Chapter 2:
Maya Embedded Language Basic Concept

Topics:

Part I: variables, random, get attributes, and set attributes

Part II: statements, conditional operators, and loops

Chapter 2:
Maya Embedded Language Basic Concepts

เอกสารประกอบการเรียน
รายวิชา ANI 951301
สาขาวิชาแอนนิเมชันและเกม
วิทยาลัยศิลปะ สื่อ และเทคโนโลยี
มหาวิทยาลัยเชียงใหม่

951301: Advanced Animation and Rendering Techniques

วัตถุประสงค์

1. เพื่อให้นักศึกษามีความเข้าใจในคุณลักษณะ ประเภท และการใช้งานของ variables ในโปรแกรม Maya และ MEL scripts ว่ามีการทำงานอย่างไร
2. เป็นการปูพื้นฐานความเข้าใจใน programming basic concepts และ MEL scripting language
3. แนะแนวทางการประยุกต์ใช้เพื่อการ apply ความรู้ดังกล่าวในมิติของการ generate ตัวละครสามมิติ เพื่อเป็นแนวทางพื้นฐานให้นักศึกษาสามารถนำไปพัฒนาองค์ความรู้และต่อยอดได้ต่อไป
4. ให้นักศึกษาเข้าใจถึงการใช้ statements แบบต่างๆใน MEL
5. เข้าใจถึง concepts ของการใช้คำสั่ง loops ประเภทต่างๆ

เนื้อหาการสอน

ส่วนที่ 1: การใช้ variables, random, get attributes, และ set attributes

ส่วนที่ 2: การใช้ statements, conditional operators, และ loops ประเภทต่างๆ

MEL Basic Concepts

MEL เป็นโปรแกรมภาษาอันหนึ่ง ซึ่งมี rules และ structures เช่นเดียวกับโปรแกรมภาษาอื่นๆ นักศึกษาที่มีพื้นฐานการเขียนโปรแกรมอยู่แล้ว สามารถเข้าใจพื้นฐานการใช้ MEL ได้อย่างรวดเร็ว ส่วนนักศึกษาสายแอนนิเมชัน อาจยังขาดความรู้พื้นฐานในการเริ่มต้นทำความเข้าใจการเขียนโปรแกรมอยู่ เนื้อหาวิชานี้ออกแบบเพื่อให้นักศึกษาที่ไม่มีพื้นฐานการเขียนโปรแกรมสามารถเข้าใจได้ และช่วยปูพื้นฐานเบื้องต้นที่จำเป็น

โดยเนื้อหาในคาบนี้จะมุ่งเน้นที่ basic concepts ในการเขียนโปรแกรม MEL ซึ่งเป็นพื้นฐานในการประยุกต์ใช้โปรแกรมต่อไปในภายหน้า เนื้อหาอาจมีความซับซ้อนสำหรับบางคน แต่ขอให้ทำความเข้าใจความรู้พื้นฐานที่จำเป็นเหล่านี้เพื่อเป็นพื้นฐานการเรียนในภาคการศึกษานี้

Variables

Variables คือตัวแปรที่เราใช้ในการเขียนโปรแกรม เป็นสิ่งที่เก็บ data รูปแบบต่างๆใน MEL ถ้าสับสนว่า variable มีหน้าที่การทำงานอย่างไร ให้ลองจินตนาการตามสถานการณ์นี้ในการเปรียบเทียบ

นายสุชาติต้องการไปเบิกเงินที่ธนาคาร เมื่อเดินเข้าไปในธนาคารพบว่ามีผู้คนอยู่มากมาย ทางเจ้าหน้าที่ธนาคารจึงบอกให้นายสุชาติกดบัตรคิว นายสุชาติกดบัตรคิวออกมาได้เบอร์ A101 เมื่อถึงเวลาที่คิวของนายสุชาติมาถึง เจ้าหน้าที่ก็เรียกคิว A101 ออกมาโดยที่ทุกๆคนที่คอยคิวอยู่ในธนาคารทราบว่าหมายถึงนายสุชาติที่ถือบัตรคิวหมายเลขนั้นอยู่

ในเชิงการเขียนโปรแกรมก็เป็นเช่นเดียวกัน ให้คิดว่าในการเขียนโปรแกรมหนึ่งๆคือธนาคาร ส่วนลูกค้าที่เดินเข้ามาคือสิ่งที่เราต้องการนำค่าเข้ามาในโปรแกรม ส่วน variable ก็คือบัตรคิวนั่นเอง เพื่อให้ทุกคนในธนาคารทราบว่าเราหมายถึงใคร ทางธนาคารจึงสร้างตัวแปรขึ้นมาในการอ้างอิงถึงบุคคลนั้น

ในการทำงานของ MEL เราต้องสร้างสิ่งๆหนึ่งขึ้นมาเพื่อใช้ในการที่เราจะใส่ค่า data ต่างๆเข้าไป โดยเพื่อให้โปรแกรมทราบว่าสิ่งๆนั้นคือ variable เราต้องใส่เครื่องหมาย $ นำหน้าชื่อของสิ่งนั้น ยกตัวอย่างเช่นการประกาศตัวแปรชื่อ MyNumber สามารถทำได้โดยการตั้งชื่อว่า $MyNumber ทั้งนี้ตัว data ที่เราจะนำเข้ามามีหลายลักษณะ ซึ่งต้องเลือกประเภทของ variable ให้เหมาะสม เราลองมาดูกันว่าประเภทของ variables มีอะไรบ้าง

The Variable Types

เนื่องจากบทเรียนนี้เป็นการทำความเข้าใจขั้นพื้นฐาน concepts การใช้ MEL จึงขอแบ่ง variables ออกเป็นห้าประเภทหลักๆดังนี้

1. Integer Numbers

เรามาทำความรู้จักกับ variable ประเภทแรกกันเรียกว่า integer หมายถึงตัวเลขเต็มแบบไม่ทีจุดทศนิยม เช่น 1, 23, 512 เป็นต้น เราจะใช้ตัวเลขเหล่านี้เมื่อไหร่ ยกตัวอย่างของสถานการณ์ที่เห็นได้ชัดเช่น การทำ Boolean หรือการตรวจสอบ objects ภายใน list หรือการ return ค่าจากการสร้างเมนู เป็นต้น

ในการทำ Boolean เราใช้ตัวเลข 0 แทน off, no หรือ false เมื่อ 1 แทนค่า on, yes หรือ true (นักศึกษาจะเข้าใจขึ้นในบทต่อๆไป) ส่วนการตรวจสอบลำดับของ objects จากใน list โปรแกรม Maya จะ return ค่าเป็น integer numbers เช่น 0 หมายถึง ลำดับแรก 1 หมายถึงลำดับที่สอง 2 หมายถึงลำดับที่ 3 และต่อๆไปเป็นต้น และอีกสถานการณ์ที่พบเห็นได้บ่อยคือการ return ค่าจากปุ่ม radio buttons ในการสร้างเมนูด้วย MEL ซึ่งจะอธิบายอย่างละเอียดในบทหลังๆต่อไป

โดยที่คำสั่งในการประกาศตัวแปรแบบ integers สามารถทำได้โดยใช้คำสั่ง int ตามด้วย $ชื่อของตัวแปรนั้นๆ ตัวอย่างเช่น

```
int $myNumber = 4;
```

