Chapter 9:
Crowd Behaviour and Environment

Topics:

Pseudo random, interaction, and crowd system

Chapter 9:
Crowd Behaviour and Environment

เอกสารประกอบการเรียน
รายวิชา ANI 951301
สาขาวิชาแอนนิเมชันและเกม
วิทยาลัยศิลปะ สื่อ และเทคโนโลยี
มหาวิทยาลัยเชียงใหม่

951301: Advanced Animation and Rendering Techniques

วัตถุประสงค์

1. เข้าใจถึงการสร้างสิ่งกีดขวาง (obstacles) ให้กับการจำลองฝูงชน และความเปลี่ยนแปลงทางพฤติกรรมของฝูงชน
2. เข้าใจถึงความแตกต่างและการใช้งานของ passive rigid body และ active rigid body
3. เข้าใจถึงหลักการสร้าง crowd system ขนาดใหญ่
4. สามารถสร้างหน้าต่างควบคุมการจำลองพฤติกรรมของฝูงชนขนาดใหญ่
5. สามารถนำความรู้ที่ได้ไปพัฒนาสร้างระบบการจำลองพฤติกรรมฝูงชนในรูปแบบอื่นๆได้ด้วยตนเอง

เนื้อหาการสอน

การจำลองการเคลื่อนที่แบบสุ่มด้วย pseudo random การสร้างการโต้ตอบระหว่างฝูงชน และการจำลองพฤติกรรมของฝูงชน crowd system

951301: Advanced Animation and Rendering Techniques

Creating Environment for Crowd System

ในบทนี้เราจะมาเรียนรู้การเปลี่ยนสถานภาพวัตถุธรรมดา (objects) ให้เป็นสิ่งกีดขวาง (obstacles) ที่สามารถโต้ตอบกับ crowd system ที่เราสร้างไว้ในบทก่อนหน้านี้ได้ ในการสร้างการโต้ตอบให้กับวัตถุและสิ่งกีดขวางนั้น เราจะต้องทำให้ทั้งคู่กลายเป็น rigid body ก่อน แตกต่างกันที่ วัตถุที่เคลื่อนที่จะถูกสร้าง rigid body แบบ active ในขณะที่สิ่งกีดขวางจะเป็นแบบ passive ทั้งนี้เนื่องจากวัตถุเสมือนฝูงชนจะเคลื่อนที่ตลอดเวลา ในขณะที่สิ่งกีดขวางจะคงอยู่ไม่มีการเคลื่อนไหว

Creating Obstacles

เราจะเริ่มขั้นตอนจากการสร้างออกแบบสิ่งกีดขวางกันก่อน เพื่อให้สามารถเห็นความแตกต่างอย่างชัดเจน เราจะลองสร้างกำแพงขึ้นมาสี่ด้าน โดยให้มีช่องว่างระหว่างกำแพงแต่ละด้านเป็นช่องทางให้วัตถุเราสามารถเข้าออกได้ และเพื่อการขยาย interactions ระหว่างวัตถุกับสิ่งกีดขวาง เราจะสร้างสิ่งกีดขวางขึ้นมาจำนวนหนึ่งไว้ระหว่างกำแพงทั้งสี่ด้านด้วย นักศึกษาสามารถดูได้จากภาพประกอบด้านล่าง

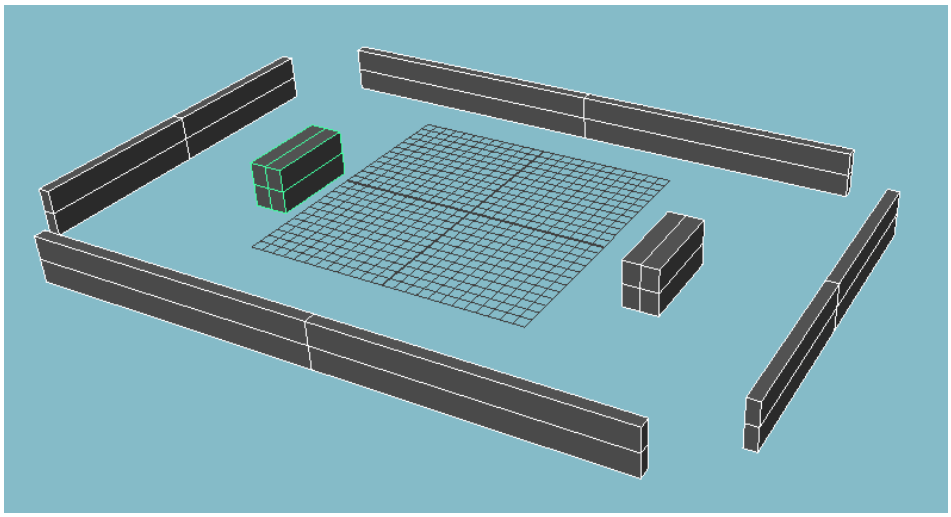

Fig 09-01: ตัวอย่างการจำลองฉาก โดยให้สร้างกำแพงขึ้นมาสี่ด้าน และให้มีวัตถุอยู่ภายในจำนวนสองอัน

เรามาเริ่มต้นสร้างสิ่งกีดขวางกัน โดยเริ่มจากการสร้าง polygon cube ที่ชื่อว่า ob1 ให้อยู่ในตำแหน่ง X = 0, Y = 0, Z = 20 ด้วยคำสั่ง xform ก่อนที่จะใช้คำสั่ง scale จัดการให้ polygon

cube ของเรามีลักษณะเหมือนกำแพงขนาดใหญ่ โดยให้เพิ่มขนาดตามแนวแกน X = 50 เท่า, แกน Y = 4 เท่า, และแกน Z มีขนาดเท่าเดิม ตามคำสั่งด้านล่าง

```
	// สร้าง polygon cube ชื่อว่า ob1 ให้มี subdivisions ตามแนวแกน X, Y, Z เป็น 2, 2, 1
	polyCube -name ob1 -sx 2 -sy 2 -sz 1;
	// ย้ายวัตถุไปที่ตำแหน่ง Z = 20 และให้แก้ไขสัดส่วนเป็น 50, 4 และ 1 เท่าจากค่าเดิม
	xform -worldSpace -translation 0 0 20 -scale 50 4 1 ob1;
```

เมื่อเสร็จแล้วให้ทดลอง execute คำสั่งดูจะได้ผลลัพธ์ดังภาพตัวอย่างต่อไป

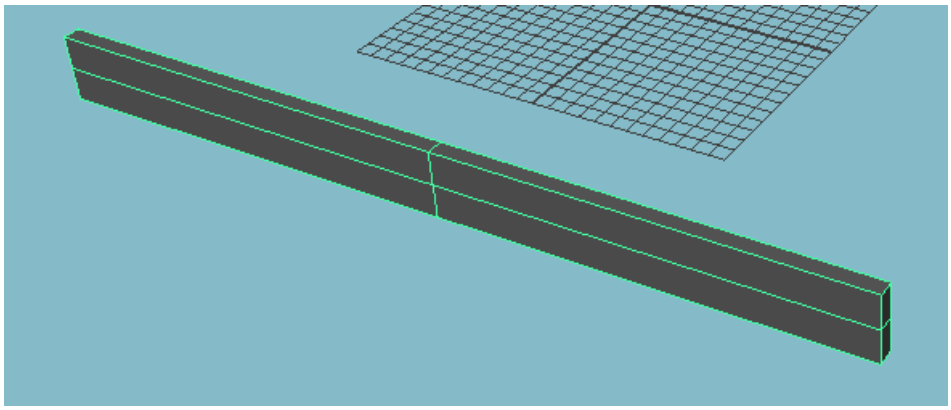

Fig 09-02: แสดงกำแพงที่ถูกสร้างขึ้นมาจาก polyCube

จากนั้นให้สร้างกำแพงขึ้นมาอีกสามด้านด้วยวิธีการเดียวกับที่สร้างกำแพงอันแรก เพื่อสร้างห้องลักษณะสี่เหลี่ยมผืนผ้าขึ้นมา โดยกำแพงที่สองจะมีขนาดเท่ากับกำแพงแรกแต่จะวางอยู่ในตำแหน่ง Z = -20 (ตรงข้ามกับกำแพงอันแรก)

```
// ตั้งชื่อกำแพงที่สองว่า ob2
polyCube -name ob2 -sx 2 -sy 2 -sz 1;
// วางให้อยู่ตรงข้ามกับกำแพงแรกตามแนวแกน Z
xform -worldSpace -translation 0 0 -20 -scale 50 4 1 ob2;
```

