว สามารถปรับเปลี่ยนแก้ไขได้ โดย

- เปิด Library Panel ดับเบิลคลิกที่ชื่อ Symbol ขึ้นที่ต้องการแก้ไข หรือกรณีที่มี Instance ของ Symbol ปรากฏบน Stage ก็สามารถดับเบิลคลิกที่ Instance นั้นๆ ได้ทันที
- ปรากฏหน้าต่างการทำงานในโหมดแก้ไข Symbol โดยสังเกตได้ว่าตรงกลางจอภาพ มีสัญลักษณ์ + และปรากฏชื่อหน้าต่างเป็นชื่อ Symbol นั้นๆ ที่มุมบนซ้ายของ Stage
- แก้ไข Symbol เหมือนกับการแก้ไขวัตถุปกติทั่วไป
- เมื่อแก้ไขเรียบร้อยแล้ว สามารถกลับมาทำงานในโหมดปกติได้โดยคลิกที่ชื่อ Scene1 (หรือ Scene หมายเลขใดๆ ก็ได้ตามลักษณะของชิ้นงาน) หมายเหตุ การแก้ไข Symbol จะส่งผลต่อ Instance ที่เกิดจาก Symbol นั้นๆ ทุก Instance ที่นำมาใช้งานบน Stage

แก้ไข Instance

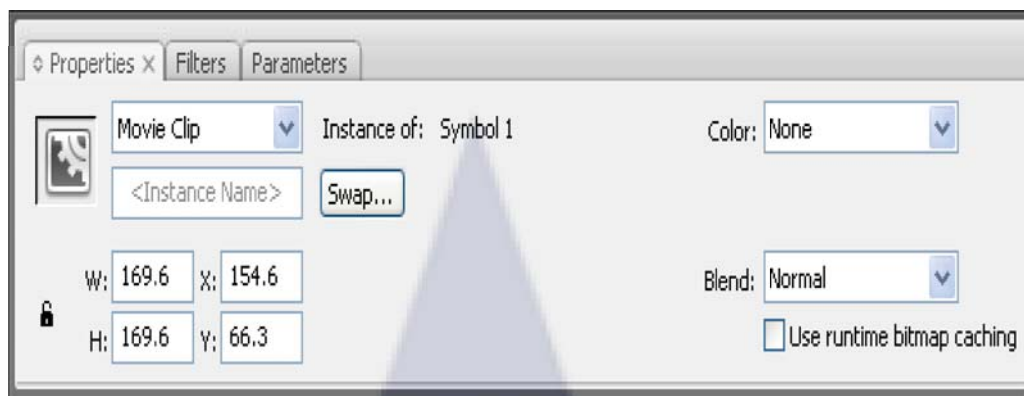
Instance เปรียบเสมือนวัตถุชิ้นหนึ่ง ซึ่งสามารถใช้คำสั่งจัดการวัตถุ มาดำเนินการปรับเปลี่ยนได้ โดยไม่ส่งผลกระทบต่อ Symbol ต้นฉบับ โดยมีวิธีการจัดการดังนี้