หมายถึงการประกาศตัวแปรแบบ integer ที่ชื่อ myNumber ให้มีค่าเท่ากับ 4 นั่นเอง

2. Floating-Point Number

Floating points คือตัวเลขทศนิยมแบบ 32 bits เช่น 1.2, 3.32357, 0.013 เป็นต้น เราจะต้องใช้ตัวเลขแบบนี้บ่อยมากในการทำงาน ยกตัวอย่างเช่นค่าตัวเลขต่างๆจากการ transform วัตถุ การกำหนดค่าสี หรือการกำหนดค่า weighing เป็นต้น

นักศึกษาอาจมีความสงสัยว่าเหตุใดเราจึงไม่ใช้ floating points ทั้งหมดแทน integers ในเมื่อ floating points ก็สามารถจัดการกับเลขที่ไม่มีเศษได้เช่นกัน (เช่น 3.0, 12.0 หรือ 423.0) ทั้งนี้เนื่องจากว่า floating-point number จะใช้ทรัพยากร RAM มากกว่า integers อย่างมาก เราจึงควรวางแผนและตัดสินใจเลือกใช้ integers ในทุกโอกาสที่ทำได้ และใช้ค่า floating-points เมื่อมีความจำเป็นเหมาะสมเท่านั้น อีกเรื่องหนึ่งที่เราควรจะต้องพึงระวังไว้ คือการที่ Maya จะปัดเศษทศนิยมออกให้เหลือเพียง 6

หลัก ในการ print ค่าใน interface หรือ file ซึ่งนักศึกษาควรจะพึงระวังไว้เมื่อต้องการนำตัวเลขไปใช้ในคำสั่งอื่นๆต่อไป

โดยที่คำสั่งในการประกาศตัวแปร floating-points สามารถทำได้โดยใช้คำสั่ง float ตามด้วย $ชื่อตัวแปรนั้นๆ ตัวอย่างเช่น

```
Float $pi = 3.1415926536;
```

หมายถึงการประกาศค่าตัวแปรแบบ floating-points ที่ชื่อ pi ให้มีค่าเท่ากับ 3.1415926536 นั่นเอง

3. Vector Numbers

ใน MEL นั้น vector numbers จะแสดงในรูปของ floating-points สามตัว คั่นด้วยเครื่องหมาย comma (,) และอยู่ระหว่างวงเล็บ angled brackets (<<…>>) ตัวอย่างของค่า vector numbers เช่น << 0, 1, 0 >>

เนื่องจาก vector numbers มีคุณลักษณะในการแสดงออกมาเป็นตัวเลขสามตัว จึงถูกนำไปใช้ในการบอกตำแหน่ง (position) ของจุด (vertex) วัตถุ, ค่าสี, IK handle's pole vector

โดยที่คำสั่งในการประกาศตัวแปร vector numbers สามารถทำได้โดยใช้คำสั่ง vector ตามด้วย $ชื่อตัวแปรนั้นๆ ตัวอย่างเช่น

```
vector $moveY = << 0, 1, 0 >>;
```

หมายถึงการประกาศค่าตัวแปรแบบ vector numbers ที่ชื่อ moveY ให้มีค่าเท่ากับ 0, 1, 0 นั่นเอง

4. Strings

Strings จะถูกใช้ในการจัดการกับข้อความ texts เช่นตัวอักษร, ชื่อ object หรือ ชื่อของส่วนประกอบต่างๆใน UI เมื่อตัวแปรถูกประกาศให้เป็น strings ไม่ว่าจะเป็นตัวเลขหรือตัวหนังสือโปรแกรมจะคำนึงมันเป็นเพียงตัวหนังสือเท่านั้น

ในการทำงานกับตัวแปร strings มีข้อพึงระวังคือ หนึ่งคือเรื่องของโปรแกรมจะคำนึงถึงตัวหนังสือแบบ case-sensitive นั่นคือ MyNumber จะมีความแตกต่างกับ mynumber หรือ myNumber และสองคือเรื่องของ escape characters โดยเราสามารถใช้ เครื่องหมาย "\" ในระหว่างคำเพื่อแสดงว่าตัวอักษรที่ตามหลังเครื่องหมาย "\" เป็น escape character

951301: Advanced Animation and Rendering Techniques

อะไรคือ escape characters? ทุกครั้งที่โปรแกรมเห็นเครื่องหมาย "\" โปรแกรมจะทำการมองไปที่ตัวหนังสือถัดไปว่ามีความหมายพิเศษอย่างไร ซึ่งอาจเป็น function พิเศษเช่น \n หมายถึงให้เริ่มบรรทัดใหม่, \r หมายถึง carriage return หรือ \t หมายถึงการ tab character นั่นเอง นอกจาก function พิเศษดังที่ยกตัวอย่างมาแล้ว escape characters ยังหมายถึงการเปลี่ยนความหมายของสัญลักษณ์เดิมที่มีอยู่เป็นอย่างอื่นได้ ตัวอย่างจากบทเรียนที่แล้วเช่นเมื่อเราจะให้โปรแกรม print ข้อความอะไรขึ้นมาสามารถทำได้โดยใส่ข้อความไประหว่างเครื่องหมาย "" เช่น

```
print "Hello"; //โปรแกรมจะ return ข้อความว่า Hello
```

แต่ถ้าเราต้องการให้แสดงตัว "" ด้วยสามารถทำได้โดยกำหนดให้ "" เป็น escape characters ด้วย \ ดังนี้

```
print "\"Hello\""; //โปรแกรมจะ return ข้อความว่า "Hello"
```

5. Array

ประโยชน์ของ array คือเราสามารถ collect ค่าหลายๆค่ามาไว้ในตัวแปรเดียวได้ ค่าแต่ละตัวใน array สามารถเป็นได้ทั้ง integers, floating-points, strings หรือ vector แต่ทุกว่าที่เก็บในหนึ่ง array จะต้องเป็นชนิดเดียวกัน ใน Maya เราจะเรียกว่าต่างๆที่เก็บไว้ใน array ว่า scalar นักศึกษาจะสังเกตได้จากหน้าต่าง feedback area เมื่อใดก็ตามที่ค่าใน array มีปัญหา เราจะได้ feedback รายงานว่า scalar มีปัญหา

เราสามารถตั้งค่า array ได้โดยการใช้เครื่องหมาย "[]" ต่อท้ายชื่อของ variable ที่ตั้ง ตัวอย่างเช่น

```
int $myNumber[];
float $objectDepth[];
string $objectName [];
```

ในกรณีที่เราต้องการกำหนดค่าไว้ในตัว array เลยตอนที่สร้างขึ้นมา สามารถทำได้โดยการใช้เครื่องหมาย "{}" ครอบค่าที่ต้องการโดยใช้ comma เป็นตัวแบ่งดังตัวอย่าง

```
int $myList[] = {1, 2, 3, 4, 5};
float $myList2[] = {0.1, -24, 15.99, 42};
string $myObject[] = {"myCube", "myCylinder", "mySphere",
"myCone"};
```

ในการใช้งาน array ค่าทุกๆค่าในหนึ่ง array จะเรียกว่า array element โดยที่แต่ละ array element จะมีหมายเลขประจำตัวของตัวเองอยู่เรียกว่า index ค่าแรกสุดใน array element จะมีหมายเลข index = 0 ตัวที่สองจะมี index = 1 ตัวที่สาม index = 4 แต่เพิ่มขึ้นแบบนี้ไปเรื่อยๆ ในการทำงานอาจเกิดความสับสนในช่วงแรกได้ เพราะ index เริ่มจาก 0 ไม่ใช่ 1 เหมือนการนับทั่วไปของเรา ตัวอย่างเช่น

