Chapter 7:
Introduction to Windows Structure

Topics:

Part I: window Structures design, window identification, and UI template

Part II: workshops;
workshop I: creating an object rename window
workshop II: creating an object selecting and listing window

Chapter 7:
Introduction to Windows Structure

เอกสารประกอบการเรียน
รายวิชา ANI 951301
สาขาวิชาแอนนิเมชันและเกม
วิทยาลัยศิลปะ สื่อ และเทคโนโลยี
มหาวิทยาลัยเชียงใหม่

951301: Advanced Animation and Rendering Techniques

วัตถุประสงค์

1. สามารถสร้างและควบคุม particles ด้วย MEL scripts ได้
2. เข้าใจถึงการตั้ง particle goals ให้กับ particles เพื่อสร้างความสัมพันธ์ระหว่าง particles ได้
3. สามารถปรับประเภทและค่าคุณลักษณะให้กับ particles ให้สอดคล้องกับความต้องการ
4. เรียนรู้เรื่องของ interaction ระหว่าง particles กับ surfaces ของวัตถุ ด้วย solid body dynamics
5. ใช้การผสมผสานการเคลื่อนที่ของ particles และการเคลื่อนที่ของ solid body dynamics ให้เกิดผลตามต้องการได้

เนื้อหาการสอน

ส่วนที่ 1: การออกแบบและการวางโครงสร้างหน้าต่างการทำงาน การสร้างข้อมูลในการจำแนกหน้าต่างการทำงาน และการใช้ UI template เพื่อช่วยในการทำงาน

ส่วนที่ 2: การสร้างหน้าต่างเพื่อใช้ในการเปลี่ยนชื่อให้กับวัตถุจำนวนมากพร้อมกัน และการสร้างหน้าต่างเพื่อใช้ในการสร้างรายชื่อและเลือกวัตถุจำนวนมาก

Introduction to Windows Structures

ในการออกแบบหน้าต่างการทำงาน สิ่งแรกที่เราควรคำนึงถึงคือการออกแบบโครงสร้างของหน้าต่างที่จะสร้างขึ้นมา เราสามารถทำได้โดยการร่างภาพแบบคร่าวๆของสิ่งที่เราต้องการออกแบบเพื่อแบ่งโครงสร้างและองค์ประกอบของมันก่อนเริ่มลงมือเขียนโปรแกรม เนื่องจากในการสร้างหน้าต่างการทำงานขึ้นมามีความซับซ้อน เราต้องเข้าใจถึงการจัดเก็บองค์ประกอบต่างๆภายในเป็น layers อย่างเป็นระบบ หัวใจของการออกแบบคือการกำหนดความสัมพันธ์ (hierarchy) ให้กับ layouts และ controls ของเราว่าควรมีลำดับขั้นและหน้าตาเป็นอย่างไร เรามาเริ่มทำความเข้าใจการออกแบบโครงสร้างจากตัวอย่างด้านล่างกัน

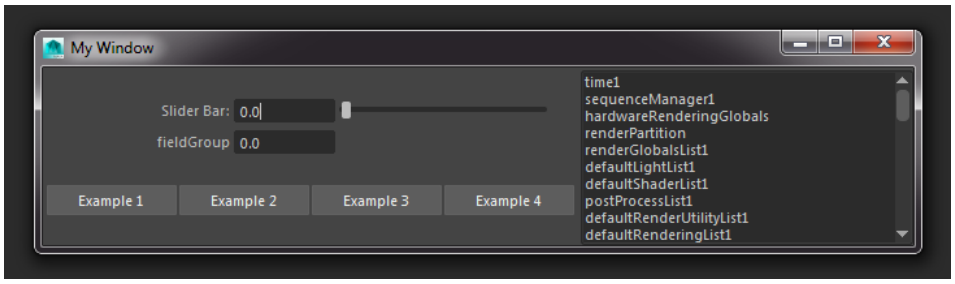

Fig 07-01: ตัวอย่างหน้าต่างที่ประกอบด้วย slider bar, field group, ปุ่มควบคุม, และ text list

สมมุติว่านี่คือหน้าต่างที่เราต้องการสร้าง ขั้นแรกเราสามารถแบ่งหน้าต่างออกเป็นสองส่วนใหญ่ๆ คือส่วนด้านซ้ายและส่วนด้านขวา ซึ่งส่วนด้านซ้ายจะจัดเก็บ slider bar, field group และปุ่มสี่ปุ่ม ในขณะที่ส่วนด้านขวาจะจัดเก็บ text list

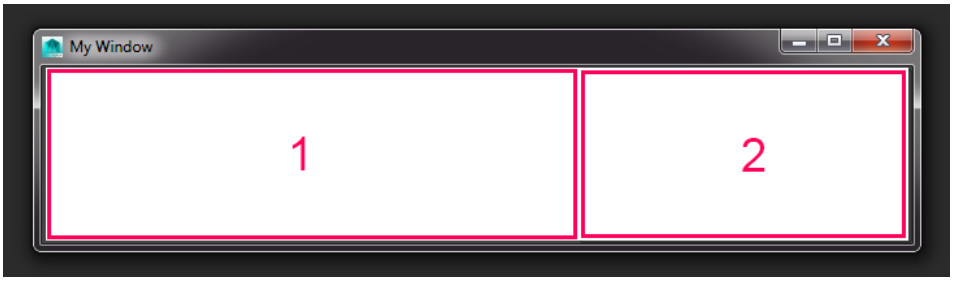

Fig 07-02: หน้าต่างสามารถแบ่งเนื้อที่การแสดงผลเป็นสองส่วนหลักๆ คือส่วนที่ 1 กับส่วนที่ 2

ถ้าเราเริ่มจากหน้าต่างเปล่าๆ เราสามารถใช้ row layout สร้างเป็น layer ที่ 1 เพื่อแบ่งพื้นที่ว่างนี้เป็นสองส่วนตามแนวนอนดังภาพตัวอย่างด้านบน เป็น layer ชั้นแรกที่เราสร้างขึ้นมา จากนั้นเราลองมาดูกันทีละส่วน โดยเริ่มจากส่วนที่ 1 ด้านซ้ายมือกันก่อน จากภาพตัวอย่างแรก ทีนี้เราลองมาแบ่งพื้นที่ในส่วนนี้ตามความต้องการใช้งานของเราดู