กำแพงแรกและกำแพงที่สองจะอยู่ห่างกัน 40 units (Z = 20, Z = -20) เราจะสร้างกำแพงที่สามและสี่ให้มีขนาดเพียง 30 units เพื่อให้มีช่องว่างระหว่างให้วัตถุสามารถผ่านเข้าออกได้ ตามตัวอย่างด้านล่าง

```
// สร้างกำแพงที่สามชื่อว่า ob3
polyCube -name ob3 -sx 2 -sy 2 -sz 1;
// วางให้อยู่ในตำแหน่ง X = 30 และ scale ขนาด 30 เท่าตามแนวแกน X ก่อนที่จะหมุนมัน 90 องศา
xform -worldSpace -translation 30 0 0 -scale 30 4 1 -rotation 0 90 0 ob3;
// สร้างกำแพงที่สี่ชื่อว่า ob4
```


951301: Advanced Animation and Rendering Techniques

```
polyCube -name ob4 -sx 2 -sy 2 -sz 1;
// วางให้อยู่ในตำแหน่งตรงข้ามกับกำแพงที่สาม และหมุนมัน 90 องศาเช่นกัน
xform -worldSpace -translation -30 0 0 -scale 30 4 1 -
rotation 0 -90 0 ob4;
```

เมื่อลอง execute คำสั่งดูจะได้ผลลัพธ์ตามภาพตัวอย่างหน้าถัดไป ให้ตรวจสอบดูว่ากำแพงแต่ละด้านไม่มีส่วนที่ซ้อนทับกันอยู่ และมีช่องว่างตรงมุมทั้งสี่ของห้องโดยต้องมีเนื้อที่มากพอให้วัตถุวิ่งผ่านเข้า-ออกไปได้

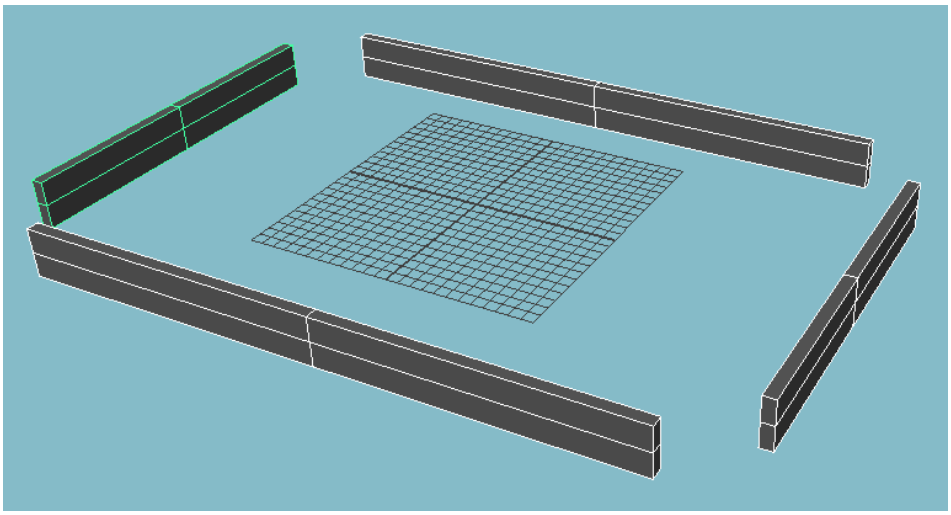

Fig 09-03: แสดงผลสำเร็จเมื่อสร้างกำแพงออกมาทั้งสี่ด้าน

หลังจากที่เราสร้างกำแพงทั้งสี่ด้านแล้ว เรามาเพิ่มสิ่งกีดขวางภายในพื้นที่ตรงกลางกัน ให้ทดลองสร้างวัตถุ polygon cube ขึ้นมาสองชิ้นวางอยู่ในตำแหน่งตรงกันข้ามกัน ใกล้กันกับกำแพง ob3 และ ob4 (ตำแหน่ง X = 17, Y = 0, Z = 0 และ X = -17, Y = 0, Z = 0) และให้ปรับ scale ของทั้งสองวัตถุที่สร้างเป็น X = 3, Y = 4, Z = 8 และตั้งชื่อวัตถุว่า ob5 และ ob6 ตามลำดับ เมื่อ execute คำสั่งดูจะได้ผลลัพธ์ดังนี้

951301: Advanced Animation and Rendering Techniques

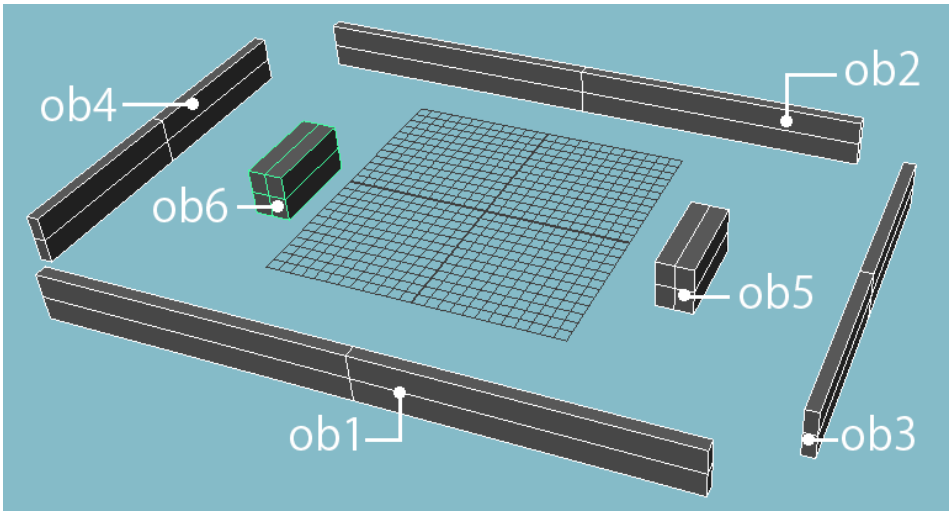

Fig 09-04: แสดงตำแหน่งการตั้งชื่อของกำแพงทั้งสี่ด้านและวัตถุทั้งสองอัน

เมื่อทดลอง execute scripts แล้วได้ผลลัพธ์ตามที่ต้องการ ขั้นต่อไปให้จัดเก็บ scripts ไว้ในรูปแบบของ global procedure เนื่องจากเราจะต้องเรียกใช้อีกในส่วนต่อไป โดยให้เพิ่มข้อความใน scripts เดิมดังนี้

```
global proc environment()
{
  … scripts ที่เราเขียน...
}
```

จากนั้นให้ save เป็น dot MEL format และจัดเก็บไว้ที่ C:\...\Documents\maya\2016\scripts\ โดยให้ตั้งชื่อไฟล์เช่นเดียวกับชื่อของ procedure นั่นคือ environment.mel เมื่อเสร็จแล้ว ขั้นตอนสุดท้ายคือการ save เฉพาะคำสั่ง

```
      environment;
```