951301: Advanced Animation and Rendering Techniques

```
string $myObject[] = {"myCube", "myCylinder", "mySphere",
"myCone"};
print $myObject[1];
```

โปรแกรมจะ return ว่า myCylinder แทนที่จะเป็น myCube เนื่องจาก myCube มี index = 0 นั่นเอง

เราสามารถใช้ array ในการเรียกดู object ที่ถูก selected ใน listได้ด้วยวิธีดังนี้

```
string $myObjectList[] = `ls -selection`;
print $myObjectList[0]; // โปรแกรมจะ print ชื่อ object ตัวแรกใน list ให้ดู
```

สำหรับการ print objects ที่ถูก selected ภายใน list สามารถทำได้โดย

```
print `ls -selection`;
// flag –selection สามารถย่อได้เป็น –sl
// ls คือตัว L เล็ก + S เล็ก โปรดอย่าสับสน
```

Workshop 1

(Kunkhet, 2017; Kunkhet and Klaynak, 2016)

ทีนี้เราลองนำความรู้ที่ได้มาประยุกต์ใช้ในแบบทดสอบง่ายๆกัน ยกตัวอย่างว่าถ้าเราต้องการสร้างรูปคนขึ้นมาอย่างง่ายๆโดยจะมีส่วนประกอบคือ หัว ตัว และขาสองข้างดังตัวอย่างด้านล่าง

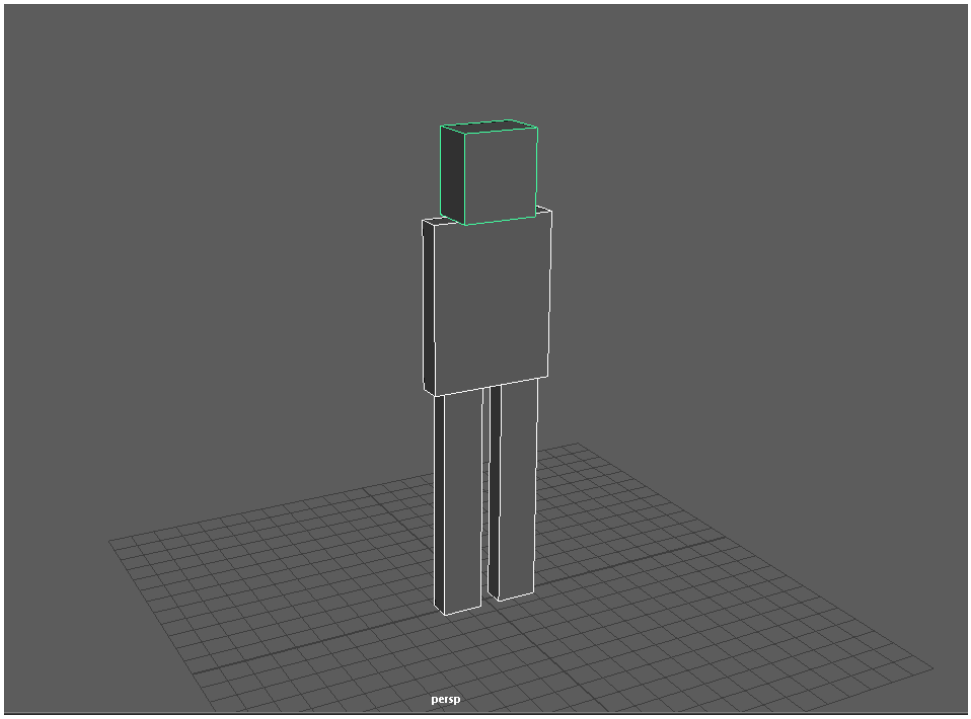

Fig 02-01: ตัวอย่างการจัดวางตัวละครขั้นพื้นฐานโดยการคำนวณจากสัดส่วนของวัตถุ

951301: Advanced Animation and Rendering Techniques

เราจะทดลองสร้างตัวละครตัวนี้ขึ้นด้วย MEL scripts โดยเริ่มจากการสร้างขาซ้ายก่อน แล้วจึงไล่ขึ้นไปถึงส่วนหัว ให้ตัวละครยืนอยู่บนระนาบของ grid lines ดังตัวอย่าง โดยให้นักศึกษาประกาศค่าตัวแปรเพื่อใช้ในเก็บค่าสัดส่วนต่างๆของตัวละคร ในการใช้คำนวณ proportion ของตัวละครทั้งหมด โดยกำหนดสัดส่วนดังนี้

1. ให้ขามีขนาดเริ่มต้นเท่าไหร่ก็ได้
2. ให้ลำตัวมีขนาดความสูงเป็น 0.7 เท่าของความสูงขา
3. ให้ลำตัวมีความกว้างเป็น 3 เท่าของความกว้างขา
4. ให้ลำตัวมีความลึกเป็น 1.1 เท่า ของความลึกขา
5. ให้ส่วนหัวมีความสูงเป็น 0.5 เท่าของความสูงลำตัว
6. ให้ส่วนหัวมีความกว้างเป็น 0.6 เท่าของความกว้างลำตัว
7. ให้ส่วนหัวมีความลึกเป็น 2 เท่าของความลึกลำตัว

เราสามารถทำได้โดยการใช้คำสั่ง MEL ดังนี้

```
// สร้างส่วนขา
     float $legH = 6;
     float $legW = 1;
     float $legD = 1;
     polyCube -h $legH -w $legW -d $legD -name LeftLeg;
     move -y ($legH/2);
// สร้างส่วนลำตัว
     float $bodyH = ($legH*0.7);
     float $bodyW = ($legW*3);
     float $bodyD = ($legD*1.1);
     polyCube -h $bodyH -w $bodyW -d $bodyD - name Body;
     select -r Body;
     move  -y ($legH+($bodyH/2));
     select -r LeftLeg;
     move -x (-$bodyW/4);
     polyCube -h $legH -w $legW -d $legD -name RightLeg;
     move -y ($legH/2);
     move -x ($bodyW/4);
// สร้างส่วนศีรษะ
     float $headH = ($bodyH*0.5);
     float $headW = ($bodyW*0.6);
     float $headD = ($bodyD*2);
     polyCube -h $headH -w $headW -d $headD -name Head;
     move -y ($legH+$bodyH+($headH/2));
```


951301: Advanced Animation and Rendering Techniques

นักศึกษาอาจเกิดความสงสัยว่าทำไมเราถึงต้องตั้งค่า variables ให้กับสัดส่วนต่างๆของตัวละคร ในเมื่อเราสามารถนั่งคำนวณแล้วใส่เป็นค่าตัวเลขไปได้ ทั้งนี้เนื่องจากเราสามารถนำไปประยุกต์ใช้ให้เป็นประโยชน์ได้ต่อไปในหลายลักษณะ ดังตัวอย่างต่อไปนี้ ถ้าเราเกิดต้องการสร้างตัวละครขึ้นมามากกว่าหนึ่งตัวล่ะ ถ้าเราต้องการสร้าง random characters ขึ้นมาหลายๆตัวให้มีขนาดไล่เลี่ยกัน แต่ไม่เหมือนกันทีเดียว เราสามารถทำได้โดยแก้คำสั่งที่เราสร้างขึ้นมาเมื่อครู่เพียงนิดเดียว แต่ก่อนอื่นเราลองทำความรู้จักคำสั่งที่ชื่อว่า random กันก่อน