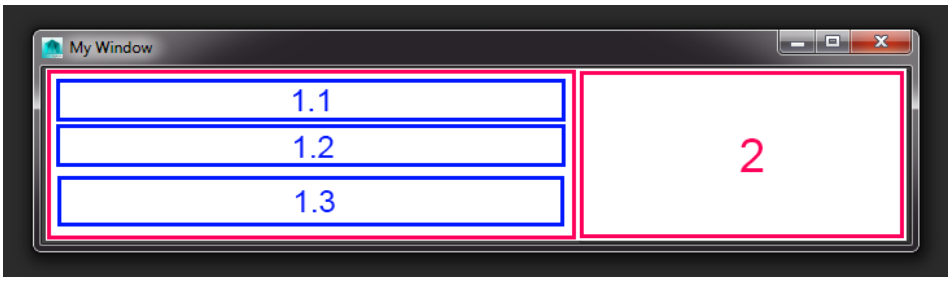

Fig 07-03: เราสามารถแบ่งส่วนที่หนึ่งออกเป็นส่วนย่อยที่ 1.1, 1.2, และ 1.3

จะสังเกตได้ว่าเราสามารถแบ่งพื้นที่ออกเป็นสามส่วนตามภาพตัวอย่าง คือ 1.1, 1.2 และ 1.3 โดยทั้งสามส่วนจะถูกสร้างด้วย column layout ขึ้นมาเป็น layer ที่ 2 ข้างใน row layout ชั้นแรก แต่เนื่องจาก 1.3 เราต้องการสร้างปุ่มสี่ปุ่มเรียงกันตามแนวนอน เราจึงต้องสร้าง row layout ขึ้นมาเป็น layer ที่ 3 ข้างใน 1.3 (layer 2) เพื่อจัดเก็บปุ่มทั้งสี่นี้ดังภาพตัวอย่างด้านล่าง

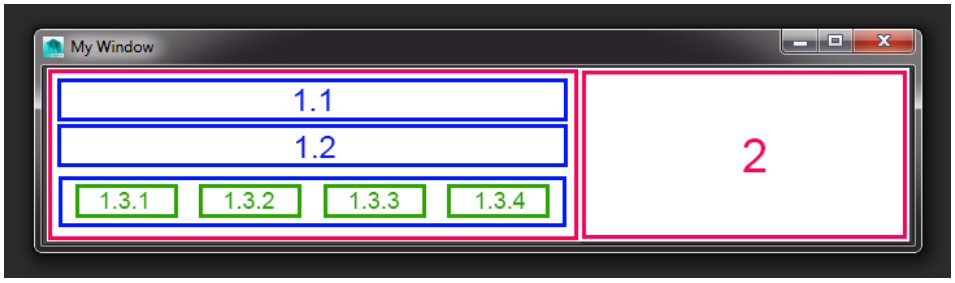

Fig 07-04: ในส่วนของส่วนย่อยที่ 1.3 ยังสามารถถูกแบ่งออกเป็นส่วนย่อยได้อีกสี่ส่วน

951301: Advanced Animation and Rendering Techniques

จากตัวอย่างข้างต้นเราสามารถสร้างแผงผังความสัมพันธ์ขององค์ประกอบหน้าต่างของเราเป็นลำดับขั้นได้ดังนี้

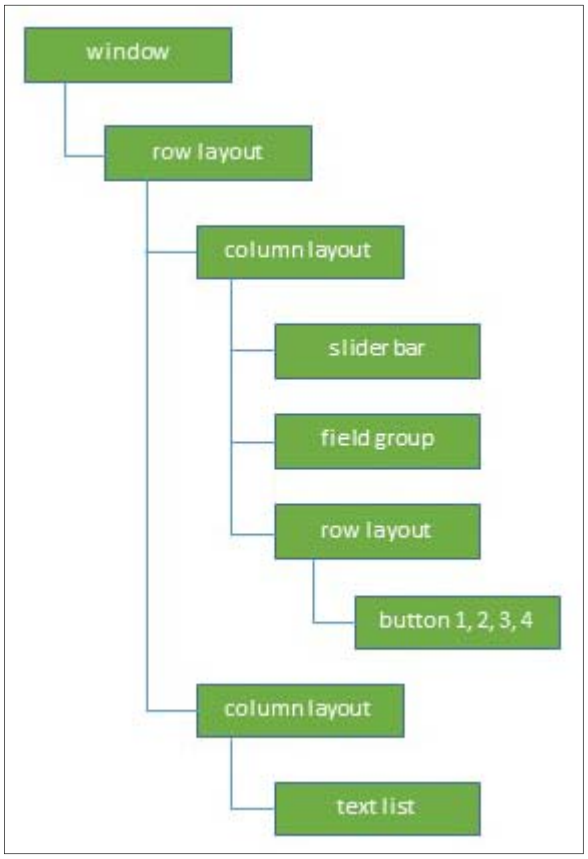

Fig 07-05: แสดงตัวอย่างแผนผังความสัมพันธ์ขององค์ประกอบหน้าต่างแบบลำดับขั้น

ในการเขียนโปรแกรมเราจะใช้คำสั่ง setParent.. ในการย้อนกลับออกมาหนึ่ง layer เช่นเมื่อเราอยู่ใน layer ที่สาม (button 1, 2, 3, 4) เราต้องสั่ง setParent..; สองครั้ง เพื่อกลับมาที่ row layout (layer ที่หนึ่ง) ก่อนที่เราจะสร้าง column layout ให้กับ text list ได้ เราควรออกแบบโครงสร้างและลำดับขั้นความสัมพันธ์ดังตัวอย่างที่ให้มาทุกครั้งก่อนลงมือสร้างหน้าต่าง

Window Title

เราควรตั้งชื่อให้กับหน้าต่างที่เราสร้างทุกครั้ง เพื่อประโยชน์ในการอ้างอิง หรือเรียกใช้ในโอกาสข้างหน้า ชื่อของหน้าต่างแตกต่างกับ title ของหน้าต่าง title คือข้อความที่จะปรากฏอยู่ด้านบน

951301: Advanced Animation and Rendering Techniques

หน้าต่างสามารถทำได้ด้วย flag –title (ที่เราใช้ในบทที่ผ่านมา) ในขณะที่ชื่ออ้างอิงของหน้าต่างจะไม่ต้องใช้ flag แต่ให้ใส่อยู่ด้านหลังสุดของคำสั่งสร้างหน้าต่างทั้งหมด ดังตัวอย่างด้านล่าง

```
window
-title "This is a Title not a Name" windowName;
```