ไปเก็บไว้บน shelf เพื่อโอกาสในการเรียกใช้หลังจากนี้

ก่อนที่เราจะทำงานขั้นตอนต่อไป ให้เรานำ scripts ในการสร้างพฤติกรรมให้กับวัตถุสองชิ้นที่เราทำไว้ในบทที่ 7 กลับมาใช้อีกที โดยให้ทำเป็น global procedure ตั้งชื่อว่า createSimpleCrowd ซึ่งมี scripts ทั้งหมดดังนี้

```
global proc createSimpleCrowd()
{
rigidSolver -create -current -name crowdSolver
-velocityVectorScale 0.5
-displayVelocity on;
setAttr crowdSolver.allowDisconnection 1;

polyCube -name vehicleF_1 -width 2 -height 2.5 -depth 2;
```


951301: Advanced Animation and Rendering Techniques

```
// add a vehicle force field for follower
radial -position 0 0 0 -name vehicleFForce_1 -magnitude 50
-attenuation 0.3 -maxDistance 8.0;
parent vehicleFForce_1 vehicleF_1;
playbackOptions -min 1 -max 300;

rigidBody -name rigidVehicleF_1 -active -mass 1 -
bounciness 0
-damping 1.5 -position -10 0 0 -impulse 0.15 0.0 0.0 -
standInObject cube
-solver crowdSolver vehicleF_1;
disconnectAttr rigidVehicleF_1ry.output vehicleF_1.rotateY;
disconnectAttr rigidVehicleF_1rx.output vehicleF_1.rotateX;
disconnectAttr rigidVehicleF_1rz.output vehicleF_1.rotateZ;

$expString = "// vehicle type Follower";
$expString += "// set wander motion of vehicle and add
multipliers.\n";
$expString += "float $xMult1 = 2.0;\n";
$expString += "float $zMult1 = 1.5;\n";
$expString += "rigidVehicleF_1.impulseX = sin(time *
$xMult1);\n";
$expString += "rigidVehicleF_1.impulseZ = (noise(time) *
$zMult1);\n\n";
$expString += "// set orientation of vehicle according to
the velocity direction.\n";
$expString += "float $fVel1[] = `getAttr
rigidVehicleF_1.velocity`;\n\n";
$expString += "vehicleF_1.rotateX = 0;\n";
$expString += "vehicleF_1.rotateY = atan2d($fVel1[0],
$fVel1[2]);\n";
$expString += "vehicleF_1.rotateZ = 0;\n";
expression -s $expString -name wander -alwaysEvaluate true
-unitConversion all;

polyCube -name vehicleL_1 -width 2 -height 2.5 -depth 2;

radial -position 0 0 0 -name vehicleLForce_1 -magnitude 50
-attenuation 0.3 -maxDistance 8.0;
parent vehicleLForce_1 vehicleL_1;

radial -position 0 0 0 -name vehicleLeadGlobalForce_1
-magnitude -1 -attenuation 1.2;
parent vehicleLeadGlobalForce_1 vehicleL_1;

rigidBody -name rigidVehicleL_1 -active -mass 1 -
bounciness 0
-damping 1.5 -position 10 0 0 -impulse 0.15 0.0 0.0
-standInObject cube -solver crowdSolver vehicleL_1;
disconnectAttr rigidVehicleL_1ry.output vehicleL_1.rotateY;
disconnectAttr rigidVehicleL_1rx.output vehicleL_1.rotateX;
disconnectAttr rigidVehicleL_1rz.output vehicleL_1.rotateZ;

$expString = "// vehicle type Leader";
```


951301: Advanced Animation and Rendering Techniques

```
$expString += "// set wander motion of vehicle and add
multipliers. \n";
$expString += "float $xMult2 = -2.0; \n";
$expString += "float $zMult2 = 2.5; \n";
$expString += "rigidVehicleL_1.impulseX = sin(time *
$xMult2); \n";
$expString += "rigidVehicleL_1.impulseZ = (noise(time) *
$zMult2); \n\n";
$expString += "// set orientation of vehicle according to
the velocity direction. \n";
$expString += "float $fVel2[]; \n";
$expString += "$fVel2 = `getAttr rigidVehicleL_1.velocity`;
\n\n";
$expString += "vehicleL_1.rotateX = 0; \n";
$expString += "vehicleL_1.rotateY = atan2d($fVel2[0],
$fVel2[2]); \n";
$expString += "vehicleL_1.rotateZ = 0; \n";
expression -s $expString -name wanderL_exp1 -alwaysEvaluate
true
-unitConversion all;

connectDynamic -fields vehicleLForce_1 rigidVehicleF_1;
connectDynamic -fields vehicleLeadGlobalForce_1
rigidVehicleF_1;
connectDynamic -fields vehicleFForce_1 rigidVehicleL_1;
}
```

ให้ทำการบันทึก scripts ทั้งหมดข้างต้นในรูปแบบของ dot MEL format และจัดเก็บไว้ที่ C:\...\Documents\maya\2016\scripts\ โดยให้ตั้งชื่อไฟล์เช่นเดียวกับชื่อของ procedure ใหม่ที่เราตั้งให้นั่นคือ createSimpleCrowd.mel เช่นเดียวกับขั้นตอนก่อนหน้า ให้ save คำสั่ง createSimpleCrowd; ไว้บน shelf เมื่อเสร็จสิ้นขั้นตอน

เมื่อเรียบร้อยแล้วเราลองมาดูผลลัพธ์ที่ได้จาก procedures ทั้งสองตัวที่เราเพิ่งสร้างกัน ให้เปิด scene ขึ้นมาใหม่ แล้วเรียก procedure ที่ชื่อ environment ขึ้นมากก่อนเพื่อสร้างสิ่งกีดขวาง จากนั้นให้เรียก createSimpleCrowd เพื่อสร้างวัตถุขึ้นมา สามารถทำได้โดยการป้อนคำสั่งดังนี้

```
environment;
createSimpleCrowd;
```


951301: Advanced Animation and Rendering Techniques

จะได้ผลลัพธ์ตามภาพตัวอย่างด้านล่าง

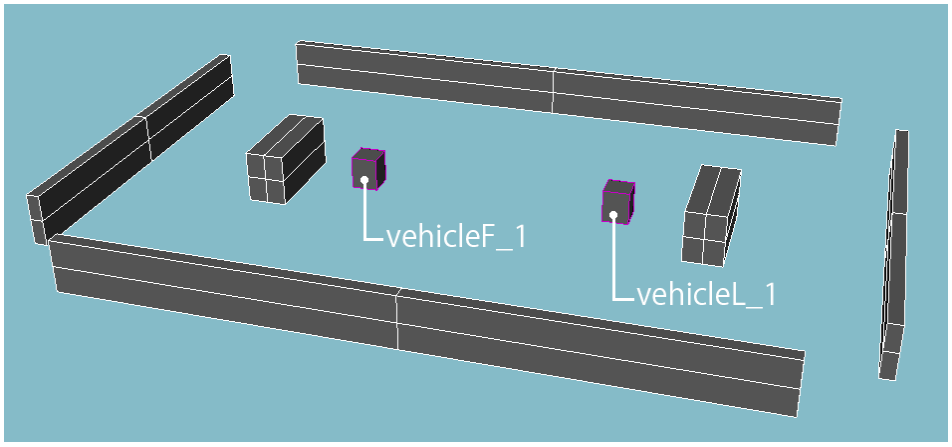

Fig 09-05: แสดงตำแหน่งของ vehicleF_1 และ vehicleL_1 ที่ถูกสร้างมาแทนฝูงชน

ให้ปรับ playback time เป็น 300 frames แล้วลอง play animation ดู เราจะพบปัญหาคือวัตถุวิ่งกินเข้าไปในพื้นผิวของกำแพง (ดังภาพตัวอย่างถัดไป) ทั้งนี้เนื่องจากเรายังไม่ได้สร้างกำแพงให้เป็น passive rigid body และค่า forces ที่ส่งผลกับวัตถุทั้งสองในฉากนั่นเอง