วิธีที่ 1

- คลิกเลือก Instance
- ใช้เครื่องมือ Arrow ปรับขนาดหรือหมุนวัตถุ

วิธีที่ 2

- คลิกเลือก Instance
- กำหนดลักษณะของสีที่ต้องการจาก Instance Properties

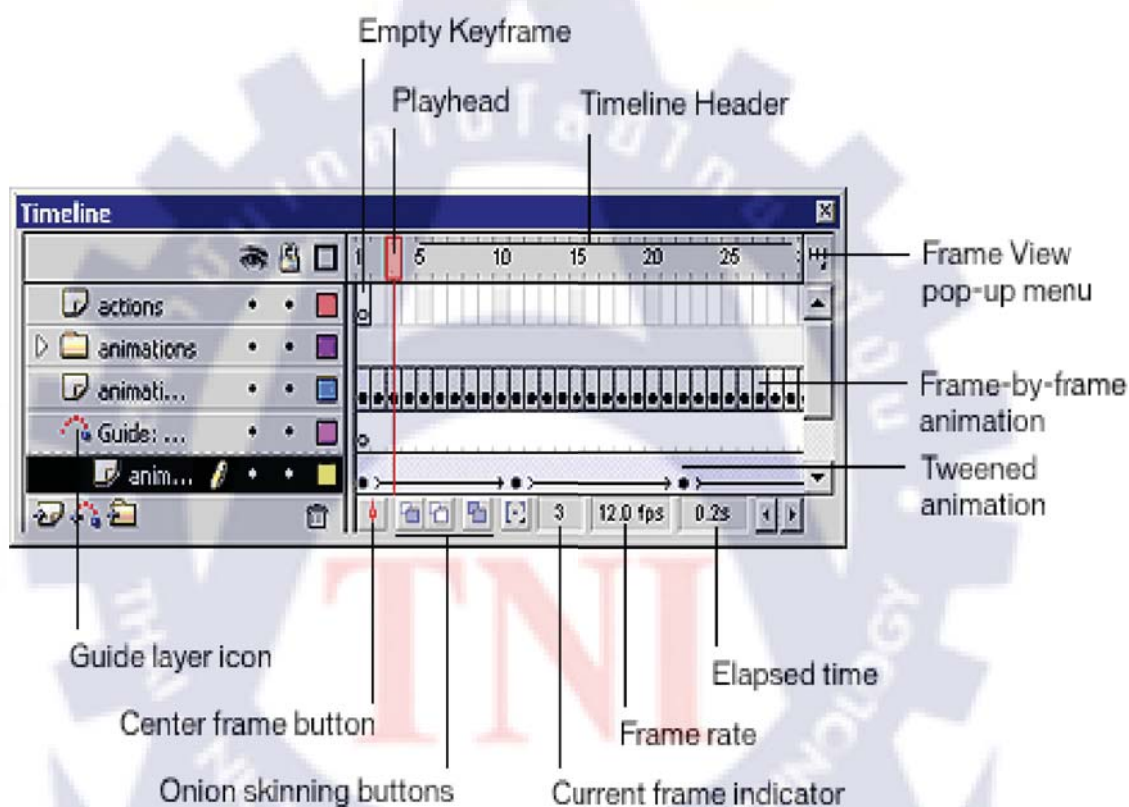


ภาพที่ ก.65 Instance Properties

- o Brightness ความสว่าง
- o Tint การปรับแก้ไขสี/ความโปร่งแสง
- o Alpha การปรับแก้ไขความโปร่งแสง
- o Advanced การปรับแก้ไขสีและค่าความโปร่งแสงแบบ Advanced

เฟรมและ Timeline

การสร้างภาพยนตร์ หรือภาพเคลื่อนไหวใด ก็คือการสร้างอริยาบถของภาพให้มีการเปลี่ยนแปลงไปตามช่วงเวลา เช่น การวาดการ์ตูนลงกระดาษแต่ละใบ ให้มีอริยาบถแตกต่างกันไป เมื่อนำการ์ตูนทุกภาพมาแสดงอย่าง รวดเร็ว ก็จะทำให้ภาพมีลักษณะเป็นภาพเคลื่อนไหว เมื่อมีการนำคอมพิวเตอร์มาใช้สร้างสรรค์ภาพยนตร์ หรือภาพเคลื่อนไหว เฟรมก็เปรียบเสมือนกระดาษแต่ละใบ ที่มีการวาดอริยาบถของการ์ตูนลงไป และ Timeline ก็คือช่วงเวลาที่ใช้ควบคุมการนำเสนอ ดังนั้น Frame และ Timeline จึงเป็นส่วนสำคัญอีกส่วนในการสร้างภาพเคลื่อนไหวของ Flash โดยมีลักษณะหน้าตาต่าง ดังนี้



ภาพที่ ก.66 ส่วนประกอบของ เฟรมและ Timeline

Timeline

เปรียบเทียบเวลาที่ใช้ในการสร้างภาพเคลื่อนไหว หรือ Movie โปรแกรมจะแบ่งช่วงให้หน่วยละ 5 ช่วงเวลา



ภาพที่ ก.67 หน้าต่าง Timeline

เฟรม

คือช่องเล็กที่เรียงต่อกันตามช่วงเวลา แต่ละช่องเปรียบเสมือนกระดาษ 1 แผ่นที่วาด Movie 1 อิริยาบถ และเปลี่ยนไปทุกๆ เฟรม เพื่อให้เกิดการเคลื่อนไหวของวัตถุ เช่น เฟรมช่องที่ 1 วาดวงกลมไว้ริมซ้ายสุดของ Stage เฟรมที่ 2 วาดวงกลมให้เลื่อนมาด้านขวานิดหนึ่ง และทำซ้ำๆ ไปเรื่อยๆ ซึ่งเป็นการสร้าง Movie ให้วงกลมวิ่งจากด้านซ้ายมาด้านขวานั้นเอง



ภาพที่ ก.68 หน้าต่าง เฟรม

Playhead



ภาพที่ ก.69 ตำแหน่งของเฟรมปัจจุบันที่จะปรากฏบน Stage

เฟรม และคีย์เฟรม (Frame & Keyframe)

คีย์เฟรม คือ การกำหนดตำแหน่งการสร้างภาพเคลื่อนไหวที่มีเนื้อหาภายในแตกต่างกันไป โดยตำแหน่งเฟรมใดที่ต้องการกำหนดให้มีสถานะเป็น Keyframe ให้ใช้คำสั่ง Insert, Timeline, Keyframe