จากคำสั่งด้านบนเราจะสามารถสร้างหน้าต่างที่มีข้อความว่า This is a Title not a Name และมีชื่ออ้างอิงว่า windowName ดังตัวอย่างด้านล่าง

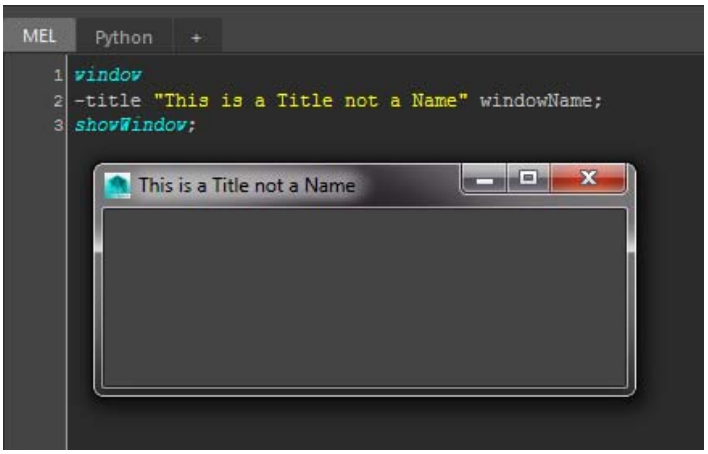

Fig 07-06: ตัวอย่างการระบุ window title

ให้นักศึกษาลองสร้างหน้าต่างขึ้นมาอีกสักสามหน้าต่างโดยให้มีชื่ออ้างอิงไม่ซ้ำกันเช่น windowName1, windowName2, windowName3 โดยที่แต่ละหน้าต่างให้มี title ที่ไม่ซ้ำกันด้วย เมื่อทำเสร็จแล้วให้ปิดหน้าต่างที่สร้างใหม่ทั้งหมด จากนั้นให้ทดลองเรียก render หน้าต่างที่สร้างด้วยการอ้างอิงจากชื่อของมันดู ดังตัวอย่างด้านล่าง

```
showWindow windowName1;
```

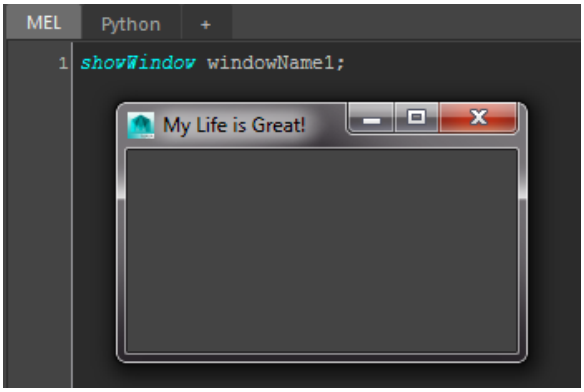

Fig 07-07: ตัวอย่างการเรียกแสดงผลหน้าต่างโดยการระบุ window title

951301: Advanced Animation and Rendering Techniques

จะพบว่าเราสามารถใช้คำสั่ง showWindow เรียกใช้หน้าต่างแต่ละตัวได้โดยการอ้างอิงจากชื่ออ้างอิงของมันนั่นเอง ในการทำงานนั้น เราสามารถเรียกค่าต่างๆได้โดยการระบุค่าที่ต้องการจากชื่อของหน้าต่างใดๆ หรือการอ้างอิงจากค่าตัวแปรที่เราจัดเก็บในหน้าต่างนั้นได้

จากที่กล่าวมาจะพบว่าชื่ออ้างอิงของหน้าต่างมีความสำคัญในการทำงานเป็นอย่างมาก เสมือนเป็น ID ของหน้าต่างนั้นๆ ดังนั้นเราจะไม่สามารถสร้างหน้าต่างที่มีชื่ออ้างอิงซ้ำกันได้เลย ถ้าเราตั้งชื่ออ้างอิงซ้ำให้กับหน้าต่างที่สร้าง เราจะได้รับข้อความเตือนว่า name is not unique ดังตัวอย่างด้านล่าง

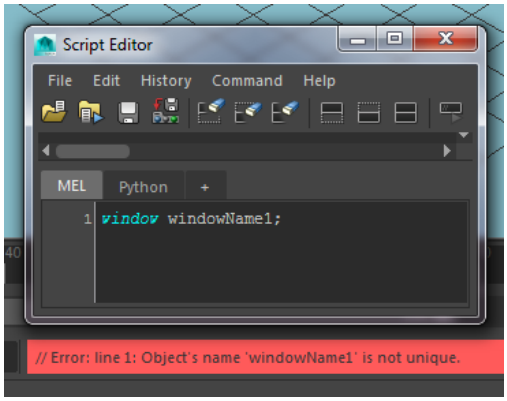

Fig 07-08: ตัวอย่างการแสดงผล error เมื่อมีการระบุ window title แบบซ้ำซ้อน

ในกรณีที่เราต้องการลบชื่ออ้างอิงของหน้าต่างที่เราสร้างไปแล้ว สามารถทำได้โดยการใช้คำสั่ง deleteUI ตามด้วยชื่ออ้างอิงของหน้าต่างที่ต้องการลบ เช่นถ้าเราต้องการลบหน้าต่าง windowName1จะมีรูปแบบคำสั่งดังนี้

```
deleteUI windowName1;
```

ข้อมูลทุกอย่างจะถูกลบออกไปพร้อมกับหน้าต่างตัวนั้น ดังนั้นควรตรวจสอบให้แน่ใจว่าเราต้องการลบออกไปหรือไม่ ในกรณีที่การสร้างหน้าต่างอยู่ใน flow ของการเขียนโปรแกรม (เราไม่สามารถตรวจสอบได้ด้วยตนเองว่าชื่อนี้ถูกใช้ไปแล้วหรือยัง) เราสามารถใช้ if ในการตรวจสอบว่าหน้าต่างนั้นถูกสร้างหรือยัง ถ้าถูกสร้างแล้วให้ลบมันออกแล้วสร้างตัวใหม่ขึ้นมาแทน สามารถทำได้ดังตัวอย่างด้านล่าง

```
if (`window -exists windowName1`) // ตรวจสอบว่าหน้าต่างชื่อ windowName1 ถูกสร้างมาแล้วหรือยัง
    deleteUI windowName1;// ถ้าสร้างแล้วให้ลบ windowName1 ทิ้ง
```