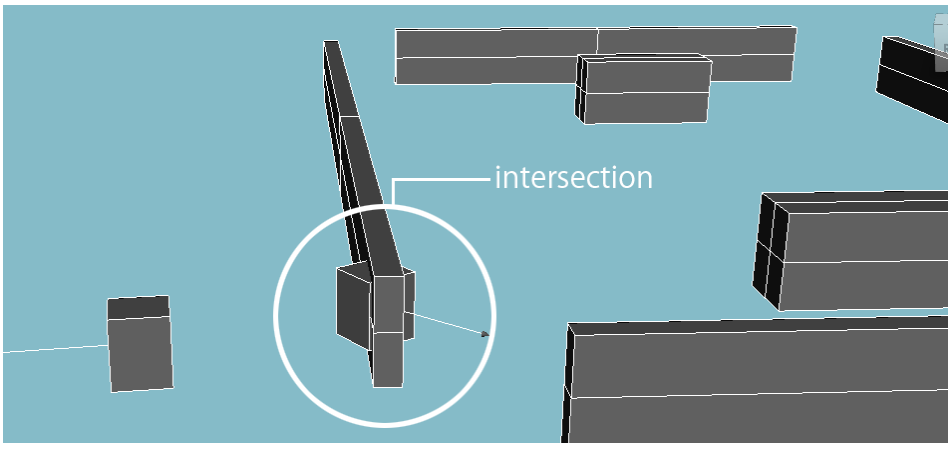

Fig 09-06: ตัวอย่างการเกิด intersection ระหว่างฝูงชนจำลองกับฉาก

ในการที่จะทำให้โปรแกรมเข้าใจว่าวัตถุใดๆไม่สามารถวิ่งผ่านทะลุได้ เราต้องสร้างให้เป็น rigid body ก่อน ให้เปิด script editor ขึ้นมาใหม่แล้วพิมพ์คำสั่งด้านล่าง เพื่อสั่งให้วัตถุที่ถูกเลือกทั้งหมดกลายเป็น passive rigid body

```
// สร้าง global procedure ชื่อว่า collectObstacles
global proc collectObstacles()
{
// สั่งให้เก็บค่าวัตถุทุกตัวที่ถูกเลือกอยู่ไปไว้ใน $collectObjects
string $collectObjects[] = `ls -sl`;
// ใช้ for loop ให้ run จำนวนครั้งเท่ากับจำนวนวัตถุภายใน list
for ( $i = 0; $i < size($collectObjects); $i++)
```


951301: Advanced Animation and Rendering Techniques

```
{
// สั่ง apply passive rigid body ให้กับวัตถุทุกตัวใน list
rigidBody -passive -name ($collectObjects [$i] + "RigidB" +
$i)
-bounciness 2.0 ($collectObjects [$i]);
};
}
```

จากคำสั่งด้านบน เราสามารถ apply passive rigid body ให้กับสิ่งกีดขวางที่เราสร้างโดยการ select สิ่งกีดขวางทั้งหมดที่ต้องการ จากนั้นก็ run คำสั่งด้านบน อย่าลืมว่าเราจะต้องมีสองสิ่งเตรียมไว้อยู่ใน scene ก่อน นั่นคือวัตถุที่เป็น active rigid body (วัตถุ vehicles) ที่ไม่ถูกเลือก และ สิ่งกีดขวาง (obstacles) ที่ถูกเลือกอยู่ เนื่องจากคำสั่งจะ apply rigid body ให้กับวัตถุที่ถูกเลือกอยู่เท่านั้น เพื่อป้องกันความสับสนกับวัตถุอื่นใน scene ที่เราไม่ต้องการ apply เช่นตัว vehicles ทั้งสองเป็นต้น

แทนที่เราจะสั่งสร้าง rigid body ให้กับวัตถุแต่ละตัวในขั้นตอนที่ create วัตถุนั้นๆขึ้นมา การใช้ global procedure ที่จะ apply rigid body ให้กับวัตถุทุกตัวที่ถูก selected อยู่จะมีประโยชน์กว่าอย่างมากในการทำงานจริง เนื่องจากเราสามารถเพิ่มจำนวนกำแพงหรือสิ่งกีดขวางเมื่อไหร่ก็ได้ตามความต้องการ โดยไม่ต้องเสียเวลาในการ apply rigid body ให้กับวัตถุใหม่ทีละตัว เพียงแค่ select วัตถุที่ต้องการแล้วสั่ง run procedure โปรแกรมจะใช้ for loop ในการ apply และ ตั้งชื่อให้กับวัตถุนั้นๆให้เป็น passive rigid body ให้เอง โดยจะตั้งชื่อตามที่เราสั่งไว้แล้ว run หมายเลขลำดับให้ต่อท้ายชื่อนั้นๆเพื่อความสะดวกในการทำงาน

เมื่อเสร็จการสร้าง scripts ด้านบนแล้ว ให้ save scripts ให้อยู่ในรูป dot MEL format และจัดเก็บไว้ที่ C:\...\Documents\maya\2016\scripts\ เช่นกัน โดยให้ตั้งชื่อไฟล์เช่นเดียวกับชื่อของ procedure นั่นคือ collectObstacles.mel และให้ save คำสั่ง collectObstacles; ไว้บน shelf เป็นขั้นตอนสุดท้าย

เมื่อเราเตรียมขั้นตอนทุกอย่างพร้อมแล้ว ให้ลอง *run procedures* ในการสร้าง *crowd system* ดู โดยให้ทำตามขั้นตอนดังนี้

1. เริ่มขั้นตอนจากการสร้างสิ่งกีดขวางด้วยคำสั่ง environment;
2. ตามด้วยการสร้างวัตถุ crowd ด้วยคำสั่ง createSimpleCrowd;
3. ทำการ select วัตถุทั้งหมดที่ต้องการทำให้เป็นสิ่งกีดขวาง จากนั้นใช้คำสั่ง collectObstacles;

951301: Advanced Animation and Rendering Techniques

แล้วลอง play animation ดูจะพบว่าวัตถุทั้งสองจะวิ่งไปมาโดยที่ไม่สามารถวิ่งผ่านทะลุสิ่งกีดขวางได้อีก ให้ทดลองปรับขนาดของสิ่งกีดขวาง หรือเพิ่มจำนวนของสิ่งกีดขวางดู แล้วลองสังเกตพฤติกรรมของวัตถุทั้งสองตัวว่าเป็นอย่างไร

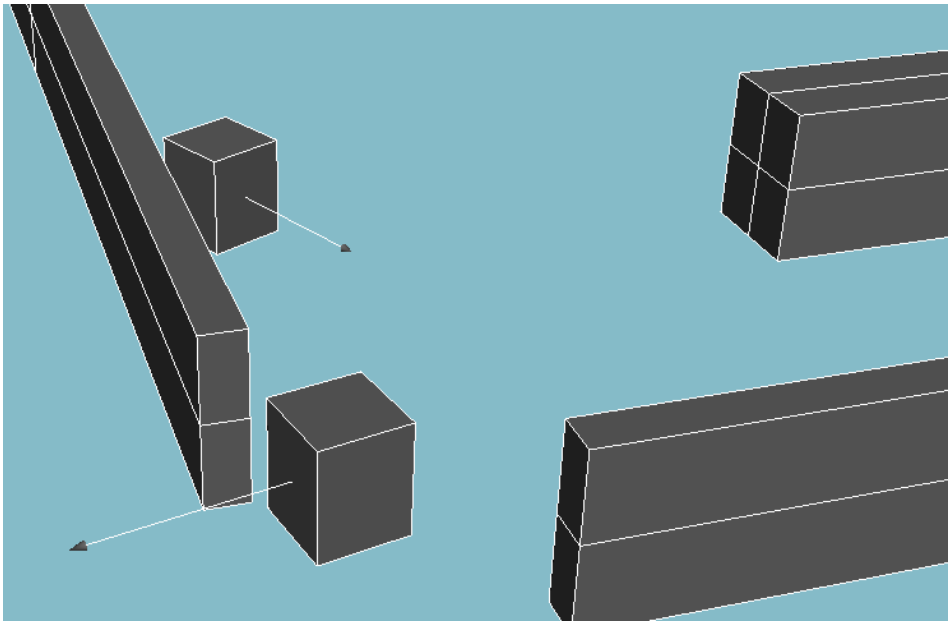

Fig 09-07: แสดงฝูงชนที่ไม่สามารถวิ่งทะลุผ่านกำแพงได้

จากภาพตัวอย่างด้านบน วัตถุทั้งสองจะวิ่งหลบกำแพงทั้งสี่ด้านแล้วหาทางหนีออกไปทางช่องว่าง โดยที่ไม่วิ่งผ่านทะลุกำแพงเหมือนตอนแรก