Random

ในกรณีที่เราไม่ต้องการกำหนดค่าตัวแปรออกมาตายตัว แต่เราต้องการให้โปรแกรม random ค่าให้สามารถทำได้ด้วยคำสั่งดังนี้

```
float $Depth = rand(0.5,2);
```

ถ้าเราป้อนคำสั่งตามนี้ เราจะได้ค่าตัวแปรที่ชื่อว่า $Depth ที่มีค่าอยู่ระหว่าง 0.5 ถึง 2 การตั้งค่า random สามารถตั้งให้ตัวแปรได้ทั้งแบบ integers และ floating-points ตัวเลขในวงเล็บคือค่าต่ำสุดและสูงสุดที่อนุญาตให้สุ่มได้ โดยจะใช้ comma ในการคั่นระหว่างค่าสูงสุดและต่ำสุด

จากความรู้ตรงนี้เราสามารถแก้เฉพาะส่วนเริ่มต้นสามบรรทัดแรกของคำสั่งที่เราสร้างขึ้นเมื่อสักครู่ที่ผ่านมา โดยกำหนดให้ค่าตั้งต้นของขนาดขาเป็นค่า random ได้ดังนี้

เพิ่มสามบรรทัดนี้เข้าไปก่อนหน้าคำสั่งเดิมที่เขียนไว้ เพื่อสร้างตัวแปรขนาด กว้าง x ยาว x สูง ของขาเป็นตัวเลขแบบ random โดยกำหนดให้ค่าที่ได้เก็บในตัวแปรชื่อ $H, $W และ $D

```
float $H = rand(4,10);
float $W = rand(0.5,2);
float $D = rand(0.5,2);
```

แก้สามบรรทัดแรกของคำสั่งเดิมที่สร้างขึ้น แทนที่เราจะกำหนดเป็นตัวเลข แต่เราให้ดึงเอาค่า random มาใช้แทน

```
float $legH = $H;
float $legW = $W;
float $legD = $D;
```

จากนั้นลอง run คำสั่งดูจะพบว่าในแต่ละครั้งเราสามารถ generate ตัวละครที่มีสัดส่วนต่างๆกันออกมาได้ไม่จำกัด นี่เป็นตัวอย่างง่ายๆในการประยุกต์ใช้ MEL เบื้องต้น ในการทำงานและขอบเขตความสามารถของ MEL ยังมีอีกมากมายซึ่งจะได้นำเสนอในบทต่อไป

Get Attributes and Set Attributes

อีกสองคำสั่งที่มีความจำเป็นคือคำสั่งในการเรียกค่า attribute (getAttr) และคำสั่งในการตั้งค่า attribute (setAttr)

ตัวอย่างต่อไปจะช่วยให้นักศึกษาเข้าใจการใช้งานทั้งสองคำสั่งนี้มากขึ้น ให้นักศึกษาลองสร้าง sphere ขึ้นมาลูกหนึ่งโดยตั้งชื่อว่า Ball จากนั้นให้ move มันขึ้นตามแนวแกน Y = 5 units

```
polySphere -name Ball;
move -y 5;
```

จากนั้นเราลองใช้คำสั่ง get attribute เพื่อตรวจสอบค่าตำแหน่งของวัตถุที่ชื่อ Ball ดังตัวอย่างด้านล่าง

```
float $X = `getAttr Ball.tx`;
float $Y = `getAttr Ball.ty`;
float $Z = `getAttr Ball.tz`;
vector $BallPosition = << $X, $Y, $Z>>;
print $BallPosition;
//โปรแกรมจะ return ค่า 0 5 0 ที่ feedback area
```

จากตัวอย่างด้านบน เราตั้งค่าตัวแปรแบบ float ขึ้นมาสามตัว คือ $X, $Y และ $Z โดยใช้ get attribute สั่งให้เรียกค่า attributes ของ translate X, Y และ Z ตามลำดับ จากนั้นเราประกาศตัวแปรแบบ vector ขึ้นมาเพื่อเก็บค่าสามตัวนี้ไว้ แล้วจึงสั่ง print ค่าออกมาดูในตอนสุดท้าย

การใช้งาน getAttr จะพบได้บ่อยมากในเวลาที่เราต้องการค่าจากการคำนวณหรือ modify สักอย่างเข้ามาโดยที่เราไม่จำเป็นที่จะต้องระบุค่านั้นๆตอนประกาศ

เราได้เรียนรู้ถึงคำสั่ง move กันไปแล้ว move จะทำงานโดยเคลื่อน object จากจุดเดิมที่มันอยู่ไปยังตำแหน่งใหม่ด้วยค่าที่กำหนด นอกจากการใช้คำสั่ง move และ move -r แล้ว เรายังสามารถใช้คำสั่งกำหนดตำแหน่งของวัตถุได้ด้วยการใช้ set attribute จากโจทย์เดิมที่เรามี ให้เราลองย้ายวัตถุชื่อ Ball ของเราให้ไปอยู่ที่ตำแหน่ง translate X, Y และ Z เป็น 5, 1, 5 ด้วยคำสั่งด้านล่าง

```
setAttr Ball.tx 5;
setAttr Ball.ty 1;
setAttr Ball.tz 5;
```


951301: Advanced Animation and Rendering Techniques

เรายังสามารถใช้คำสั่ง set attribute ได้กับการทำงานอีกหลายๆประเภท เช่นการ set material

- ให้นักศึกษาทดลองสร้าง object ขึ้นมา 1 ชิ้น

- สร้าง material ใหม่ขึ้นมา ในที่นี้ขอใช้เป็น Lambert

- เปลี่ยนชื่อ material จาก lambert1 เป็น myMaterial

- ให้ apply material ตัวนี้ลงใน object ที่สร้าง

ทีนี้เราลองมาดูวิธีการปรับแต่งค่าต่างๆของ material ตัวนี้กัน โดยเราจะปรับแต่งค่าความโปร่งใส (transparency) และค่าสี (colour) ของมัน โดยเราสามารถปรับค่าความโปร่งใสได้ด้วยคำสั่ง

```
setAttr "myMaterial.transparency" -type double3 0 0 0;
```

โดยหมายเลข 0 0 0 ด้านท้ายแสดงค่าว่ามีความโปร่งใสเป็นศูนย์ (ทึบสนิท) ถ้าเราต้องการให้มันมีค่าความโปร่งใสเพิ่มขึ้นเท่าไหร่ให้เพิ่มค่าที่เลขสามตัวนี้ โดยจะมีค่าความโปร่งใสสูงสุดที่ 1 1 1

จากนั้นเราลองมาปรับค่าสีกัน โดยสามารถทำได้ด้วยคำสั่ง

```
setAttr "myMaterial.color" -type double3 0 0 0;
```

โดยหมายเลข 0 0 0 (สีขาว) แสดงค่าของการแทนสีแบบ RGB นั่นเองโดยแต่ละค่ามีค่าเต็มที่ 1 1 1 ซึ่งจะเป็นสีดำ ให้ลองเปลี่ยนค่าเป็น 1 0 0 จะได้สีแดงเป็นต้น