951301: Advanced Animation and Rendering Techniques

UI Template

ในการสร้างองค์ประกอบต่างๆให้กับหน้าต่าง เราต้องกำหนดคุณลักษณะให้กับมันทุกครั้งที่สร้าง เช่นขนาดของปุ่มแต่ละอัน การจัดวางตำแหน่งของข้อความและปุ่มต่างๆ ขนาดของ layouts แบบต่างๆเป็นต้น ซึ่งในหลายๆครั้งเราจะใช้การตั้งค่าแบบใดแบบหนึ่งเป็นประจำ เราสามารถลดขั้นตอนนี้ได้ด้วยการสร้าง template ให้กับ UI ของเราขึ้นมา แล้วเรียกใช้ template ตัวนี้เมื่อใดก็ตามที่เราต้องการค่าแบบที่สร้างไว้ ในการสร้าง UI template สามารถทำได้ด้วยคำสั่ง uiTemplate ตามด้วยชื่อของ template ที่เราตั้งดังตัวอย่างด้านล่าง

```
uiTemplate myTemplate;
// เป็นการสร้าง template ชื่อว่า myTemplate ขึ้นมา
```

ขั้นตอนต่อไปเราจะกำหนดคุณลักษณะต่างๆที่ต้องการมาจัดเก็บไว้ในตัว template ที่เราสร้างด้วยการใช้ flag ว่า defineTemplateพร้อมกับอ้างอิงจากชื่อของ template ที่ตั้ง เพื่อบอกว่าจะจัดเก็บไว้ที่ไหน ในที่นี้คือ myTemplate

```
button // ระบุประเภทของ element ที่ต้องการสร้าง template ให้กับปุ่ม
-defineTemplate myTemplate
// ให้สร้าง template ขึ้นมาให้กับ element ด้านบน ตั้งชื่อว่า myTemplate
-height 25;
// กำหนดขนาดความสูงของปุ่ม = 25
columnLayout
// ระบุประเภทของ element ที่ต้องการสร้าง template ให้กับ column layout
-defineTemplate myTemplate
// ให้สร้าง template ขึ้นมาให้กับ element ด้านบน ตั้งชื่อว่า myTemplate
-columnAttach "both" 5;
// กำหนดให้วัดค่าขนาดของปุ่มจากทั้งทางซ้ายและขวา โดยมีช่องว่างจากกรอบ = 5
```

เมื่อเราจัดเก็บคุณลักษณะที่ต้องการลงใน template ที่ชื่อ myTemplate แล้ว ขั้นต่อไปเราจะมาดูวิธีการเรียกใช้ template ที่เรา defined กัน ก่อนอื่นให้ลองสร้างหน้าต่างแบบธรรมดาที่ยังไม่มีการใช้ template ขึ้นมาก่อน เพื่อการเปรียบเทียบ

```
window; // สร้างหน้าต่าง frame เปล่า
columnLayout // สร้าง column layout ขึ้นมาใน frame เปล่า
-rowSpacing 2 // กำหนดช่องห่างระหว่างบรรทัด
-columnWidth 250; // กำหนดขนาดความกว้างของ column

button // สร้างปุ่มแบบ default
-label "Button One"; // แสดงข้อความบนปุ่มว่า Button One
button // สร้างปุ่มแบบ default
-label "Button Two"; // แสดงข้อความบนปุ่มว่า Button Two
button // สร้างปุ่มแบบ default
-label "Button Three"; // แสดงข้อความบนปุ่มว่า Button Three
```


951301: Advanced Animation and Rendering Techniques

```
button // สร้างปุ่มแบบ default
-label "Button Four"; // แสดงข้อความบนปุ่มว่า Button Four
button // สร้างปุ่มแบบ default
-label "Button Five"; // แสดงข้อความบนปุ่มว่า Button Five
showWindow; // สั่ง render window
```

เมื่อทดลอง execute คำสั่งจะได้หน้าต่างแบบ default ดังภาพตัวอย่างด้านล่าง จะสังเกตได้ว่าปุ่มทุกปุ่มโดยค่าตั้งต้นจะถูกจัดเรียงแบบชิดซ้าย และจะเหลือเนื้อที่ด้านขวาไว้ไม่สวยงาม

Fig 07-09: เมื่อหน้าต่างถูกสร้างขึ้นมาด้วยค่าตั้งต้น ปุ่มทั้งหมดจะถูกจัดวางที่ตำแหน่งของด้านซ้าย

จากนั้นให้ลองเรียก template ที่สร้างไว้ขึ้นมา สามารถทำได้ง่ายๆโดยการเพิ่มคำสั่ง setUITemplate ลงไปที่บรรทัดที่สอง หลังคำสั่งสร้าง window บรรทัดแรงดังตัวอย่างด้านล่าง

```
window;
setUITemplate -pushTemplate myTemplate;

// สั่งเรียกใช้ template ที่ถูก defined ไว้ชื่อ myTemplate

// ...ที่เหลือเหมือนเดิม
```

จะได้หน้าต่างตามค่า template ที่ตั้งไว้ดังภาพตัวอย่างด้านล่าง ซึ่งมีการตั้ง column attachment แบบทั้งซ้ายและขวา

Fig 07-10: เมื่อเราเรียกใช้ window template ปุ่มทั้งหมดจะถูกจัดเรียงตามค่า template ที่ถูกระบุไว้

951301: Advanced Animation and Rendering Techniques

ตอนนี้นักศึกษาได้เรียนรู้พื้นฐานที่จำเป็นในการออกแบบหน้าต่าง UI กันพอสมควรแล้ว เพื่อความเข้าใจในการประยุกต์ใช้ ในหัวข้อต่อไปเราจะมาทดลองสร้างชุดหน้าต่าง UI แบบง่ายๆ เพื่ออำนวยความสะดวกในการทำงานขั้นตอนต่างๆของ Maya กัน และนำความเข้าใจไปประยุกต์ใช้ออกแบบและสร้างหน้าต่าง UI ของตนเองต่อไป

Workshop 1:

การสร้างหน้าต่างแก้ไขชื่อวัตถุ (rename) แบบทีละวัตถุ (single) และ หลายๆวัตถุพร้อมๆกัน (multiple)