Creating a Crowd System

ตอนนี้นักศึกษาควรมีความเข้าใจในหลักการที่สำคัญในการจำลองพฤติกรรมให้กับฝูงชนขนาดเล็กกันพอสมควรแล้ว ในส่วนสุดท้ายนี้เราจะนำความรู้ที่ได้มาต่อยอดเพื่อสร้าง crowd system ขนาดใหญ่ดู โดยผลลัพธ์สุดท้ายจะมีลักษณะดังภาพตัวอย่างถัดไป

951301: Advanced Animation and Rendering Techniques

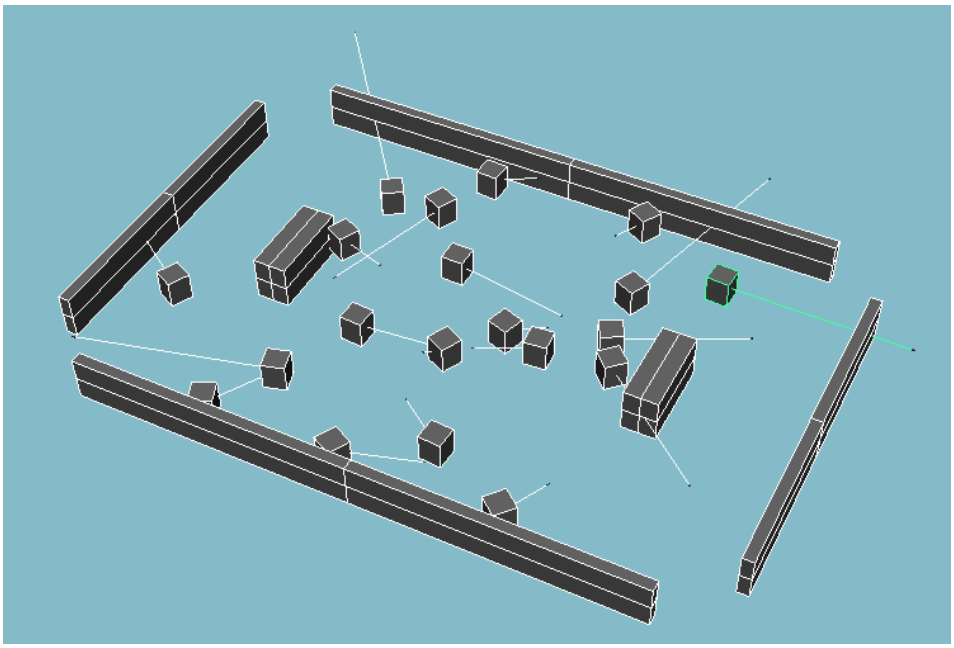

Fig 09-08: ตัวอย่างการจำลองฝูงชนที่มีสมาชิกยี่สิบคนภายในฉากที่สร้าง

เพื่อให้นักศึกษาสามารถประยุกต์ใช้และพัฒนาความรู้ด้วยตนเองได้ต่อไป ในส่วนนี้เราจะเพิ่มเมนูควบคุมให้ผู้ใช้สามารถเลือกขนาดของ crowd ได้ โดยเราจะเริ่มจากพื้นฐานง่ายๆโดยการให้ทางเลือกกับผู้ใช้คือ 10 หรือ 20 vehicles สำหรับ 10 vehicles เราจะกำหนดให้มี leaders จำนวน 2 ตัว และ followers จำนวน 8 ตัว ในขณะที่ 20 vehicles จะประกอบด้วย leaders จำนวน 4 ตัว และ followers จำนวน 16 ตัว

Crowd System

เรามาเริ่มขั้นตอนการทำโดยการสร้าง global procedure crowdSystem เพื่อสร้างหน้าต่าง options ตัวนี้ขึ้นมาดังตัวอย่างด้านล่าง

```
// สร้าง global procedure ที่ชื่อว่า crowdSystem
global proc crowdSystem()
{
```

เพื่อความละเอียดในการทำงาน เราควรให้โปรแกรมตรวจสอบก่อนว่าชื่ออ้างอิงของหน้าต่างที่เราจะสร้าง ถูกสร้างมาก่อนแล้วหรือยัง ถ้าถูกสร้างไว้แล้วให้โปรแกรมลบมันทิ้งไปก่อน เพื่อป้องกันการ error

951301: Advanced Animation and Rendering Techniques

```
    if (`window -exists crowdWin` == 1)
    {
        deleteUI -window crowdWin;
    }
```

เมื่อตรวจสองเสร็จแล้วเราสามารถเริ่มสร้างหน้าต่างของเราได้ โดยให้ตั้งชื่อว่า crowdWin

```
  window
    -title "Crowd System"
    -widthHeight 300 200
    crowdWin;
```

เราจะใช้ column layout ในการจัดเก็บปุ่มส่วนแรกของเรา โดยให้ตั้งชื่อว่า Number of Vehicles ในส่วนนี้เราจะใช้ radio buttons ในการเก็บค่า input จากผู้ใช้ เราต้องการสองปุ่มในที่นี้ให้ผู้ใช้เลือกระหว่าง 10 vehicles กับ 20 vehicles ในการส่งต่อค่าที่ผู้ใช้เลือกไปยังระบบต่อไปสามารถทำได้หลายวิธี ในที่นี้จะใช้ onCommand ใส่ค่าไว้ในตัวแปร $vNumber เพื่อนำค่าจาก $vNumber ไปบอกโปรแกรมว่าจะต้องสร้าง vehicles จำนวนเท่าไหร่ต่อไป

```
  string $form = `columnLayout`;
    string $txt = `text -label "Number of Vehicles"`;
    string $collection = `radioCollection`;
    string $radiob1, $radiob2;
    $radiob1 = `radioButton -label "10" -onCommand "$vNumber
= 10"`;
    $radiob2 = `radioButton -label "20" -onCommand "$vNumber
= 20"`;
    setParent..;
    setParent..;
```

เราจะใช้ row layout ในการจัดเก็บปุ่ม Create และ Cancel และเราจะใช้ $button1 แทนค่าของปุ่ม Create และ button2 แทนค่าของปุ่ม Cancel

```
  rowLayout
    // Create buttons
    -numberOfColumns 2;
    string $button1 = `button -label "Create"`;
    string $button2 = `button -label "Cancel"`;
```

เมื่อเสร็จแล้วเราจะใส่คำสั่งไว้ภายในปุ่ม $button1 และ $button2

```
    button -edit -command
    ("createCrowd $vNumber; deleteUI crowdWin;") $button1;
    button -edit -command ("deleteUI crowdWin;") $button2;
    showWindow crowdWin;
}
```


951301: Advanced Animation and Rendering Techniques

เมื่อเสร็จตามขั้นตอนข้างต้นแล้ว ให้ execute scripts ดู ถ้าไม่พบ error อันใด ให้เรียกคำสั่ง crowdSystem; จาก command line เราจะได้หน้าต่างควบคุมดังภาพตัวอย่างถัดไป