ถ้าเราต้องการสร้าง material ประเภท lambert ขึ้นมาใหม่ แล้วตั้งชื่อว่า myMaterial ผ่าน MEL สามารถทำได้โดย

```
shadingNode -asShader lambert -name myMaterial;
// สร้าง lambert ขึ้นมา ให้ชื่อว่า myMaterial //
sets -renderable true -noSurfaceShader true -empty -name
myMaterialSG;
connectAttr -f myMaterial.outColor
myMaterialSG.surfaceShader;
// เนื่องจากใน Maya ตัง material ตัวหนึ่งจะมี node มากกว่าหนึ่งตัวจึงต้องสร้างแล้ว
connect มันเข้าไว้ด้วยกัน โดยให้เราใช้ชื่อที่เราตั้งตามด้วย SG เพื่อป้องกันความสับสน //
```

แล้วเราสามารถ apply material ให้วัตถุด้วยคำสั่ง

```
select -r myObject1 myObject2 myObject3 myObject4;
// บรรทัดแรกเป็นการเลือกวัตถุที่เราต้องการ apply material ทั้งหมด (ถ้ามีอันเดียวใส่ชื่อเดียว)
//
sets -e -forceElement myMaterialSG;
// เป็นการ connect วัตถุที่เลือกเข้ากับ node ของ material (ในที่นี้ชื่อ
myMaterialSG) //
```


951301: Advanced Animation and Rendering Techniques

*ข้อพึงระวัง MEL จะใช้การสะกดคำแบบ American English เช่นคำว่า color ผู้ที่คุ้นเคยกับ British English (colour) ควรระมัดระวังในการเรียกใช้คำสั่ง

Workshop 2

(Kunkhet, 2017; Kunkhet and Klaynak, 2016)

จากความรู้ที่ได้เรียนมา ให้นักศึกษาลองออกแบบสร้างตัวละครขึ้นมาใหม่ (หรือจะใช้ตัวเดิมจาก workshop1 มาแก้ไขก็ได้) โดยที่ตัวละครตัวนี้ต้องมีส่วนประกอบของ หัว ตัว ขาสองข้าง และแขนสองข้าง เมื่อเสร็จแล้วให้ทดลองใส่ material ลงไปให้ตัวละครด้วย ดังตัวอย่างด้านล่าง

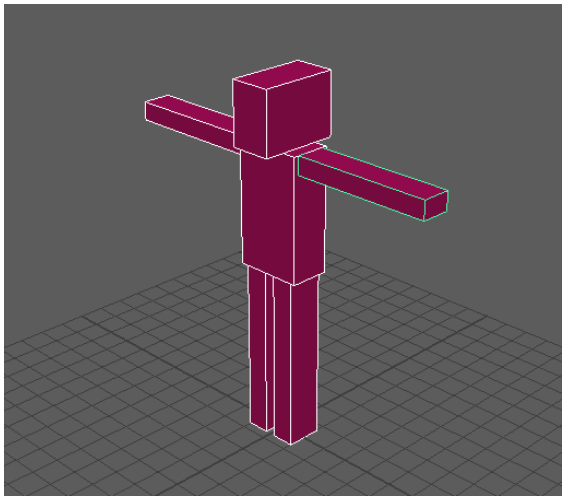

Fig 02-02: **ตัวละครที่ประกอบด้วยศีรษะ ลำตัว แขน และขา ที่มีการใส่พื้นผิว**

โดยในคราวนี้ให้นักศึกษาทดลองใช้คำสั่ง get attribute (getAttr) ในการหาตำแหน่งของวัตถุ แล้วทดลองใช้ set attribute (setAttr) ในการจัดวางตำแหน่งของวัตถุ ถ้ามีความสงสัยให้ลองทำตามตัวอย่างด้านล่าง แล้วทำความเข้าใจเพื่อไปประยุกต์ใช้ต่อ

```
// คำสั่งจาก workshop1 //
      float $H = rand(4,10);
      float $W = rand(0.5,2);
      float $D = rand(0.5,2);

      float $legH = $H;
      float $legW = $W;
      float $legD = $D;

      polyCube -h $legH -w $legW -d $legD -name LeftLeg;
      move -y ($legH/2);

      float $bodyH = ($legH*0.7);
```


951301: Advanced Animation and Rendering Techniques

```
	float $bodyW = ($legW*3);
	float $bodyD = ($legD*1.1);
	polyCube -h $bodyH -w $bodyW -d $bodyD -name Body;
	select -r Body;
	move  -y ($legH+($bodyH/2));

	select -r LeftLeg;
	move -x (-$bodyW/4);

	polyCube -h $legH -w $legW -d $legD -name RightLeg;
	move -y ($legH/2);
	move -x ($bodyW/4);

	float $headH = ($bodyH*0.5);
	float $headW = ($bodyW*0.6);
	float $headD = ($bodyD*2);
	polyCube -h $headH -w $headW -d $headD -name Head;
	move -y ($legH+$bodyH+($headH/2));

// คำสั่งใหม่เริ่มจากตรงนี้
	float $armW = ($legH*0.8);
	float $armH = ($legW*0.8);
	float $armD = ($legD*0.8);
	polyCube -h $armH -w $armW -d $armD -name LeftArm;
	move -y ($legH + $bodyH - $armH/2);
	move -x (-($armW/2+$bodyW/2));

	polyCube -h $armH -w $armW -d $armD -name RightArm;
	float $armPosY = `getAttr LeftArm.ty`;
	float $armPosX = `getAttr LeftArm.tx`;
	setAttr RightArm.tx (-$armPosX);
	setAttr RightArm.ty $armPosY;

	shadingNode -asShader lambert -name myMaterial;
	sets -renderable true -noSurfaceShader true -empty -name
myMaterialSG;
	connectAttr -f myMaterial.outColor
myMaterialSG.surfaceShader;

	select -r LeftLeg Body RightLeg Head LeftArm RightArm;
	sets -e -forceElement myMaterialSG;

	setAttr "myMaterial.transparency" -type double3 0 0 0;
	setAttr "myMaterial.color" -type double3 1 0 0.5;
```

Statements

คือการบอกถึงการตั้งค่าหรือเปรียบเทียบค่าว่าเป็นอย่างไรภายใน MEL ซึ่งสามารถแบ่งออกได้เป็นสามประเภทหลักๆคือ assignment statement, arithmetic statement และ condition statement

951301: Advanced Animation and Rendering Techniques

1. Assignment Statement

นี่เป็น statement ที่ง่ายที่สุดในการใช้ MEL ซึ่งเราได้ใช้กันไปพอสมควรแล้วตอนที่ประกาศค่าตัวแปร ส่วนที่สำคัญที่สุดของ assignment statement คือเครื่องหมายเท่ากับ (=) การสร้าง statement ตัวนี้เราจะประกาศตัวแปรในฝั่งซ้าย แล้วจึงประกาศค่าของมันทางขวามือ เราไม่จำเป็นที่จะกำหนดค่าตัวแปรตอนที่ประกาศเท่านั้น แต่เราสามารถปรับค่าได้ตลอดเวลาโดยการใช้ assignment statement ให้กับค่าใหม่ของมัน

```
int $myNumber = 20;
```

เมื่อเราประกาศตัวแปร $myNumber ให้มีค่าเท่ากับ 20

```
$myNumber = 2;
print $myNumber;
```

เราจะได้ผลลัพธ์ว่า $myNumber ถูกเปลี่ยนเป็น 2 ในการเปลี่ยนค่า (reassign) ให้กับตัวแปรใดๆ เราไม่จำเป็นต้องประกาศตัวแปรใหม่ เพียงใส่ค่าไปใหม่เท่านั้น