ในการทำงานกับวัตถุแบบกลุ่มทีละหลายๆชิ้นเช่นการสร้าง joints เพื่อควบคุมการเคลื่อนไหวของตัวละคร เราต้องทำการแก้ไขชื่อของ joints แต่ละตัวทีละชิ้นซึ่งใช้เวลาและไม่สะดวกเป็นอย่างมาก ในหัวข้อนี้เราจะสร้างหน้าต่างแบบ prompt dialog เพื่อใช้ในการแก้ไขชื่อวัตถุ สามารถใช้แก้ชื่อวัตถุที่ถูกเลือกทีละวัตถุ หรือจะแก้ชื่อของวัตถุหลายๆวัตถุที่ถูกเลือกพร้อมกันได้ ซึ่งจะช่วยอำนวยความสะดวกในการทำงานเป็นอย่างมาก ที่ต้องใช้ prompt dialog เนื่องจากเราต้องการช่องโต้ตอบแบบที่ผู้ใช้สามารถป้อนข้อความตัวหนังสือ (text) ลงไปได้ และเราจะใช้ for loop ในการแก้ไขชื่อของวัตถุหลายๆชิ้นซ้ำๆกัน โดยจะสร้างตัวเลขตามหลังชื่อแบบอัตโนมัติ

Procedures

ในการทำงานกับชุดคำสั่งขนาดยาวที่จะต้องถูกเรียกใช้เป็นส่วนๆบ่อยๆ เราควรสร้างให้มันเป็น procedure เพื่อความสะดวกในการเรียกใช้และแก้ไขในอนาคต เราจะมาเรียนรู้การทำงานของ procedure ในส่วนนี้กันก่อน เนื่องจากมีความสำคัญเป็นอย่างมากในการสร้างชุดคำสั่งต่างๆในอนาคตต่อไป

Procedures มีคุณลักษณะคล้ายกับ variables คือจะมีขอบเขตการใช้งานของมันอยู่ ซึ่งสามารถแบ่งได้เป็นสองประเภทคือ local และ global procedures การทำงานของ local procedure นั้นจะถูกจำกัดขอบเขตอยู่ที่ตัว MEL scripts ที่เราเขียนมันขึ้นมา ในขณะที่ global procedure นั้นจะสามารถเรียกใช้ได้ในทุกๆ MEL scripts ที่ต้องการเช่นเดียวกับคำสั่งต่างๆที่เราใช้ใน MEL scripts มีให้เราใช้ ประโยชน์ของ procedure คือเราสามารถเรียกใช้ชุดคำสั่งใดๆที่เรา

951301: Advanced Animation and Rendering Techniques

เก็บไว้ใน procedure นั้นๆได้ตลอดเวลา โดยการอ้างอิงจากชื่อของ procedure นั้น ทำให้เราไม่ต้องมาอธิบายให้โปรแกรมเข้าใจถึงสิ่งที่เราต้องการทำซ้ำอีก เราสามารถสร้าง procedure ได้ด้วยคำสั่ง proc ตามด้วยชื่อที่เราต้องการ โดยเราจะต้องใส่คำสั่งทั้งหมดที่จะเก็บไว้ภายในวงเล็บปีกกา { } ดังตัวอย่างด้านล่าง

```
proc testing () // สร้าง procedure ที่ชื่อว่า testing
{
  print "Your Procedure is Working!\n";
  // คำสั่งที่เราต้องการจัดเก็บไว้ใน procedure
}
```

จากตัวอย่างด้านบน เราสั่งสร้าง procedure แบบ local ที่ชื่อว่า testing โดยให้เก็บคำสั่ง print ที่เราต้องการไว้ ถ้าเรา run คำสั่งนี้ใน script editor เราจะยังไม่เห็นผลลัพธ์ใดๆ แต่จะเห็นข้อความที่ส่วน history แสดงว่าโปรแกรมได้จดจำ procedure ที่เราสร้างขึ้นแล้ว จากนั้นให้ทดลองพิมพ์คำว่า testing ตรง command line แล้ว enter ดู จะพบว่าที่ feedback area ปรากฏข้อความ Your Procedure is Working! ขึ้นมา ซึ่งเป็น command ที่เราเก็บไว้ใน testing แปลว่าเราทำได้ถูกต้องแล้ว

ทีนี้ให้ลองปิดโปรแกรมแล้วเปิดขึ้นมาใหม่ แล้วลอง run คำสั่ง testing ขึ้นมาใหม่ จะพบข้อความ error ขึ้นมา เนื่องจาก procedure testing ที่เราสร้างเป็นแบบ local ซึ่งจะมีขอบเขตการใช้งานเฉพาะในที่ๆเราประกาศแล้วเท่านั้น ถ้าต้องการใช้อีกเราต้อง run คำสั่งในการสร้าง procedure ขึ้นมาใหม่ ในกรณีที่เราต้องการให้สามารถเรียกใช้ procedure ได้ตลอดเวลาในทุกๆขอบเขตการทำงาน (รวมถึงตอนเปิดโปรแกรมขึ้นมาใหม่) เราต้องสร้างให้เป็น global procedure ด้วยคำสั่ง global proc ดังตัวอย่างด้านล่าง

```
global proc testing () // สร้าง procedure แบบ global ที่ชื่อว่า testing
{
  print "Your Procedure is Working!\n";
  // คำสั่งที่เราต้องการจัดเก็บไว้ใน procedure
}
```

จากนั้นให้ save scripts ที่เราสร้างให้เป็น MEL format (.mel) โดยการกดปุ่ม save ที่หน้าต่าง script editor แล้วเลือก folder ไปที่ C:\…\maya\2016\scripts (2016 คือ version ของ Maya ที่เราใช้ ถ้านักศึกษาใช้ version อื่นให้ใส่ version ของตนเองลงไปแทนที่) และให้ตั้งชื่อไฟล์เช่นเดียวกับชื่อ procedure ที่เราสร้าง (ในที่นี้คือ testing) จากนั้นให้ทดลองปิดโปรแกรมแล้วเปิดขึ้นมาใหม่ แล้ว run คำสั่ง testing ดูจะพบข้อความ Your Procedure is Working! แสดงว่าโปรแกรมจำ procedure ของเราได้แม้จะเปิดโปรแกรมขึ้นมาใหม่

951301: Advanced Animation and Rendering Techniques

จากตัวอย่างด้านบนเราจะเห็นคุณลักษณะของ procedures แบบ local และ global ที่ต่างกัน และผลลัพธ์ของการสร้างทั้งสองแบบว่ามีการเรียกใช้งานอย่างไร ในขั้นตอนต่อไปเราจะใช้ procedure ในการสร้างหน้าต่าง UI ที่กล่าวไว้ตอนเริ่ม workshop กันต่อ