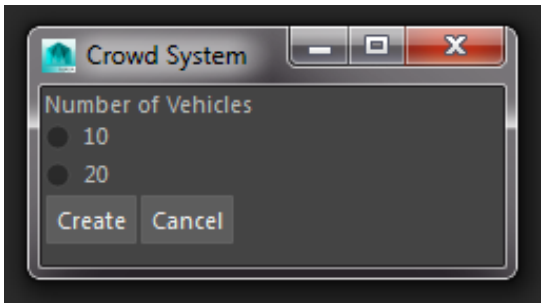

Fig 09-09: แสดงหน้าต่างที่ถูกสร้างขึ้นเพื่อระบุจำนวนสมาชิกในฝูงชน

นอกจากการใช้ flag –onCommand ที่เราใช้ในบทเรียนนี้ เพื่อเรียกค่าที่ผู้ใช้เลือกจากปุ่ม radio button แล้ว เรายังสามารถใช้คำสั่ง radioButton –query ในการเรียกค่าได้อีกด้วย เมื่อได้หน้าต่าง Crowd System แล้ว ให้ทดลองกดเลือกไปที่ 20 vehicles ในหน้าต่างเมนูของเรา จากนั้นในขณะที่ 20 ถูกเลือกอยู่ให้ทดลองพิมพ์คำสั่ง print $vNumber จาก command line แล้ว enter ดู จะพบว่าที่ feedback area จะปรากฏข้อความว่า “20” แปรว่าการเรียกค่า $vNumber จากปุ่ม $button1 และ $button2 ของเราสำเร็จเรียบร้อยแล้วดังภาพตัวอย่างด้านล่าง

เมื่อเสร็จการสร้าง scripts ด้านบนแล้ว ให้ save scripts ให้อยู่ในรูป dot MEL format และจัดเก็บไว้ที่ C:\...\Documents\maya\2016\scripts\ โดยให้ตั้งชื่อไฟล์เช่นเดียวกับชื่อของ procedure นั่นคือ crowdSystem.mel และให้ save คำสั่ง crowdSystem; ไว้บน shelf เป็นขั้นตอนสุดท้าย

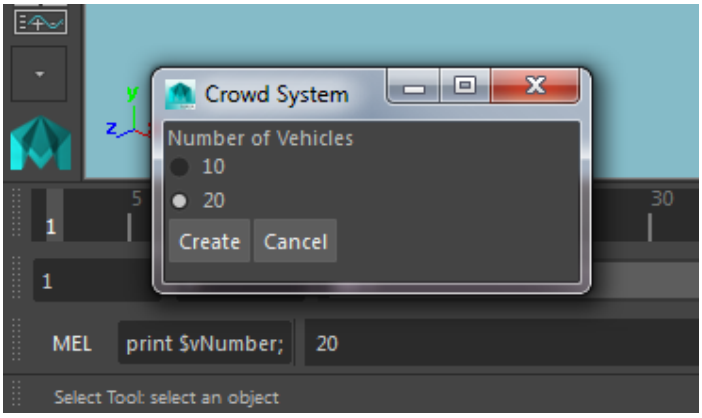

Fig 09-10: ให้สังเกตบริเวณ feedback area จะแสดงจำนวนสมาชิกฝูงชนที่ถูกสร้างขึ้น

951301: Advanced Animation and Rendering Techniques

เมื่อเราเตรียมความพร้อมของเมนูเรียบร้อยแล้ว ขั้นต่อไปเราจะมาเพิ่มจำนวนของ crowd vehicles จากเดิม 2 vehicles เป็น 10 และ 20 vehicles กัน นอกจากการแก้ไขคำสั่งในเรื่องของจำนวน vehicles แล้ว เรายังต้องเพิ่มในส่วนของการ random position ให้กับ vehicles ที่จะสร้างด้วย เมื่อตรวจสอบจาก procedure environment ที่เราสร้างไว้ เราจะพบว่าถ้าเราต้องการ random ให้ vehicles ทุกตัวมีจุดกำเนิดอยู่ภายในกำแพงทั้งสี่ด้านนั้น เราควรจะหว่านมันลงไปภายในพื้นที่ระหว่าง -17 และ 17 units และเรื่องสุดท้ายที่เราต้องระวังนั่นคือเราต้องการค่า $vNumber จาก procedure crowdSystem (หน้าต่างเมนูที่เราเพิ่งสร้างมาก่อนหน้านี้) มาใช้ ในการส่งผ่านค่าตัวแปรระหว่าง procedures เราจะต้องใส่ชื่อตัวแปรนั้นไว้ในวงเล็บ () ที่ตามหลังชื่อของ procedure ตัวที่เราต้องการส่งค่าให้ด้วย เรามาเริ่มทำขั้นตอนการสร้าง crowd vehicles เพิ่มกัน

Create Crowd

ก่อนอื่นเราจะเริ่มจากการสร้าง procedure ตัวใหม่ชื่อว่า createCrowd() แต่ในคราวนี้เราจะต้องใส่ int $vNumber เข้าไปในวงเล็บข้างหลังด้วย เพื่อเป็นการส่งค่า $vNumber จาก procedure อื่นเข้ามาใน procedure createCrowd อันนี้

```
global proc createCrowd(int $vNumber)
{
```

เริ่มจากการสร้าง master crowd solver ให้กับ vehicles ทุกตัว ให้ตั้งชื่อว่า crowdSolver และกำหนด velocity เป็น 0.5 พร้อมกับเปิดการแสดงผลทิศทางการเคลื่อนที่ (ลูกศร) และให้เปิดการทำงานของการ disconnection เพื่อให้ vehicles สามารถหันหน้าตามทิศทางที่จะเคลื่อนที่ได้

```
    // Vehicle creation for crowd system
    // Create a master crowdSolver for all vehicles
    rigidSolver -create -current -name crowdSolver
    -velocityVectorScale 0.5 -displayVelocity on;
    setAttr crowdSolver.allowDisconnection 1;
```

ขั้นต่อมาเราจะประกาศตัวแปรแบบ integer ขึ้นมาใหม่สองตัว เพื่อบอกโปรแกรมว่าเราจะใช้สัดส่วนระหว่าง leaders และ followers อย่างไร โดยให้ $FvNumber แทนจำนวนของ followers และ $LvNumber แทนจำนวนของ leaders

```
    // Get total number of vehicles from interface options
    int $FvNumber;
    int $LvNumber;
```


951301: Advanced Animation and Rendering Techniques

เราใช้ switch statement ในการจัดการกับค่าของ $vNumber ที่เรารับมา เนื่องจากค่าที่ได้จะมีแค่สองค่าคือถ้าไม่ใช่ 10 ก็จะเป็น 20 โดยเราจะสั่งให้ใส่ค่าสัดส่วนระหว่าง leaders และ followers ตามค่า $VNumber ที่ได้มา

```
switch ($vNumber)
    {
        case 10:
        $FvNumber = 8; // Followers
        $LvNumber = 2; // Leaders
        break;

        case 20:
        $FvNumber = 16; // Followers
        $LvNumber = 4; // Leaders
        break;
    }
```

เมื่อได้จำนวนตัวเลขทุกอย่างพร้อมแล้ว เราจะมาเริ่มสร้าง vehicles กัน โดยเริ่มจากการสร้าง followers ก่อน เนื่องจาก crowd system ของเรามีขนาดใหญ่ เราสามารถใช้ for loop เพื่อให้โปรแกรมจัดการสร้าง vehicles ซ้ำๆแทนเรา โดยกำหนดให้สร้าง followers เท่ากับค่าของ $FvNumber และเราจะให้โปรแกรม run เลขข้างหลังชื่อของ vehicle (vehicleF_) ตามลำดับที่สร้าง

```
    for ($i = 0; $i < $FvNumber; $i++)
    {
        polyCube -name ("vehicleF_" + $i) -width 2 -height
2.5 -depth 2;
```