เมื่อใดก็ตามที่ฝั่งขวามือของเครื่องหมาย (=) เป็นค่าที่เราต้องการจาก return value เราสามารถทำได้โดยใช้สัญลักษณ์ (``) ครอบคำสั่งนั้น ดังที่เราเห็นได้จากตัวอย่าง get attribute ที่ผ่านมา

```
float $tx =`getAttr Ball.tx`;
```

สิ่งสำคัญในการใช้ assignment statement คือเราต้องประกาศประเภทของตัวแปรให้เหมาะสมกับชนิดของค่าที่เราต้องการจัดเก็บ การประกาศค่าตัวแปรผิดประเภทอาจทำให้เกิด error ได้

2. Arithmetic Statement

Arithmetic statement แตกต่างจาก assignment statement คือมันจะมีตัวเลขเป็นองค์ประกอบ เราไม่จำเป็นต้อง capture ผลลัพธ์ของ mathematical statement ลงไปใน variable โดยตรง แต่เราสามารถประหยัดเวลาได้โดยการใส่ mathematical statement ลงในคำสั่งโดยตรงได้เลย ดังตัวอย่างด้านล่าง

```
float $X = `getAttr Ball.tx`;
setAttr Ball.tx ($X + 5);
setAttr Ball.tx (`getAttr Ball.tx` + 5);
```

โดยที่บรรทัดที่สองและสามมีความหมายเหมือนกัน มีค่าเท่ากับ move –x 5; นั่นเอง

นอกจากเราสามารถใช้ mathematical statement กับตัวเลขได้แล้ว เรายังสามารถใช้กับข้อความตัวหนังสือ (string) ได้ด้วย

951301: Advanced Animation and Rendering Techniques

```
string $myString = ("My" + " " + "Cat");
print $myString;
//เราจะได้ผลลัพธ์ว่า My Cat
```

เรายังสามารถเปลี่ยนค่าของตัวแปรได้โดยการใช้ mathematical statement เช่นกัน

```
float $num = 5;
$num = ($num*2);
// Result: $num = 10
```

MEL ก็เช่นเดียวกับโปรแกรมภาษาอื่นๆ เมื่อเราเข้าใจพื้นฐานแล้วเราจะพบว่ามี shortcuts มากมาย ยกตัวอย่างเช่นการบวก ลบ คูณ หาร

```
float $myNum = 5;
$myNum /= 3;
//จะมีค่าเท่ากับคำสั่งเต็ม $myNum = ($myNum / 3);
//เช่นเดียวกับ +=, -=, *=
```

เราสามารถใช้ mathematical statement ในการเพิ่มหรือลดค่าของค่าตัวแปรที่เรามีทีละ 1 ได้โดยการใช้เครื่องหมาย (++) กับ (--) ใส่ที่ด้านหน้าหรือด้านหลังตัวแปรที่ต้องการ

```
int $x = 0;
//เมื่อใส่ (++ หรือ --) ไว้ด้านหน้า โปรแกรมจะเพิ่มหรือลดค่าให้กับตัวแปรตัวนั้น (ในที่นี้คือ $x) และมีผลต่อการคำนวณบรรทัดนั้นทันที ($y)
int $x = 0;
int $y = ++$x;
//$x จะมีค่า = 1
//$y จะมีค่า = 1

//เมื่อใส่ (++ หรือ --) ไว้ด้านหลัง โปรแกรมจะเพิ่มหรือลดค่าให้กับตัวแปรตัวนั้น (ในที่นี้คือ $x) แต่ค่าที่เปลี่ยนแปลงยังไม่มีผลต่อการคำนวณในบรรทัดนั้น ($y)
 int $x = 0;
          int $y = $x++;
//$x จะมีค่า = 1
//$y จะมีค่า = 0 หรือคือค่า $x ก่อนใส่ ++
```

3. Conditional Statement

เป็น statement ที่สร้างขึ้นมาเพื่อให้โปรแกรมตรวจสอบค่าที่ให้ไว้ เช่นว่าเป็น true หรือ false ซึ่งรูปแบบของมันสามารถแบ่งย่อยได้หลายประเภทดังนี้

3.1 If Statement

นี่เป็นคำสั่งพื้นฐานใน conditional statement เราสามารถเรียกใช้ได้ดังตัวอย่าง

```
if (condition)
statement;
```


951301: Advanced Animation and Rendering Techniques

condition คือเงื่อนไขที่เราตั้ง แล้ว statement คือ action ที่จะให้ทำถ้าเงื่อนไขเป็น true ตัวอย่างเช่น ถ้าเราต้องการทราบว่า object ที่ชื่อว่า Head มีอยู่ใน scene ที่เราทำงานอยู่ไหม ถ้ามีให้ print ข้อความว่า “Head is there” ให้เราทราบ สามารถทำได้โดย

```
if      (`objExists Head`)
                print "Head is there";
```

นอกจากการใช้ if แล้ว เรายังสามารถใช้ if + else เพื่อเพิ่มประสิทธิภาพในการทำงานได้อีกหลากหลาย ตัวอย่างของ if-else ได้แก่

```
if      (condition)
                statement;
else
                statement;
```

การใช้งาน if-else จะสามารถเพิ่มความยืดหยุ่นในการทำงานได้มากกว่า if ซึ่งจะ execute คำสั่งเมื่อ condition เป็น true เท่านั้น ถ้าไม่ใช่ก็จะไม่มีการกระทำต่อแต่อย่างใด แต่ if-else โปรแกรมจะตรวจสอบดูว่า condition เป็นจริงไหม ถ้าเป็นจริงก็ทำ ถ้าไม่จริงก็จะทำตามที่ else สั่ง

อีกประเภทของการใช้ if คือ else-if ซึ่งเป็นเสมือนการสร้าง layers ให้กับคำสั่ง if ตัวอย่างการใช้ else-if คือ

```
if      (condition)
        statement;
else if(condition)
        statement;
```

เราสามารถใช้ else-if สร้าง layers ให้ if มากเท่าไหร่ก็ได้ ตัวอย่างสถานการณ์ที่ใช้ เช่นเมื่อเราต้องการทำ actions ที่ต่างกัน ขึ้นอยู่กับประเภทของ object ที่ถูก selected ราสามารถใช้ทั้ง if, else-if และ else ร่วมกันได้ดังตัวอย่างด้านล่าง

```
if      (condition)
                statement;
else if(condition)
                statement;
else
                statement;
```


951301: Advanced Animation and Rendering Techniques

3.2 Switch Statement

การใช้ switch statement เหมาะกับการทำงานกับค่าที่ return มาของ rules based ที่มีความซับซ้อน เราสามารถกำหนดสิ่งที่ต้องการให้ทำ จาก values ที่ได้ return มาโปรแกรมจะเลือก execute คำสั่งที่เหมาะสม การใช้คำสั่ง switch statement เป็นการทำงานแบบ flow control ดูตัวอย่างการใช้ด้านล่าง

```
switch (condition)
 {
       case possibility:
             statement;
             break; // เราจะต้องมีการใช้คำสั่ง break ขั้นระหว่าง case possibilities
ทุกครั้ง
                              case possibility:
             statement;
             break;
                              case possibility:
             statement;
             break;
                              case possibility:
             statement;
             break;
       default: // มีค่าเท่ากับ else คือถ้า condition ไม่ตรงข้อไหนเลย ให้ทำตามข้อนี้
             statement;
             break;
 }
```