ในการสร้างหน้าต่าง rename วัตถุนี้ เราจะเริ่มจากการสร้าง procedure แบบ global มาจัดเก็บ scripts ของเราก่อนโดยให้ตั้งชื่อว่า groupRename โดยสิ่งที่ต้องการจัดเก็บให้อยู่ใน procedure นั้นจะต้องอยู่ภายในเครื่องหมายปีกกา { }

```
 global proc groupRename ()
// สร้าง procedure แบบ global ชื่อว่า groupRename
```

จากนั้นเราจะสร้างตัวแปรที่ชื่อว่า $newName เพื่อใช้ในการจัดเก็บชื่อใหม่ที่ผู้ใช้จะป้อนในอนาคต

```
{
 string $newName; // สร้างตัวแปรชื่อว่า $newName
```

เราจะยังไม่ทำอะไรกับตัวแปร $newName ในตอนนี้ แต่เราจะสร้างหน้าต่างแบบ prompt dialog ขึ้นมาเพื่อให้ผู้ใช้สามารถป้อนชื่อใหม่ให้กับวัตถุที่ต้องการ rename เข้าไปได้ โดยให้ค่าต่างๆที่ผู้ใช้โต้ตอบถูกจัดเก็บไว้ในตัวแปรชื่อว่า $promptResult

```
string $promptResult = `promptDialog
 // สร้างหน้าต่าง prompt dialog ให้เก็บค่าใน $promptResult
 -title "Mass Rename Objects" // ตั้งข้อความบน bar ด้านบนให้กับหน้าต่าง
 -message "Enter New Name: " // ข้อความที่แสดงบนหน้าต่าง
 -button "OK" // ให้สร้างปุ่ม OK
 -button "Cancel" // ให้สร้างปุ่ม Cancel
 -defaultButton "OK" // กำหนดว่าถ้าผู้ใช้กด enter ให้มีค่าเท่ากับกด OK
 -cancelButton "Cancel" // กำหนดว่าถ้าผู้ใช้กด Esc ให้มีค่าเท่ากับกด Cancel
 -dismissString "Cancel"` // กำหนดว่าถ้าผู้ใช้กดปิดหน้าต่าง ให้มีค่าเท่ากับกด Cancel
 ;
```

จากนั้นเราจะใช้ if statement ในการเรียกค่าจาก $promptResult ว่าเมื่อผู้ใช้เลือก OK จะให้เก็บข้อความ texts ที่ผู้ใช้ป้อนใน prompt dialog ไว้ในตัวแปรแบบ string ชื่อ $newName

```
 if ($promptResult == "OK") // ถ้าผู้ใช้เลือก OK
  {
     $newName = `promptDialog - query`;
  // ให้เอาชื่อที่ผู้ใช้พิมพ์ไว้ใน prompt dialog ไปใส่ไว้ในตัวแปร $newName ที่เราสร้างไว้ตอนแรก
```


951301: Advanced Animation and Rendering Techniques

ก่อนที่เราจะทำการ rename ให้กับวัตถุได้ เราจะต้องมีรายชื่อของวัตถุทั้งหมดที่ต้องการ rename ก่อน ซึ่งสามารถทำได้โดยการให้โปรแกรมเรียก list รายชื่อของวัตถุทุกชิ้นที่ถูก selected อยู่ เพื่อนำไปจัดเก็บไว้ในตัวแปรแบบ string ชื่อ $selList[] เพื่อใช้ในการแก้ไขต่อไป

```
  string $selList[] = `ls -selection`;
```

เพื่อนำรายชื่อจาก $selList[] มาทำการแก้ไขเป็นชื่อใหม่ที่ผู้ใช้ป้อนเข้าไป เนื่องจากวัตถุที่ถูกเลือกสามารถมีได้มากกว่าหนึ่งชิ้น ทำให้เราต้อง apply คำสั่งต่อไปนี้ซ้ำๆให้กับวัตถุทุกชิ้น เพื่อย่นเวลาเราจึงสามารถนำ for loop มาช่วย โดยสั่งว่าให้เข้าไปค้นหาชื่อของวัตถุใน $selList[] ทีละชิ้น แล้วนำไปแก้ไขชื่อตามค่าที่ได้จาก $newName

```
    int $i = 0; // สร้างตัวแปร $i ให้มีค่า = 0
    for ($each in $selList) // ให้โปรแกรมทำซ้ำกับทุกวัตถุภายใน $selList
    {
    string $newList[] = `ls -selection -long`;
    // สร้างตัวแปรที่ชื่อ $newList[] มาจัดเก็บชื่อวัตถุทุกตัวที่ถูกเลือก
    rename $newList[$i] ($newName + (($i++)+1));
  // ให้ rename ชื่อวัตถุที่ถูกเลือกและจัดเก็บไว้ใน $newList ทั้งหมด โดยให้แทนที่ด้วยชื่อที่ผู้ใช้ป้อนไป
  // ใน $newName และให้ run หมายเลขเลขด้านหลังชื่อตามลำดับเริ่มจาก 1
```

อย่าลืมตรวจสอบเครื่องหมายปีกกาที่เปิดไว้ทั้งหมด ต้องมีตัวปิดด้วยทุกตัว

```
    }
  }
}
```

เมื่อเสร็จสิ้นคำสั่งทั้งหมดแล้วให้นักศึกษา save scripts ที่เราสร้างให้เป็น MEL format (.mel) โดยการกดปุ่ม save ที่หน้าต่าง script editor แล้วเลือก folder ไปที่ C:\...\maya\2016\scripts (2016 คือ version ของ Maya ที่เราใช้) และให้ตั้งชื่อไฟล์เช่นเดียวกับชื่อ procedure ที่เราสร้าง (ในที่นี้คือ groupRename.mel) จากนั้นให้ทดลองปิดโปรแกรมแล้วเปิดขึ้นมาใหม่ แล้ว run คำสั่ง groupRename ที่ command line ดูจะเป็นการเรียกหน้าต่าง rename ที่เราสร้างไว้มาใช้ (เช่นเดียวกับการเรียกใช้คำสั่งทั่วไปอื่นๆใน MEL) และนักศึกษาสามารถเก็บเพียงคำสั่งสั้นๆว่า groupRename ไปไว้บน shelf เพื่อความสะดวกในการเรียกใช้งานต่อไป

951301: Advanced Animation and Rendering Techniques

Workshop 2:

การสร้างหน้าต่างเพื่อทำการ List และ Select เลือกวัตถุที่ต้องการแยกตามประเภทของวัตถุ