จากนั้นก็เริ่มต้นสร้าง force fields ให้กับ followers ส่วนนี้จะคล้ายกับ scripts ในบทที่ 7 เพียงแต่เราจะใช้ตัวแปร $i พ่วงเข้าไปท้ายชื่อของ follower แต่ละตัว

```
        // Add vehicle force field for follower
        radial -position 0 0 0 -name ("vehicleFForce_" + $i)
        -magnitude 50 -attenuation 0.3 -maxDistance 8.0;
        parent ("vehicleFForce_" + $i) ("vehicleF_" + $i);
        hide ("vehicleFForce_" + $i);
```

ขั้นตอนต่อไปเราจะทำให้ followers ทุกตัวเป็น rigid body พร้อมกับการ randomise ตำแหน่งในการกำเนิดของ follower แต่ละตัว อย่าลืมว่าขนาดของกำแพงที่เราสร้างไว้ใน procedure environment คือเท่าไหร่ จัดวาง followers ทุกตัวให้อยู่ภายในเนื้อที่นั้น ในที่นี้คือเนื้อที่ระหว่าง -17 กับ 17 units

951301: Advanced Animation and Rendering Techniques

```
        // Make vehicle a rigid body with random placement
        rigidBody -name ("rigidVehicleF_" + $i) -active -
mass 1 -bounciness 0 -damping 1.5
        -position (rand(-17,17)) 0 (rand(-17,17)) -impulse
0 0 0 -standInObject cube
        -solver crowdSolver ("vehicleF_" + $i);
```

เพื่อให้ followers หันหน้าตามทิศทางที่เคลื่อนที่ เราต้องทำการ disconnect attributes ให้กับแกนทั้งสามแกนของ followers (X, Y, Z)

```
        disconnectAttr ("rigidVehicleF_" + $i + "ry.output")
("vehicleF_" + $i + ".rotateY");
        disconnectAttr ("rigidVehicleF_" + $i + "rx.output")
("vehicleF_" + $i + ".rotateX");
        disconnectAttr ("rigidVehicleF_" + $i + "rz.output")
("vehicleF_" + $i + ".rotateZ");
```

ต่อไปคือการใช้ expression ในการควบคุมพฤติกรรมของ follower แต่ละตัว เพื่อสร้างความสมจริงให้กับ crowd system ขนาดใหญ่ เราควรเพิ่มค่า random เข้าไปในสมการการเคลื่อนที่ของแต่ละตัวด้วยเพื่อสร้างความเป็นเอกลักษณ์ให้กับ follower แต่ละตัว ในที่นี้เราควรทำให้ followers มีพฤติกรรมแตกต่างกันเพียงเล็กน้อย เนื่องจากถ้าแตกต่างกันมากเกินไป อาจทำให้เกิด chaos ขึ้นได้ ในที่นี้เราจะสร้างค่า random ให้อยู่ระหว่าง -3 กับ 3

```
        float $randX = rand(-3,3);
        float $randZ = rand(3,-3);
        $expString = ("rigidVehicleF_" + $i + ".impulseX");
        $expString += " = sin(time * ";
        $expString += ($randX);
        $expString += ");\n";
        $expString += ("rigidVehicleF_" + $i + ".impulseZ");
        $expString += " = (noise(time) * ";
        $expString += ($randZ);
        $expString += ");\n\n";
        $expString += "float $fVel" + $i + "[];\n";
        $expString  +=  "$fVel"  +  $i  +"  =  `getAttr
rigidVehicleF_" + $i + ".velocity`;\n\n";
        $expString += "vehicleF_" + $i + ".rotateX = 0;\n";
        $expString  +=  "vehicleF_"  +  $i  +  ".rotateY  =
atan2d($fVel" + $i + "[0], $fVel" + $i + "[2]);\n";
        $expString += "vehicleF_" + $i + ".rotateZ = 0;\n";

        expression -s $expString -name ("wanderF_exp" + $i)
        -alwaysEvaluate true -unitConversion all;
    }
```


951301: Advanced Animation and Rendering Techniques

เป็นอันสิ้นสุดขั้นตอนการสร้าง followers ต่อไปคือขั้นตอนการสร้าง leaders ซึ่งจะมีลักษณะคล้ายกับการสร้าง followers แตกต่างกันที่ force fields ที่ apply เรามาเริ่มต้นจากการดึงค่าจาก $LvNumber มาใช้ใน for loop เพื่อกำหนดสัดส่วนของ leaders ที่สร้างให้เหมาะสมกับ followers

```
    for ($i = 0; $i < $LvNumber; $i++)
    {
        polyCube -name ("vehicleL_" + $i) -width 2 -height
2.5 -depth 2;
```

ขั้นต่อมาคือการสร้าง force field ให้กับ leaders ด้วยคำสั่ง radial ตั้งชื่อของ force field แต่ละตัวว่า vehicleLForce_ ตามด้วย $i เพื่อแทนหมายเลขลำดับของแต่ละ leader จากนั้นก็ทำการเชื่อม force field แต่ละตัวเข้ากับ leaders ที่สร้างโดยให้จับคู่ตามลำดับ $i

```
        radial -position 0 0 0 -name ("vehicleLForce_" + $i)
-magnitude 50
        -attenuation 0.3 -maxDistance 8.0;
        parent ("vehicleLForce_" + $i) ("vehicleL_" + $i);
        hide ("vehicleLForce_" + $i);
```

เพื่อสร้างแรงดึงดูดที่ส่งผลต่อ followers เราจะต้องทำการสร้าง leader fields ให้กับ leader แต่ละตัวด้วย โดยให้ตั้งชื่อว่า vehicleGlobalLeadForce

```
   radial -position 0 0 0 -name ("vehicleGlobalLeadForce_"
+ $i ) –magnitude -1
        -attenuation 0.2 -maxDistance 50;
        parent ("vehicleGlobalLeadForce_"+ $i) ("vehicleL_"
+ $i);
        hide ("vehicleGlobalLeadForce_"+ $i);
```

จากนั้นเราจะทำให้ leaders ทุกตัวเป็น rigid body และให้ randomise ตำแหน่งในการเกิดของแต่ละตัวภายในพื้นที่เช่นเดียวกับ followers นั่นคือการ random ภายในพื้นที่ -17 ถึง 17 units

```
        rigidBody -name ("rigidVehicleL_" + $i) -active -
mass 1 -bounciness 0
        -damping 1.5 -position
        (rand(-17,17)) 0 (rand(-17,17)) -impulse 0 0 0 -
standInObject cube
        -solver crowdSolver ("vehicleL_" + $i);
```

ทำการ disconnect attributes ให้กับ leaders ทุกตัวจากแนวบังคับ rotation X, Y, Z เพื่อให้สามารถหันหน้าตามทิศทางที่เคลื่อนที่ได้

951301: Advanced Animation and Rendering Techniques

```
        disconnectAttr ("rigidVehicleL_" + $i + "ry.output")
("vehicleL_" + $i + ".rotateY");
        disconnectAttr ("rigidVehicleL_" + $i + "rx.output")
("vehicleL_" + $i + ".rotateX");
        disconnectAttr ("rigidVehicleL_" + $i + "rz.output")
("vehicleL_" + $i + ".rotateZ");
```