เฟรม คือ การกำหนดตำแหน่งภาพที่ไม่มีการเปลี่ยนแปลง 'ไม่มีการเคลื่อนไหว' เช่นการทำฉากหลัง โดยตำแหน่งเฟรมใดที่ต้องการกำหนดให้มีสถานะเป็น Frame ให้ใช้คำสั่ง Insert, Timeline, Frame

ควบคุมเฟรม

การควบคุมเฟรม เช่น การใส่คีย์เฟรม หรือแทรกเฟรม นอกจากใช้คำสั่งจากเมนู Insert ยังสามารถใช้ Context Menu ซึ่งเป็นวิธีที่สะดวกกว่า โดยคลิกขวาในเฟรมที่ต้องการ แล้วเลือกคำสั่งที่ต้องการ

ลบเฟรม

- เลือกเฟรมที่ต้องการลบ (สามารถใช้การคลิกเมาส์ขวาในเฟรมที่ต้องการ)
- เลือกคำสั่ง Remove, Frames

Movie File

Movie คือ ภาพนิ่งหรือภาพเคลื่อนไหวที่สร้างด้วย Flash โดยมีลักษณะการสร้าง 3 รูปแบบ คือ

- Movie แบบ Frame by frame เป็น Movie ที่มีการเปลี่ยนแปลงของวัตถุตลอดเวลาและวัตถุมีการเปลี่ยนแปลงที่ไม่ต่อเนื่อง หรือเปลี่ยนแปลงทุกส่วนอย่างไม่สม่ำเสมอ

เป็นมาตรฐาน เช่น ดอกไม้บาน, การงอกของต้นไม้, การบินของนก, การเดินของสัตว์ การสร้างจะต้องสร้างต้นแบบงานหลายชิ้น การสร้าง Movie แบบนี้วัตถุที่นำมาสร้างจะอยู่ในสถานะใดก็ได้

- Movie แบบ Motion Tweening เป็น Movie ที่อาศัยหลักการเปลี่ยนรูปทรง ขนาด โดยการคำนวณของโปรแกรม การสร้างกระทำได้ง่าย โดยสร้างเฉพาะภาพแรก กับภาพสุดท้าย โปรแกรม Flash จะเติมขบวนการเปลี่ยนแปลงให้โดยอัตโนมัติ เช่น ลูกบอลตกจากที่สูงมากระทบพื้น, การบินของเครื่องบิน, การหมุนรอบวัตถุใดๆ การสร้าง Movie แบบนี้ วัตถุจะต้องอยู่ในสถานะ Symbol ก่อน

- Movie แบบ Shape Tweening เป็น Movie ที่วัตถุมีการเปลี่ยนแปลงรูปทรง รูปร่างจากรูปต้นฉบับไปอย่างสิ้นเชิง เช่น เปลี่ยนจากสี่เหลี่ยมเป็นวงกลม, เปลี่ยนจากไข่นก เป็นตัวนก การสร้าง Movie แบบนี้ วัตถุจะต้องอยู่ในสถานะ Break Apart

Motion Tween

การสร้าง Movie แบบ Motion Tween เป็น Movie ที่สร้างได้ง่าย รวดเร็ว โดยยึดหลักสร้างวัตถุในตำแหน่งเริ่มต้น และตำแหน่งสุดท้าย Flash จะสร้าง Effect การเคลื่อนที่ให้โดยอัตโนมัติ รวมทั้งผู้สร้างสามารถเลือกรูปแบบการเคลื่อนที่ได้จาก Properties Panel