จาก workshop แรกนักศึกษาจะพบว่าการใช้ MEL สามารถช่วยอำนวยความสะดวกได้เป็นอย่างมากในการใช้งานลักษณะแบบซ้ำๆให้โปรแกรมจัดการเรื่องยุ่งยากเหล่านี้แทนเรา ในส่วนของ workshop นี้เราจะมาทดลองสร้างเครื่องมืออำนวยความสะดวกในการทำงานลักษณะอื่นกัน

ในการทำงานที่มีวัตถุหลายชนิด จำนวนมากในฉากๆหนึ่ง ผู้ใช้สามารถใช้ outliner ในการเลือกวัตถุแต่ละประเภทได้ แต่เราก็ต้องทำการเปิดปิดการแสดงผลของประเภทวัตถุที่เราต้องการให้แสดง (ข้อต่อ, แสง, วัตถุ เป็นต้น) โดยที่เราไม่สามารถแยกประเภทของวัตถุเป็น sets และแสดงบนหน้าต่างหลายๆอันพร้อมกันได้ ซึ่งทำให้เราต้องเสียเวลาในการเข้าถึงวัตถุเพิ่มขึ้นเป็นอย่างมาก ใน workshop นี้ เราจะมาทดลองสร้างเมนูในการ list วัตถุแต่ละประเภทแยกกัน โดยเราสามารถเลือก select วัตถุนั้นๆได้จากเมนูของเราโดยตรง ในตัวอย่างนี้เราจะมาทดลองเลือก list เฉพาะแหล่งกำเนิดแสงกันก่อน โดยที่นักศึกษาสามารถนำไปประยุกต์ใช้ในการ list วัตถุแบบอื่นๆได้ด้วยตนเอง เพียงแก้ไข scripts จำนวนไม่กี่คำ

เช่นเดียวกับการสร้างหน้าต่าง rename วัตถุ เราเริ่มจากการสร้าง procedure แบบ global ขึ้นมาเพื่อความสะดวกในการเรียกใช้ โดยตั้งชื่อว่า lightwindow และจัดเก็บชุดคำสั่งทั้งหมดไว้ภายในเครื่องหมายปีกกา { }

```
global proc lightWindow()
// สร้าง procedure แบบ global ขึ้นมาให้ชื่อว่า lightWindow
```

ขั้นตอนต่อมาคือการให้โปรแกรมสร้าง list ของสิ่งที่ต้องการด้วยคำสั่ง ls ตามด้วย flag –type เพื่อระบุประเภทของวัตถุที่ต้องการให้ list ในที่นี้เราจะให้ list แหล่งกำเนิดแสงที่สร้างทั้งหมด จึงตามด้วย “light” ถ้านักศึกษาต้องการ list วัตถุแบบอื่น สามารถทำได้โดยการแก้ไข script ตรงส่วนนี้ โดยประเภทของวัตถุที่เราสามารถ flag –type ได้ยกตัวอย่างเช่น “camera”, “mesh”, “joint” และ “locator” เป็นต้น

```
{
    string $lights[] = `ls -type "light"`;
    // สร้างตัวแปรชื่อ $lights[] เพื่อจัดเก็บ list ของแหล่งกำเนิดแสง
```

จากนั้นให้ประกาศตัวแปรเพื่อใช้ในการอ้างอิงหาความยาวของชื่อที่ถูก list เนื่องจากลักษณะการตั้งชื่อของผู้ใช้แต่ละคนไม่เหมือนกัน บางคนอาจใช้ชื่อ default เช่น ambientLight1 หรือชื่อ

951301: Advanced Animation and Rendering Techniques

สั้นๆเพื่อความสะดวกในกรณีที่มีแหล่งกำเนิดแสงไม่มากภายในฉากเช่น light1 ในขณะที่บางคนอาจตั้งชื่อแบบยาวเช่น main_light_from_top_pointing_at_the_main_character เป็นต้น เป็นเรื่องยากที่เราจะสามารถกำหนดขนาดของหน้าต่างให้เหมาะสมได้ เพื่อแก้ไขปัญหานี้เราจะกำหนดตัวแปรเพื่อใช้ในการหาความยาวของชื่อที่ยาวที่สุดที่ผู้ใช้ตั้งไว้ นำคำนวณเป็นขนาดความกว้างของหน้าต่างกัน

```
  string $lightTransforms[];
    int $longestName;
```

เมื่อได้ตัวแปรแล้ว ขั้นตอนต่อไปคือการให้โปรแกรมคำนวณหาจำนวนตัวอักษรของชื่อวัตถุภายใน list ซึ่งจะต้องทำซ้ำๆกับวัตถุทุกตัวที่อยู่ภายใน list เราจึงควรใช้ for loop เข้ามาช่วยทำซ้ำแทนที่เรา

```
  for ($each in $lights)
    {
```

ขั้นตอนต่อไปเราจะทำการค้นหาจำนวนตัวอักษรของชื่อแสงที่ถูก listed ไว้ เพื่อการคำนวณขนาดหน้าต่างที่มีความกว้างเหมาะสมกับความยาวตัวอักษร โดยคำสั่ง size ใช้ในการหาจำนวนตัวอักษรของตัวแปรแบบ string

```
   string $parentList[] = `listRelatives -parent $each`;
   $lightTransforms     [`size     $lightTransforms`]     =
$parentList[0];
```

เราจะใช้ if เพื่อสร้างเงื่อนไขในการกำหนดขนาดหน้าต่างโดยการคำนวณจากจำนวนตัวอักษร เงื่อนไขมีอยู่ว่าถ้าจำนวนตัวอักษรที่เจอใน $parentList มีจำนวนน้อยกว่าจำนวนตัวอักษรใน $longestName ให้เพิ่มค่าของ $longestName ให้เท่ากับจำนวนตัวอักษรที่มากที่สุด เพื่อเป็นการยืนยันว่า $longestName จะแสดงค่าของชื่อที่ยาวที่สุดเสมอ

```
        if (`size $parentList[0]` > $longestName)
        $longestName = `size $parentList[0]`;
    }
```

ขั้นตอนต่อมาคือการเตรียมความพร้อมในการสร้างหน้าต่าง UI ขึ้นมา โดยเราจะเริ่มจากการตรวจสอบดูว่าหน้าต่างที่ชื่อ lightWindowUI เคยถูกสร้างขึ้นมาก่อนแล้วหรือไม่ ถ้ายังไม่เคยถูกสร้างให้กำหนดค่าความกว้างของหน้าต่างจากจำนวนตัวอักษรที่ยาวที่สุดที่เก็บไว้ใน $longestName และค่าความสูงของหน้าต่างจากจำนวนวัตถุทั้งหมดที่ถูก listed ไว้ภายใน $lightTransforms[]