ขั้นต่อมาคือการสร้าง expression เพื่อกำหนดพฤติกรรมการเคลื่อนที่ของ leader แต่ละตัว เช่นเดียวกับ followers เราจะเพิ่มค่า random เข้าไปในสมการเพื่อสร้างความเป็นเอกลักษณ์ให้กับ leader แต่ละตัว โดยให้กำหนดค่า random อยู่ระหว่าง -3 และ 3

```
    float $randX = rand(-3,3);
        float $randZ = rand(3,-3);

        $expString = ("rigidVehicleL_" + $i + ".impulseX");
        $expString += " = sin(time * ";
        $expString += ($randX);
        $expString += ");\n";
        $expString += ("rigidVehicleL_" + $i + ".impulseZ");
        $expString += " = (noise(time) * ";
        $expString += ($randZ);
        $expString += ");\n\n";
        $expString  += "// Set orientation of vehicle
according to the velocity direction.\n";
        $expString += "float $fVelL" + $i + "[];\n";
        $expString  +=  "$fVelL"  +  $i  +"  =  `getAttr
rigidVehicleL_" + $i + ".velocity`;\n\n";
        $expString += "vehicleL_" + $i + ".rotateX = 0;\n";
        $expString += "vehicleL_" + $i + ".rotateY = atan2d
( $fVelL" + $i + "[0], $fVelL" + $i + "[2] );\n";
        $expString += "vehicleL_" + $i + ".rotateZ = 0;\n";

        expression -s $expString -name ("wanderL_exp" + $i)
        -alwaysEvaluate true -unitConversion all;
    }
```

ถึงตรงนี้คือสิ้นสุดการสร้าง leaders และ followers พร้อมทั้งการจัดวางตำแหน่งของทุก vehicles ในพื้นที่ๆต้องการ ซึ่งถือว่าเกือบจะเสร็จสิ้นขั้นตอนการทำงานแล้ว เหลือเพียงการ connect dynamics ต่างๆเข้ากับ vehicles ซึ่งมีขั้นตอนดังนี้

```
// ใช้ for loop ในการทำซ้ำ
    for ($i = 0; $i < $LvNumber; $i++ )
    {
// ทำการ connect dynamic ให้กับ leaders เข้ากับ followers
        for ( $j = 0; $j < $FvNumber; $j++)
        {
            connectDynamic -fields ("vehicleLForce_" + $i)
("rigidVehicleF_" + $j);
```


951301: Advanced Animation and Rendering Techniques

```
            connectDynamic                              -fields
("vehicleGlobalLeadForce_" + $i) ("rigidVehicleF_" + $j);
        }
    }

    for ($i = 0; $i < $FvNumber; $i++)
    {
// ทำการ connect dynamic ให้กับ followers เข้ากับ leaders
        for ($j = 0; $j < $LvNumber; $j++)
        {
            connectDynamic -fields ("vehicleFForce_" + $i)
("rigidVehicleL_" + $j) ;
        }
    }

        for ($i = 0; $i < $LvNumber; $i++)
    {
// ทำการ connect dynamic ให้กับ leaders เข้ากับ leaders ด้วยกัน
        for ($j = 0; $j < $LvNumber; $j++)
        {
            connectDynamic -fields ("vehicleLForce_" + $i)
("rigidVehicleL_" + $j) ;
        }
        connectDynamic -delete -fields ("vehicleLForce_" +
$i) ("rigidVehicleL_" + $i);
    }

    for ($i = 0; $i < $FvNumber; $i++)
    {
// ทำการ connect dynamic ให้กับ followers เข้ากับ followers ด้วยกัน
        for ($j = 0; $j < $FvNumber; $j++)
        {
            connectDynamic -fields ("vehicleFForce_" + $i)
("rigidVehicleF_" + $j) ;
        }
    connectDynamic -delete -fields ("vehicleFForce_" + $i)
("rigidVehicleF_" + $i);
    }
}
```

เมื่อเสร็จการสร้าง scripts ด้านบนแล้ว ให้ save scripts ให้อยู่ในรูป dot MEL format และจัดเก็บไว้ที่ C:\...\Documents\maya\2016\scripts\ โดยให้ตั้งชื่อไฟล์เช่นเดียวกับชื่อของ procedure นั่นคือ createCrowd.mel เป็นอันเสร็จสิ้นขั้นตอนการสร้างทั้งหมด

951301: Advanced Animation and Rendering Techniques

ให้ตรวจดูให้แน่ใจว่าเราได้สร้าง procedures ทุกตัวครบแล้ว ประกอบด้วย

1. proc environment()
2. proc collectObstacles()
3. proc createCrowd()
4. proc crowdSystem()

จากนั้นให้ปิดและเปิดโปรแกรมขึ้นมาใหม่และเราสามารถสร้าง crowd system ได้ด้วยการทำตามขั้นตอนดังนี้

1. run คำสั่ง environment; เพื่อสร้างสิ่งกีดขวาง
2. run คำสั่ง crowdSystem; เพื่อเรียกหน้าต่างสร้าง crowd system
3. เลือก option ขนาดของ crowd ที่ต้องการแล้วกดปุ่ม Create
4. Select สิ่งกีดขวางทุกตัวที่ต้องการ (ob1 – 0b6)
5. Run คำสั่ง collectObstacles; เพื่อทำให้สิ่งกีดขวางที่เลือกเป็น passive rigid body
6. เสร็จสิ้นขั้นตอนการสร้าง ทดลองกดปุ่ม play animation เพื่อตรวจสอบผลการสร้าง

951301: Advanced Animation and Rendering Techniques

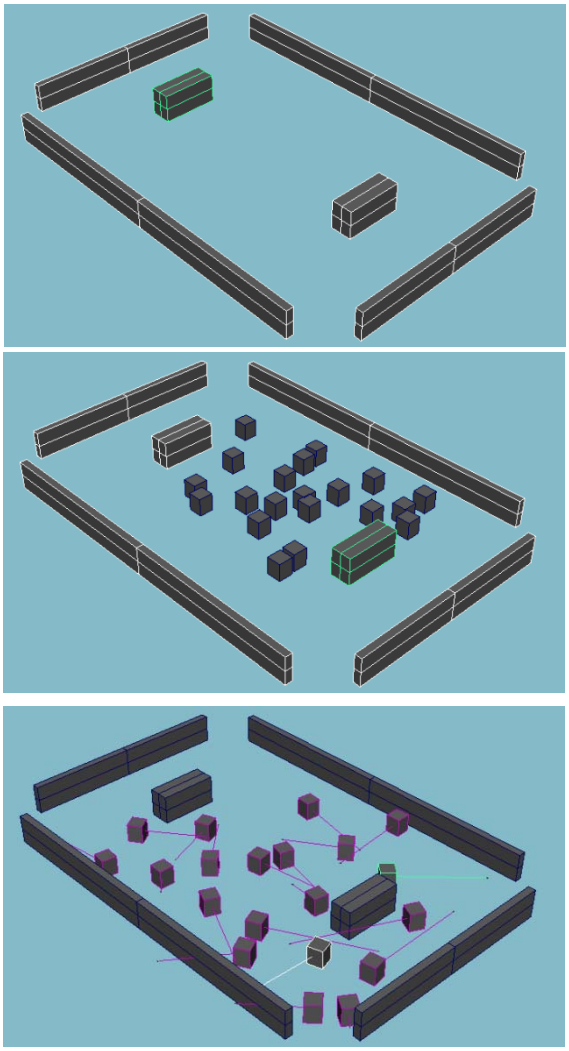

Fig 09-11: แสดงลำดับและขั้นตอนของการจำลองฝูงชนโดยเริ่มจากการสร้างฉาก การกำหนดจำนวนสมาชิกของฝูงชน และการจำลองพฤติกรรมของฝูงชน

951301: Advanced Animation and Rendering Techniques

Reference

- Wilkins, M. R. and Kazmier, C. (2005) *MEL Scripting for Maya Animations*, Morgan Kaufmann Publishers, Elsevier Inc.
- Galanakis, R. (2014) *Practical Maya Programming with Python*, Packt Publishing Ltd.
- Stripinis, D. (2003) *The MEL Companion: Maya Scripting 3D Artists*, Charles River Media, INC