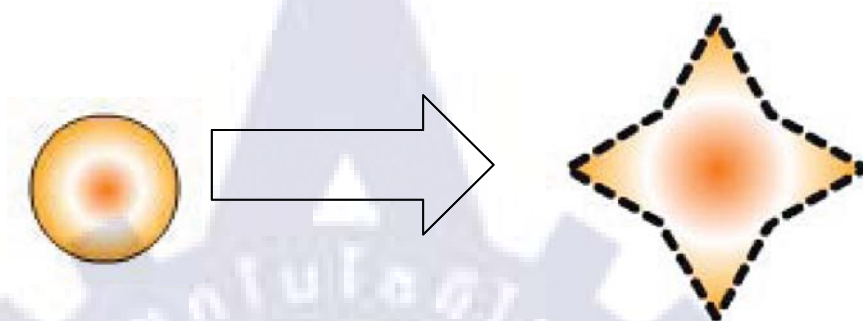
หลักการสร้าง Movie แบบ Motion Tween

- กำหนดเฟรมเริ่มต้น (ไม่จำเป็นต้องเป็นเฟรมที่ 1) โดยการคลิกเมาส์ในเฟรมที่ต้องการแล้วเลือกคำสั่ง Insert, Timeline, Keyframe หรือคลิกปุ่มขวาของเมาส์ แล้วเลือก Insert, Keyframe
- สร้างวัตถุ โดยวัตถุจะต้องอยู่ในสถานะ Group หรือ Symbol
 - o การสร้างวัตถุในสถานะ Group
 - เลือกวัตถุที่สร้าง
 - ใช้คีย์ลัด Ctrl + G
 - o การสร้างวัตถุในสถานะ Symbol
 - เลือกวัตถุที่สร้าง
 - กดปุ่ม < F8 >
 - ตั้งชื่อ Symbol และเลือก Behavior ที่เหมาะสม
- เลือกเฟรมปลายทาง แล้วเลือกคำสั่ง Insert, Timeline, Keyframe หรือคลิกปุ่มขวาของเมาส์ แล้วเลือก Insert, Keyframe
- ปรับแต่งวัตถุ หรือย้ายวัตถุตามต้องการ

- เลื่อนเมาส์กลับมาคลิกที่เฟรมต้นทาง แล้วคลิกปุ่มขวาของเมาส์ เลือกคำสั่ง CreateMotion Tween

Shape Tween

Shape Tween เป็น Movie อีกลักษณะที่ใช้เทคนิคการเปลี่ยนรูปร่างของวัตถุ เช่น จากวัตถุขนาดเล็กๆ สั้นๆ เมื่อเวลาผ่านไปก็กลายสภาพเป็นวัตถุขนาดยาว หรือจากรูปทรงวงกลม เป็นรูปหลายเหลี่ยม เป็นต้น



ภาพที่ ก.70 ตัวอย่างภาพเคลื่อนไหวแบบ Shape Tween

หลักการสร้าง Movie แบบ Shape Tween

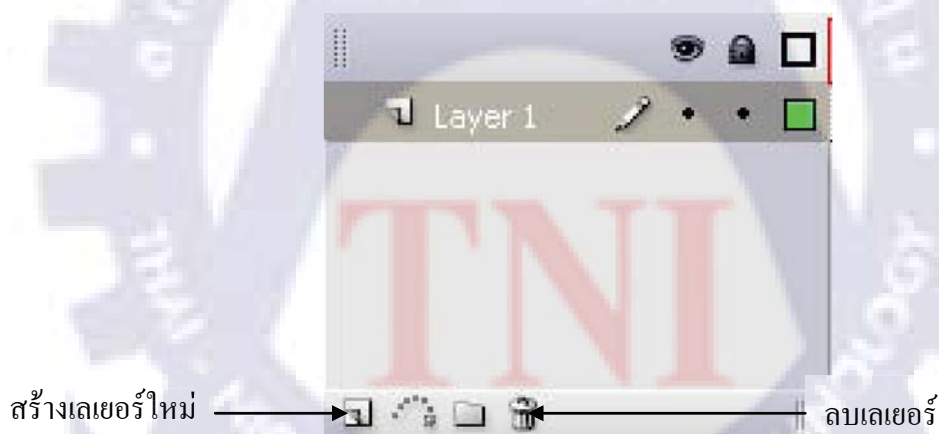
- กำหนดเฟรมเริ่มต้น (ไม่จำเป็นต้องเป็นเฟรมที่ 1) โดยการคลิกเมาส์ในเฟรมที่ต้องการแล้วเลือกคำสั่ง Insert, Timeline, Keyframe หรือคลิกปุ่มขวาของเมาส์ แล้วเลือก Insert, Keyframe
- สร้างวัตถุ โดยวัตถุจะต้องอยู่ในสถานะ Shape เท่านั้น
 - o กรณีที่วัตถุอยู่ในสถานะ Group หรือ Symbol ต้องแปลงให้อยู่ในสถานะ Shape ก่อน โดย
 - เลือกวัตถุ
 - กดปุ่ม Ctrl + B (อาจจะต้องทำหลายครั้ง) จนกว่าวัตถุมีจุดฟิกเชลเล็กๆ ประกอบในวัตถุ
- เลือกเฟรมปลายทาง แล้วเลือกคำสั่ง Insert, Timeline, Keyframe หรือคลิกปุ่มขวาของเมาส์ แล้วเลือก Insert, Keyframe
- สร้างวัตถุชิ้นใหม่ หรือปรับแต่งวัตถุเดิมตามต้องการ
- เลื่อนเมาส์กลับมาคลิกที่เฟรมต้นทาง กำหนดค่า Tween จาก Properties เป็น Shape



ภาพที่ ก.71 การปรับค่า Tween ให้เป็น Shape Tween

ทำงานกับเลเยอร์ (Layer)