เมื่อโปรแกรมเจอคำว่า switch มันจะเข้าไปดูที่ condition จากนั้นไปดูที่ case โดยไล่จากบนลงล่าง ถ้า condition ตรงกับ possibility ของ case ไหน มันก็จะเข้าไปทำตาม statement ของ case นั้นแล้วจะสิ้นสุดที่คำสั่ง break ถ้า condition ไม่ตรงกับ possibility ของ case โปรแกรมจะวิ่งเข้าไปหา case ด้านล่างถัดไป เช่นนี้เรื่อยๆ โดยที่คำสั่ง default บรรทัดสุดท้าย ทำหน้าที่เช่นเดียวกับ else คือถ้าไม่เข้า case ไหนเลยจะให้ทำอย่างไร โดยที่เราจะมี default หรือไม่มีก็ได้ ขึ้นอยู่กับการออกแบบของเรา

Workshop 3

ให้นำความรู้ที่ได้จากการใช้ switch statement มาสร้างคำสั่งในการ random สร้างวัตถุกัน โดยใช้ประโยชน์จากการใช้คำสั่ง random ในการช่วย

```
int $myNumber = rand (1, 6);
switch ($myNumber)
 {
                    case 1:
             polyCube;
             break;
               case 2:
             polySphere;
```


951301: Advanced Animation and Rendering Techniques

```
            break;
             case 3:
            polyCylinder;
            break;
              case 4:
            polyCone;
            break;
              default:
            polyTorus;
            break;
 }
```

Conditional Operators

การใช้ conditional statement ควบคู่กับการใช้ conditional operators สามารถทำได้เช่นกัน คนที่มีความคุ้นเคยกับการเขียนโปรแกรมคงจะสามารถเข้าใจถึงคุณสมบัติของมันได้เป็นอย่างดี สำหรับคนที่ไม่มีพื้นฐาน conditional operators แบ่งเป็นสองประเภทได้แก่ Logical Operators และ Relation Operators

- Logical Operators จะใช้เพื่อเปรียบเทียบ data สองตัว สามารถใช้ได้กับ variables และ conditional statement ทุกประเภท ประกอบด้วย

|| จะ return true ถ้าฝั่งใดฝั่งหนึ่งของเครื่องหมายเป็นจริง

&& จะ return true ถ้าทั้งสองฝั่งของเครื่องหมายเป็นจริง

! จะ return true ถ้า argument ที่ตาม ! มาเป็น false

ตัวอย่างของการใช้ logical operators

```
if      (argument || argument)
        statement;
```

- Relational Operators สามารถใช้ได้กับ integers, floating-points และ vectors โดยที่จะมีเท่ากับ (==) และไม่เท่ากับ (!=) ที่สามารถใช้กับ string variables ได้ด้วย relational operators ทั้งหมดประกอบด้วย

> จะ return true เมื่อค่า value ด้านซ้ายมากกว่าค่าด้านขวาของ operator

< จะ return true เมื่อค่า value ด้านซ้ายน้อยกว่าค่าด้านขวาของ operator

== เมื่อค่า value ด้านซ้ายเท่ากับค่าด้านขวาของ operator

!= เมื่อค่า value ด้านซ้ายไม่เท่ากันกับค่าด้านขวาของ operator

951301: Advanced Animation and Rendering Techniques

>= เมื่อค่า value ด้านซ้ายมากกว่าหรือเท่ากันกับค่าด้านขวาของ operator

<= เมื่อค่า value ด้านซ้ายน้อยกว่าหรือเท่ากันกับค่าด้านขวาของ operator

ตัวอย่างของการใช้ relational operators ที่สั่งให้ตรวจดูว่า object ที่ชื่อ mySphere เคลื่อนที่ไปตามแนวแกน X เรื่อยๆทีละ 5 units จนกว่าจะพ้นจุด unit mark ที่ X = 20 โดยที่เราไม่คำนึงถึงตำแหน่งปัจจุบันของมัน

```
if    (`getAttr mySphere.tx` < 20)
                              move -r 5 0 0 mySphere;
```

Loops

หัวข้อต่อไปที่เราจะเรียนกันคือเรื่องของ loops ซึ่งมีหน้าที่ในการสั่งให้โปรแกรมทำคำสั่งใดๆซ้ำไปเรื่อยๆ จนกว่าจะได้สิ่งที่เราต้องการ มันอาจดูไม่ใช่สิ่งที่น่าสนใจ แต่เราต้องศึกษาไว้เนื่องจากเป็นหนึ่งในคำสั่งที่เราจะต้องใช้เป็นประจำในการทำงานต่อไป loops ที่เราจะใช้ในภาคการศึกษานี้แบ่งเป็นสองประเภทคือ while loop และ for loop ซึ่งมีรายละเอียดแตกต่างกันเล็กน้อย

1. The While Loop

เมื่อ MEL เจอคำสั่ง while loop โปรแกรมจะทำงานตามคำสั่งภายใน loop นั้นจนกว่าจะถึงเป้าที่เราได้ให้ไว้ (given condition is met) คล้ายกับ conditional statements ที่จะถูกทำซ้ำไปเรื่อยๆตราบใดที่ test condition ให้ผลลัพธ์ว่า true การใช้งาน while นั้นมีความคล้ายคลึงกับ if statement ดังตัวอย่างที่ให้ไว้ด้านล่าง

```
      while (condition)
                          statement;
```

ในการออกแบบคำสั่ง while loops นั้นมีข้อพึงระวัง เนื่องจากโปรแกรมจะทำคำสั่งซ้ำอย่างไม่มีวันสิ้นสุดจนกว่า condition statement จะ return ค่าว่า false ถ้าเราออกแบบคำสั่งไม่ดีอาจจะมีปัญหา run แบบไม่มีวันสิ้นสุดได้ ดังนั้นการออกแบบคำสั่ง while loop ผู้ออกแบบต้องมั่นใจว่าสุดท้ายแล้วเราสามารถได้ผลลัพธ์ที่ไม่สอดคล้องกับ condition ที่เรากำหนดไว้ตอนแรกได้

ด้านล่างคือตัวอย่างของการออกแบบคำสั่งที่มีความผิดพลาด ถ้านักศึกษาต้องการทดลองเพื่อสังเกตผลลัพธ์ อย่าลืม save งานในเครื่องให้เรียบร้อยก่อน

951301: Advanced Animation and Rendering Techniques

```
int $i = 1;
while ($i > 0)
print (($i++) + "\n");
```

ถ้าเรา run คำสั่งข้างต้นจะพบว่า Maya จะ return ค่าเป็นตัวเลขที่เพิ่มค่าขึ้นทีละหนึ่งแบบไม่รู้จบ เนื่องจากเราตั้งค่าให้ $i มีค่าเท่ากับหนึ่ง แล้วเราสั่งว่าตราบใด (while) ที่ $i มากกว่าศูนย์ให้โปรแกรม print ตัวเลขเพิ่มค่าขึ้นเรื่อยๆ ทางเดียวที่เราจะหยุดการทำงานของมันได้คือเราต้อง Terminate โปรแกรมเท่านั้น