951301: Advanced Animation and Rendering Techniques

```
    if (`window -exists lightWindowUI` != 1)
    {
        int $height = ((`size $lightTransforms` * 21)+35);
        int $width = (($longestName * 6)+1);
```

เราควรตรวจสอบจำนวนของวัตถุที่ถูก listed ว่ามีเท่าไหร่ เพื่อจะได้ออกแบบขนาดของหน้าต่างได้อย่างเหมาะสม ถ้ามีมากกว่า 10 วัตถุ เราจะจำกัดความสูงของหน้าต่างไว้ที่ 245 แล้วส่วนที่เกินขอบเขตการแสดงผลจะถูกจัดเก็บไว้ในรูปแบบ scroll layout ซึ่งถ้าเป็นแบบนั้นเราควรกำหนดขนาดความกว้างของหน้าต่างเพิ่มขึ้นมาจากเดิมด้วย เนื่องจากต้องเผื่อเนื้อที่ให้กับ scroll bar ตามแนวแกนตั้งเพื่อให้สามารถ navigate วัตถุภายใน list

```
        if (`size $lightTransforms` >10)
        {
            $height = 245;
            $width = (($longestName * 6)+35);
        }
```

เมื่อทำตามขั้นตอนเรียบร้อยแล้ว ขั้นตอนต่อไปคือการเริ่มสร้างหน้าต่าง UI ขึ้น โดยในที่นี้เราจะกำหนดให้หน้าต่างที่สร้างเป็นหน้าต่างแบบ toolbox ซึ่งจะมีขนาดเล็กกว่าหน้าต่างปกติ เหมาะกับหน้าที่การใช้งานของเราในโอกาสนี้ที่โจทย์คือการประหยัดเนื้อที่ของหน้าต่างให้มากที่สุด เพื่อผู้ใช้จะได้มีเนื้อที่ในการทำงานให้มากที่สุด โดยมีคุณสมบัติเพิ่มเติมคือ จะอนุญาตให้มีการปรับขนาดหน้าต่างได้ ให้แสดงข้อความว่า Light Selection Window บน title bar และตั้งชื่อหน้าต่างว่า lightWindowUI

```
    window
        -toolbox on
        -resizeToFitChildren on
        -title "Light Selection Window"
        -iconName "Light Selection Window"
        -menuBar false
        -menuBarVisible false
        lightWindowUI
        ;
```

ต่อมาเราจะให้โปรแกรมทำการคำนวณดูว่าถ้ามีวัตถุใน list จำนวนมากกว่า 10 ชิ้น ให้จัด layout เป็นแบบ scroll layout เพื่อจำกัดขนาดของหน้าต่างไม่ให้ใช้เนื้อที่มากเกินไป และปรับขนาดความกว้างของหน้าต่างเพิ่ม เพื่อรองรับเนื้อที่ของ scroll bar ที่จะถูกสร้างมาทางด้านขวามือ

951301: Advanced Animation and Rendering Techniques

```
        if (`size $lightTransforms` >10)
        scrollLayout -horizontalScrollBarThickness 0;

        columnLayout
        -columnAttach "both" 0
        -rowSpacing 1
        -columnWidth (($longestName * 6)+10)
        ;
```

ขั้นตอนเบื้องต้นที่เราทำสำเร็จไปคือการเตรียมความพร้อมต่างๆเช่น การตั้งค่าตัวแปร, การจัดเตรียม list ของวัตถุที่ต้องการ และการออกแบบหน้าต่าง UI ให้มีขนาดเหมาะสม ในขั้นตอนสุดท้ายคือการสั่งให้โปรแกรมทำการ select ไปที่วัตถุนั้นๆเมื่อเราเลือกจากเมนูที่เราสร้าง (เป้าหมายของเราคือการสร้างหน้าต่างเพื่อใช้ในการ select วัตถุ) ในที่นี้สามารถทำได้โดยการแทนที่ชื่อของวัตถุต่างๆภายใน list ให้เป็นปุ่ม (buttons) เพื่อให้ผู้ใช้สามารถกดเลือกได้ และใส่คำสั่ง (flag -command) เพื่อบรรจุคำสั่งที่ต้องการไว้ที่ปุ่มแต่ละปุ่ม ซึ่งก็คือคำสั่งในการเลือกวัตถุตามชื่อปุ่มที่เลือกนั่นเอง ในที่นี้เราจะใช้ for loop เพื่อให้โปรแกรมวน loop ในการสร้างปุ่มตามจำนวนวัตถุภายใน list และใส่คำสั่ง select –r ตามด้วยชื่อวัตถุนั้นๆ ในที่นี้เราสามารถใช้ตัวแปร $each แทนวัตถุนั้นๆได้

```
    for ($each in $lightTransforms)
        button
        -label $each
        -height 20
        -command ("select -r " + $each);
```

เมื่อเสร็จแล้วที่เหลือก็เพียงแค่สั่งให้โปรแกรม render หน้าต่าง UI ที่เราสร้างขึ้นมา โดยให้กำหนดชื่ออ้างอิงของหน้าต่างที่ต้องการ render ไว้ข้างหลังคำสั่ง showWindow ด้วย ในที่นี้หน้าต่างของเราชื่อ lightWindowUI เมื่อเสร็จเรียบร้อยแล้วอย่าลืมปิดปีกกาที่เราเปิดไว้ทั้งหมดด้วย

```
        showWindow lightWindowUI;
    }
}
```

เพื่อความสะดวกในการเรียกใช้งานครั้งต่อไป เราจะทำการ save scripts ที่เราสร้างให้เป็น MEL format (.mel) โดยการกดปุ่ม save ที่หน้าต่าง script editor แล้วเลือก folder ไปที่ C:\…\maya\2016\scripts (2016 คือ version ของ Maya ที่เราใช้) และให้ตั้งชื่อไฟล์เช่นเดียวกับชื่อ procedure ที่เราสร้าง (ในที่นี้คือ lightWindow.mel) จากนั้นให้ทดลองปิดโปรแกรมแล้วเปิดขึ้นมาใหม่ แล้ว run คำสั่ง lightWindow จาก command line ดูจะเป็นการเรียกหน้าต่าง list and

select แสงที่เราสร้างไว้มาใช้ นักศึกษาสามารถเก็บคำสั่งว่า lightWindow ไปไว้บน shelf เพื่อความสะดวกในการใช้งานต่อไป

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