เลเยอร์ (Layer) หมายถึงชั้นของชิ้นงาน เนื่องจากการวาดภาพใดๆ ก็ตามหากรวมอยู่ในชั้นเลเยอร์เดียวกัน จะเกิดปัญหาการแก้ไขได้ง่าย ดังนั้นการสร้างวัตถุใดๆ ก็ตามด้วย Flash ให้ยึดหลักที่ว่าวัตถุ 1 ชิ้นต่อ 1 เลเยอร์เพื่อให้การควบคุม การสร้างงานกระทำได้อย่างสมบูรณ์



ภาพที่ ก.72 หน้าต่าง Layer

เปลี่ยนชื่อเลเยอร์

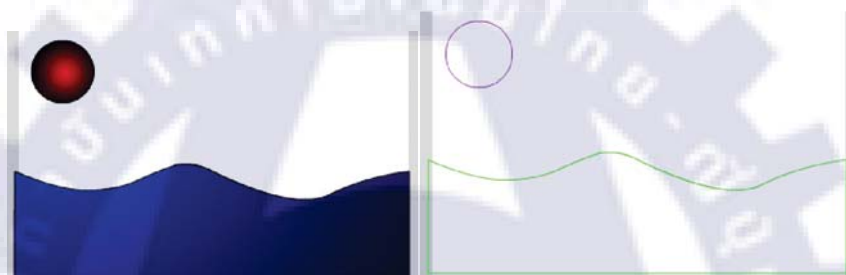
- ดับเบิลคลิกที่ชื่อเลเยอร์เดิม แล้วป้อนชื่อใหม่จากนั้นกดปุ่ม <Enter>

เปลี่ยนลำดับของเลเยอร์

- นำเมาส์ไปคลิกที่ชื่อเลเยอร์ที่ต้องการเปลี่ยนลำดับ กดปุ่มเมาส์ค้างไว้ แล้วลากไปปล่อย ณ ตำแหน่งใหม่

ควบคุมเลเยอร์

-  ซ่อน/แสดงเลเยอร์ โดยให้คลิกที่จุดกลมเล็กๆ ของเลเยอร์ได้ปุ่มนี้
-  ล็อกเลเยอร์ ป้องกันการแก้ไข
-  แสดงผลแบบโครงร่าง เหมาะสำหรับการสร้าง Movie กับเครื่องที่มีแรมไม่มากนัก



การแสดงผลแบบปกติ

การแสดงผลแบบโครงร่าง

ภาพที่ ก.73 ตัวอย่างการแสดงผลแบบโครงร่าง

ภาคผนวก ข.

วิธีการเล่นเกม และการใช้งาน



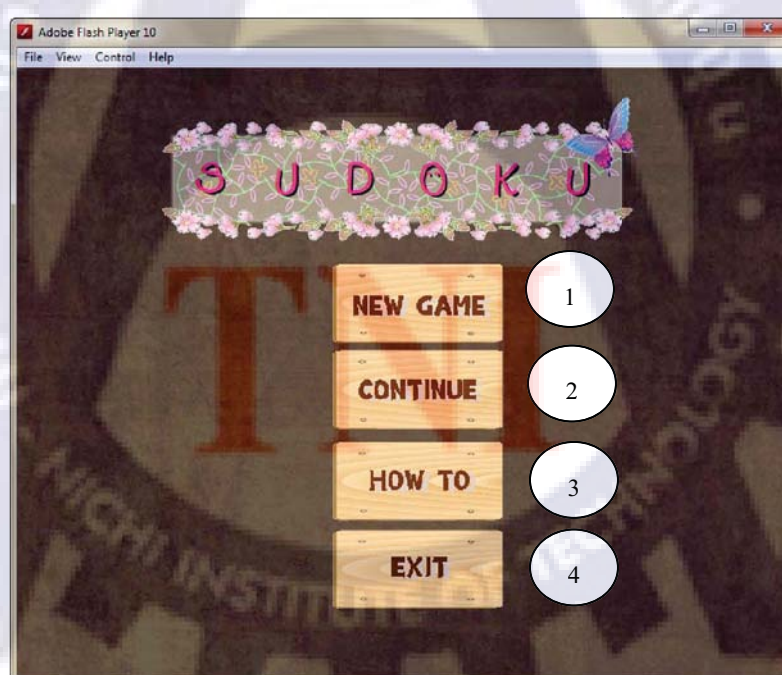
วิธีการเล่นเกม และการใช้งาน

Sudoku คือ เกมปริศนาตัวเลข ที่ผู้เล่นต้องเลือกใส่ หมายเลขตั้งแต่ เลข 1 ถึงเลข 9 โดยมีเงื่อนไขว่าในแต่ละแถวและแต่ละหลักตัวเลขต้องไม่ซ้ำกัน ตารางซูโดะกุจะมี 9×9 ช่อง ซึ่งประกอบจากตารางย่อย 9 ตาราง ในลักษณะ 3×3 แบ่งแยกกันโดยเส้นหนา และในแต่ละตารางย่อยจะต้องมีตัวเลข 1 ถึง 9 เช่นเดียวกัน เมื่อเริ่มเกมจะมีตัวเลขบางส่วนให้มาเป็นคำใบ้ และผู้เล่นจะต้องใส่ทุกช่องที่เหลือให้ครบ โดยตามเงื่อนไขว่าแต่ละตัวเลขในแต่ละแถวและหลักจะใช้ได้ครั้งเดียว รวมถึงในแต่ละขอบเขตตารางย่อย การเล่นเกมนี้จำเป็นต้องใช้ความสามารถในด้าน ตรรกะ และความอดทนรวมถึงสมาธิ