2. The Do-While Loop

ข้อแตกต่างของ do-while loop คือ ถ้าเป็น while loop ปกติโปรแกรมจะตรวจสอบ condition ก่อนที่จะตรวจสอบ statement (สังเกตได้จากลำดับที่มาก่อนหลังของ condition และ statement ในการสร้าง while loop) ภายหลังจากการ execution do-while loop ครั้งแรกแล้ว มันจะทำงานเช่นเดียวกับ while loop แต่ความสำคัญมันอยู่ที่การ execution ครั้งแรกนี่เอง เนื่องจากการใช้ while loop ไม่มีอะไรมาประกันได้ว่า statement ของเราจะถูก execution เลยแม้แต่ครั้งเดียว ในขณะที่ do-while loop นั้น statement จะถูกวางไว้หน้า condition ดังตัวอย่างด้านล่าง

```
do
        statement;
        while (condition);
```

ให้สังเกตว่า do-while loop จะต้องมีเครื่องหมาย (;) ปิดท้าย condition ด้วย ซึ่งแตกต่างกับ while loop ถ้าเราลืมใส่จะทำให้เกิด error ขึ้น ตัวอย่างของ do-while loop คือ

```
do
        move -r 5 0 0;
while (`getAttr mySphere.tx` < 10);
```

จากคำสั่งนี้โปรแกรมจะเคลื่อน object ที่ชื่อ mySphere ไปตามแนวแกน X ทันที 5 units แล้วจึงตรวจสอบดู condition ว่ามันเกิน 10 units marking point ถ้าเกินแล้วมันก็จะหยุดทำงาน แต่ถ้าไม่เกินก็จะเคลื่อนต่อจนกว่าจะเกิน ในขณะที่ถ้าเราใช้ while loop โปรแกรมจะตรวจสอบ condition ก่อนแล้วจึงจะเคลื่อนวัตถุ

951301: Advanced Animation and Rendering Techniques

3. The For-In Loop

มีประโยชน์เมื่อเราต้องการจัดการกับ elements ภายใน array คงจำกันได้ว่า array คือ variable ที่สามารถเก็บค่า values ได้มากกว่าหนึ่งค่าภายในเครื่องหมาย [] ค่าที่เก็บใน [] เรียกว่า elements เรามาดูตัวอย่างการใช้ for-in loop กัน

```
for ($variable in $array)
statement;
```

เราจะใช้คำว่า "in" คั่นระหว่าง variable กับ array ที่เราต้องการจัดการ ทีนี้เราลองมาดูตัวอย่างการใช้งานกัน ในที่นี้เราจะลองใช้ for-in loop ในการเรียกดูรายชื่อ objects ที่ถูก selected อยู่โดยโปรแกรมจะเรียง list ตามลำดับก่อนหลังในการ select ให้ ก่อนอื่นต้องมีวัตถุก่อน ลองสร้างวัตถุขึ้นมาสามชิ้นอะไรก็ได้ จากนั้น select ทั้งสามวัตถุ แล้วพิมพ์คำสั่งที่ command line ว่า

```
ls –sl // แอล เอส-เอส แอล
```

เพื่อให้โปรแกรม list ตัว objects ทั้งสามออกมา จากนั้นลองพิมพ์คำสั่งที่ script editor

```
string $myList[] = `ls -sl`;
string $myCurrentObj; // หมายถึงวัตถุที่ถูก selected อยู่ ณ ขณะนี้

for ($myCurrentObj in $myList)
{
        print ("You have selected " + $myCurrentObj + "\n");
}
```

ความหมายคือเราประกาศตัวแปรประเภทข้อความ array ชื่อ $myList[] โดยให้ elements ภายใน list คือชื่อของ objects ที่กำลังถูกเลือกอยู่ (`ls –sl`) จากนั้นเราประกาศตัวแปร $myCurrentObj ไว้เก็บค่า elements ใน array แล้วเราใช้ for-in สั่งให้โปรแกรมค้นหาชื่อภายใน array ที่เราบอกแล้วพิมพ์ข้อความ You have selected ตามด้วยชื่อของ objects เหล่านั้น ซึ่งจะได้ผลลัพธ์ประมาณนี้

```
You have selected yourObj1
You have selected yourObj2
You have selected yourObj3
```

4. The for Loop

For loop มีความสามารถมากกว่า for-in แต่ก็มีความซับซ้อนเพิ่มขึ้นมาเล็กน้อย เราจะใช้ for loop เมื่อสถานการณ์ที่เรามี variable ที่เราต้องการเริ่มที่ค่าใดค่าหนึ่ง แล้วเปลี่ยนแปลงค่าทุกครั้งที่ loop runs ในแต่ละครั้งที่ loop runs โปรแกรมจะตรวจสอบว่าควรจะ run ต่ออีกครั้งหรือหยุดลง รูปแบบการใช้ for loop คือ

951301: Advanced Animation and Rendering Techniques

```
for (initialisation; condition; update condition)
statement;
```

จะสังเกตได้ว่ามี MEL statements สามประเภทคั่นด้วยเครื่องหมาย semicolons (;) ภายในวงเล็บเรามาดูว่าแต่ละตัวมีหน้าที่อย่างไร

- Initialisation จะถูก run ครั้งแรกที่เริ่ม loop โดยปกติจะเป็นการ set ค่าให้กับตัว variable ที่จะเปลี่ยนแปลงค่าเมื่อมีการ run loop ขึ้น
- Condition ทำหน้าที่เช่นเดียวกับ condition ใน while loop คือจะเป็นเงื่อนไขชี้ชัดว่า loop ยังควรจะ run ตัว statement ภายในต่อหรือไม่
- Update condition ทุกครั้งภายหลังจากที่ statement ภายใน loop runs ก่อนที่ condition จะถูกตรวจสอบในรอบต่อไป update condition จะทำการแก้ไขค่าของ initialisation ตามที่เรา set ไว้ แล้วดูว่าค่าใหม่ที่ได้ยังอยู่ใน condition ที่ควร run ต่อไปหรือไม่ การแก้ไขค่าเช่น การเพิ่มหรือการลดค่า initialisation นั่นเอง ลองดูตัวอย่างด้านล่าเพื่อความเข้าใจ

```
int $myInitial = 1;
for ($myInitial; $myInitial < 100; $myInitial++) // หรือเราจะประกาศค่าตัวแปร $myInitial ตรงนี้ก็ได้
{
    print ("This loop has run " + $myInitial + " times" + "\n");
}
```

จากตัวอย่าง เราใช้ for loop ทำการ run ไปเรื่อยๆ ตราบใดที่ตัวแปร $myInitial มีค่าน้อยกว่า 100 โดยให้เพิ่มค่าของ $myInitial ขึ้นทีละ 1 ในทุกๆรอบที่ run ($myInitial++) โดยให้ $myInitial มีค่าเริ่มต้น = 1

References:
- Kunkhet, A., (2017) Designing an Application of 3D Character Modelling in Trialling Size of Clothes on Online Clothing Shopping, *International Conference on Digital Arts, Media and Technology*, ICDAMT 2017.
- Kunkhet, A. and Klaynak, K. (2016) Virtual Reality and Collaborative Learning to Improve Students' Learning Experiences in 3D Modelling, *International Conference on Digital Arts, Media and Technology*, ICDAMT 2016, pp. 357-361.
- Mark R .Wilkins and Chris Kazmier)2005 (*MEL Scripting for MAYA Animators*, Morgan Kaufmann Publishers, Elsevier Inc.
- Robert Galanakis (2014)*Practical Maya Programming with Python*, Packt Publishing Ltd.
- David Stripinis)2003 (*The MEL Companion :Maya Scripting for 3D Artists*, Charles River Media, INC

951301: Advanced Animation and Rendering Techniques