กติกาสำคัญมีดังนี้

ผู้เล่นต้องใส่ตัวเลข 1 – 9 ลงในตารางที่กำหนดช่องว่างไว้ โดยแต่ละแถวในแนวตั้งแนวนอน และตาราง 3×3 นั้นห้ามมีตัวเลขซ้ำกันเด็ดขาด

ส่วนประกอบต่างๆของเกม Sudoku



รูปที่ ข.1 หน้าเมนูของเกม

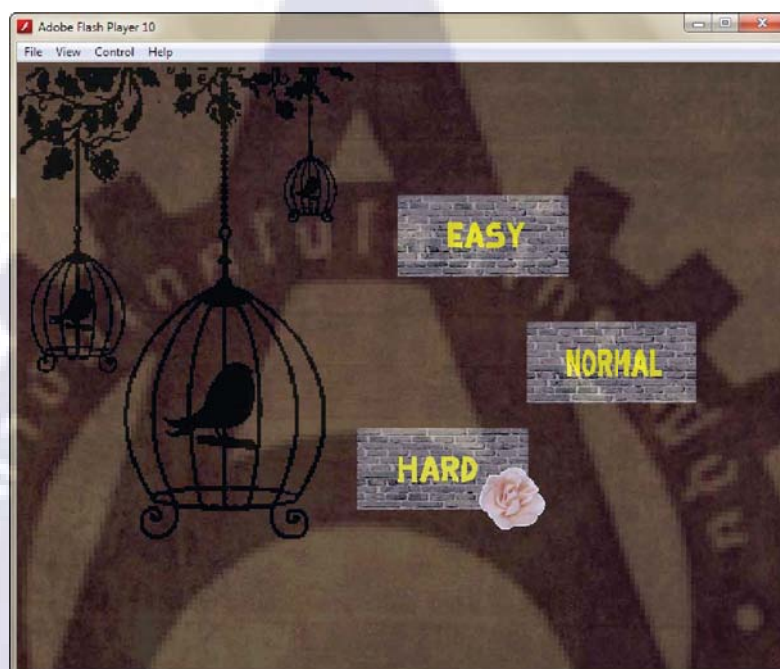
หน้าแรกของเกม โดยจะแสดงเมนูต่างๆภายในเกมทั้งหมด ซึ่งผู้เล่นสามารถเริ่มเกมใหม่หรือเลือกเรียกเกมที Save ได้ในหน้านี้ โดยจะมีปุ่มเมนูรายการต่างๆ ดังนี้

ส่วนที่ 1 ปุ่ม New Game ทำหน้าที่เริ่มเกม โดยเมื่อคลิกจะสามารถไปยังหน้า Level ได้

ส่วนที่ 2 ปุ่ม Continue ทำหน้าที่เรียกเกมจากการ Save เกมได้

ส่วนที่ 3 ปุ่ม How to ทำหน้าที่เรียกหน้า How to เพื่อแสดงวิธีการเล่นเกม

ส่วนที่ 4 ปุ่ม Exit ทำหน้าที่ปิดหน้าต่างเกม



รูปที่ ข.2 หน้าเลือก Level ของเกม

หน้าที่ 2 หน้านี้จะแสดงการเลือกระดับความยากของเกมโดยผู้เล่นสามารถเลือกได้ว่า จะเลือกระดับใด โดยมีระดับทั้งหมด 3 ระดับ

1. Easy ระดับง่ายสุด เหมาะสำหรับผู้เริ่มเล่น โดยจะมีการปิดตัวอักษรไม่เกิน 20 ตัวใน Level นี้

2. Normal ระดับปกติ เหมาะสำหรับผู้เล่นทั่วไป โดย Level นี้จะมีการปิดตัวอักษรทั้งหมดไม่เกิน 50 ตัว

3. Hard ระดับยาก เหมาะสำหรับผู้ที่ต้องการฝึกการคิดในระดับยาก เพราะจะมีการปิดตัวอักษรที่เยอะ และจะมีการเปิดแสดงตัวเลขเพียงไม่กี่ตัวเท่านั้น



รูปที่ ข.3 ส่วนประกอบภายในหน้าเกม

หน้าที่ 3 หน้าเกมหลัก โดยหน้าที่นี้จะแสดงเกม Sudoku ออกมาตามที่คุณเลือกไว้ โดยจะมีปุ่มที่ทำหน้าที่ต่างๆในการเล่นดังนี้

ส่วนที่ 1 ปุ่ม New Game ทำหน้าที่เรียกเกมใหม่ ซึ่งถ้ากด โปรแกรมจะทำการเรียกตาราง Sudoku ออกมาใหม่โดยทำการ เรียกสุ่มตัวเลขใหม่อีกด้วย

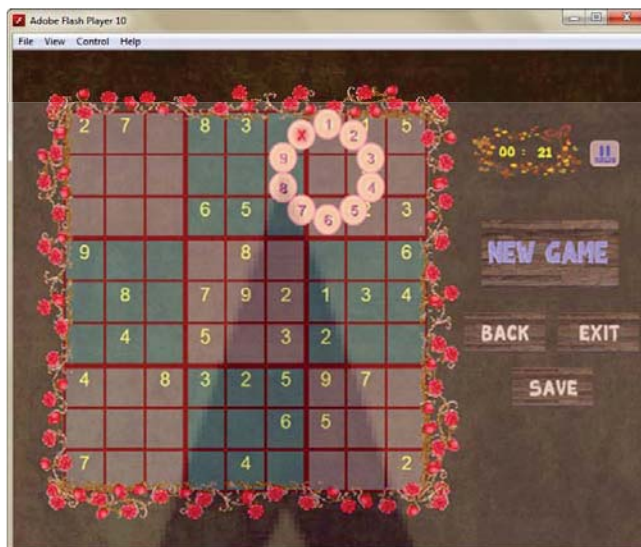
ส่วนที่ 2 ปุ่ม Back ทำหน้าที่กลับไปยังหน้า เมนูของเกม

ส่วนที่ 3 ปุ่ม Exit ทำหน้าที่ปิดหน้าต่างของเกมทั้งหมด

ส่วนที่ 4 ปุ่ม Save ทำหน้าที่เซฟเกม โดยสามารถเรียกเกมขึ้นมาจากการ Save ด้วยปุ่ม Continue

ส่วนที่ 5 ปุ่ม Pause/Play ทำหน้าที่หยุด/เริ่ม เวลาและเกมของเรา โดยถ้าเรากดปุ่ม Pause เกมจะไม่สามารถเล่นได้ในขณะนั้น และสามารถกดซ้ำเพื่อดำเนินเกมต่อไป

ส่วนที่ 6 ตารางเกม Sudoku ทำหน้าที่แสดงตัวเลขที่เราได้สุ่มออกและทำการปิดตัวเลขแล้ว ออกมา โดยผู้เล่นสามารถคลิกที่ปุ่มช่องว่าง หรือปุ่มสีเทาเพื่อทำการเรียกปุ่มตัวเลขขึ้นมาใส่ค่าได้



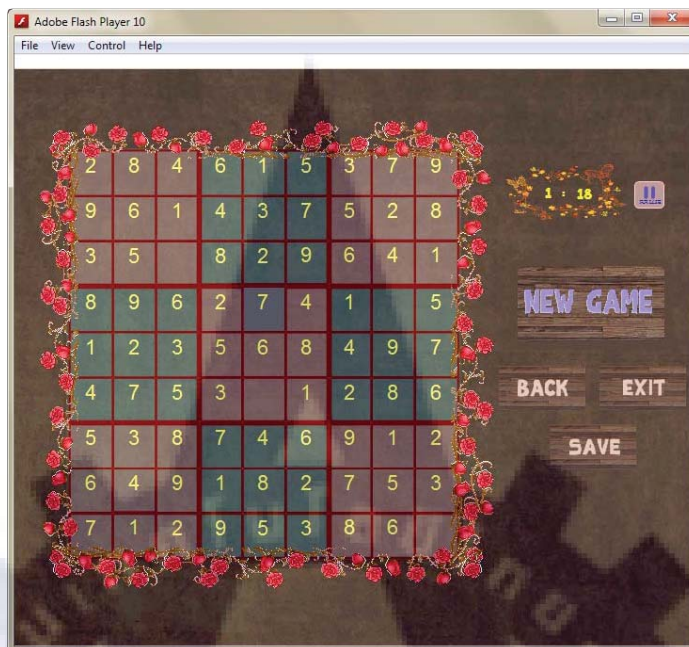
รูปที่ ข.4 การเรียกปุ่มกดเพื่อใส่ค่า



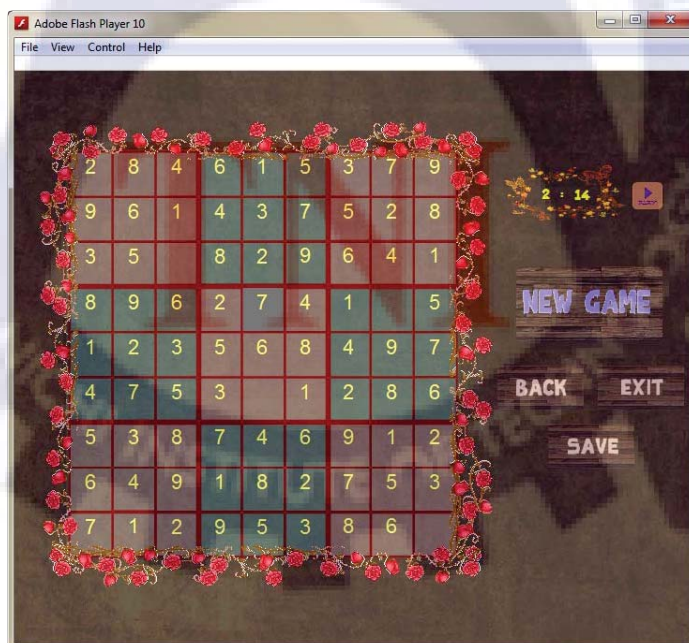
รูปที่ ข.5 เมื่อตัวเลขซ้ำกัน

จากภาพเมื่อมีการป้อนค่าตัวเลขที่ซ้ำกัน ระบบการจะคำนวณอัตโนมัติและจะทำการเปลี่ยนสีช่องที่ซ้ำทั้งให้เป็นที่ฟ้า

แต่ถ้าไม่มีตัวเลขใดๆซ้ำกัน ระบบจะมีการตรวจสอบอัตโนมัติเช่นกันและจะสามารถใส่เลขต่อไปได้เลย ดังภาพ ข.6



รูปที่ ข.6 ตัวเลขที่ไม่ซ้ำกัน



รูปที่ ข.7 หยุดเกม ก็จะไม่สามารถเล่นเกมได้และเวลาที่จะหยุดเดิน

โดยในระหว่างเกมผู้เล่นสามารถหยุดเวลาและเกมได้ตลอดเวลา โดยกดที่ปุ่ม



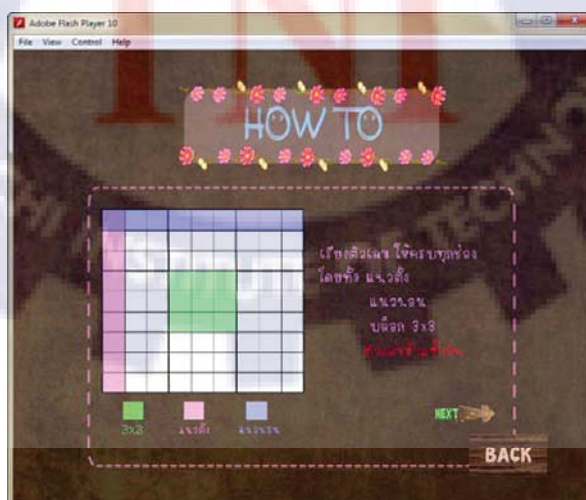
Pause และสามารถกลับมาเล่นอีกครั้งด้วยปุ่ม Play



รูปที่ ข.8 การแจ้งเตือนว่าชนะ

เมื่อผู้เล่นมีการใส่ตัวเลขทั้งหมดลงในช่องครบและตรงตามเงื่อนไขแล้ว ระบบจะทำการแจ้งเตือนว่า คุณชนะแล้ว สามารถไป level ต่อไปได้ดังภาพ

ซึ่งคุณนั้นจะสามารถดูวิธีการเล่นเกมทั้งหมด กติกาต่างๆ การใช้งานโปรแกรมได้ในหน้า How to นั้นเอง โดยสามารถเข้าหน้า How to ได้ทางหน้าต่างเมนูหลักของโปรแกรม



รูปที่ ข.9 หน้า How to

ประวัติผู้วิจัย

๕๐ - ๕๑

นางสาวณภัทร แก้วพิจิตร

วัน เดือน ปีเกิด

1 มกราคม พ.ศ. 2532

ประวัติการศึกษา

ระดับประถมศึกษา

ประณตศึกษาตอนปลาย พ.ศ.2544

โรงเรียนครุณาราชบุรี จ.ราชบุรี

ระดับมัธยมศึกษา

มัธยมศึกษาตอนปลาย พ.ศ. 2547

โรงเรียนราชโบริกานุเคราะห์ จ.ราชบุรี

ระดับอุดมศึกษา

คณะเทคโนโลยีสารสนเทศ

สาขาเทคโนโลยีสารสนเทศ พ.ศ. 2551

สถาบันเทคโนโลยีไทย – ญี่ปุ่น

ทุนการศึกษา -

ឃុំឈើ -

ผลงานที่ได้รับการตีพิมพ์ - ไม่มี -

