MODELLING AND RENDERING COMPUTER GRAPHIC AND APPLICATION USING PASCAL

การสร้างและแสดงภาพคอมพิวเตอร์กราฟิก และการประยุกต์ใช้อธิบายด้วยภาษาปาสคาล

จัดทำโดย

นาย สงกรานต์ มหิทธิกร เลขประจำตัว 32-1340

นาย สรนาถ ไรภู เลขประจำตัว 32-1360

อาจารย์ที่ปรึกษา

อาจารย์ ภากร พุตะสังกาศ

ปริญญานิพนธ์สำหรับปริญญาวิศวกรรมศาสตร์บัณฑิต

ภาควิชาเทคโนโลยีการวัดคุมทางอุตสาหกรรม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2535

ภาควิชา เทคโนโลยีการวัดคุมทางอุตสาหกรรม

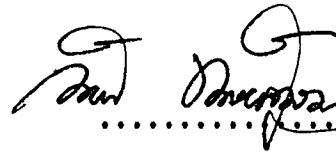
คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหารลาดกระบัง

เรื่อง MODELLING AND RENDERING COMPUTER GRAPHIC AND APPLICATION USING PASCAL

ผู้จัดทำ

นาย สงกรานต์ มหิทธิกร เลขประจำตัว 32-1340

นาย สรนาถ ไรภู เลขประจำตัว 32-1360



..... อาจารย์ที่ปรึกษา

(ภากร หุตะสังกาศ)

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ จะเป็นการศึกษาคอมพิวเตอร์กราฟฟิก ในแขนงที่เรียกว่า geometric modelling เนื้อหาจะประกอบด้วย เรื่องของการกำหนดรูปทรงด้วย ระบบ Co-ordinate ที่ต่างกัน , ความสัมพันธ์ของ Co-ordinate , การแปลง Co-ordinate , โครงสร้างข้อมูล ,ระบบฐานข้อมูล, ทฤษฎีเรขาคณิตที่ใช้ในการกำหนดรูปทรงและแสดงภาพต่างๆ,การนำเมตริกซ์ไปใช้ในการแปลงระบบ Co-ordinate : ทั้งในระบบ 2 มิติ และ 3 มิติ และกล่าวถึง เทคนิคของการใช้เรขาคณิตในการจัดภาพ (Clipping , Blanking , Hacthing , Animationในระบบ 2 มิติ) และกล่าวถึงเทคนิคการแสดงภาพในระบบ 3 มิติ ทั้งในแบบธรรมดาและในแบบที่ช่วยให้ภาพ แลดูสมจริงยิ่งขึ้น (Projection, Hidden line,Hidden surface ,การให้สีและเฉด,เงา) ซึ่งจะอธิบาย และ สร้างเป็น routine ในรูปภาษา pascal มาตรฐานที่สามารถใช้กับ Device หลาย ๆ แบบ นอกจากนี้ยังมีการแสดงถึงการใช้เหล่านี้ไปประยุกต์ใช้ในการแสดงภาพ 3 มิติ เพื่อช่วยในการศึกษารูปทรงในการออกแบบ

ABSTRACT

In this thesis , which aims to study in computer graphic concentrate on field of geometric modelling.

Thesis describes besic concept about object in different Co-ordinate system , data structure , database system , geometric theory , mod -elling and redering objects , transformation of co-ordinate system using matrix : in two-dimensional and three-dimensional space , and Techn ique of geometric for manipulating two-dimensional object (such as clip ping , blanking hatching , animation). And describe in three-dimensi onal display technique for display realistically enough : Projection , Hidden line and hidden surface algorithm , shading , shadow. All of this topic will describe and show by routine in standard pascal : can use in many type of graphic device . And application shaw adventage of these principal.

สารบัญ

วัตถุประสงค์

ในปัจจุบันคอมพิวเตอร์ในแขนง Computer Graphic เริ่มมีการนำมาใช้งานอย่างแพร่หลายมากขึ้นในประเทศไทย แต่ในการศึกษาในแขนงนี้นั้น หนังสืออ้างอิงต่าง ๆ ที่เขียนขึ้นมาเป็นภาษาไทย นับว่ามีอยู่น้อย และที่มีอยู่ก็ยังมีเนื้อหาที่ยังไม่กว้างมากนัก จึงเป็นเหตุให้เกิดความคิดที่จะศึกษาทางด้าน Computer Graphic และ ทำออกมาในลักษณะของหนังสือที่เป็นความรู้เบื้องต้นของ Computer Graphic ในระบบ 2 มิติ และ 3 มิติ มากขึ้นกว่าหนังสือภาษาไทยอื่น และมีความสะดวกในการนำไปใช้กับ Graphic Device ได้มากมาย โดยแก้ไขที่ Primitive Routine เท่านั้น เพื่อที่จะให้ใช้เป็นแนวทางในการศึกษา หรือใช้อ้างอิงเพื่อประกอบการทำ Project และสามารถนำ Routine ต่าง ๆ ที่มีมากกว่าร้อย Routine ไปใช้งานได้ ในการศึกษาและจัดทำนั้น จะใช้หนังสือ High Resolution Computer Graphic using Pascal เป็นหลักในการอ้างอิง สำหรับการเลือกหนังสือเล่มนี้นั้น เนื่องจากเป็นหนังสือ ที่อธิบายหลักการและทฤษฎีต่าง ๆ ได้ละเอียด และง่ายต่อการเข้าใจมากพอที่จะใช้เป็นความรู้เบื้องต้นและนำไปประยุกต์ใช้งานได้อย่างดี . นอกจากนี้ตัวอย่างของ Routine ต่างๆ มีความสมบรูณ์พอที่จะนำไปใช้งานได้ และโครงของมันยังง่ายต่อการเข้าใจใน Algorithm เพื่อความเข้าใจและนำไปปรับปรุงเพื่อใช้กับภาษาอื่น ๆ ได้ จึงคิดว่าน่าจะเหมาะกับจุดประสงค์ที่ตั้งไว้

ขอบเขตของการวิจัย

เนื้อหาของการ Thesis นี้ จะมุ่งเน้นไปที่ Geometric Modelling ของ Computer Graphic จะประกอบไปด้วย หลักการและ Routine ที่สร้างขึ้นจากหลักการนั้น การเขียน Routine ต่าง ๆ นั้น จะมุ่งให้สามารถนำไปประกอบขึ้นเป็น Program ใช้งานได้ (หรือดึงเอาไปใช้งานตามความประสงค์ของผู้ใช้) ในลักษณะที่ Program จะไม่ได้ติดต่อกับ Graphic Device โดยตรง แต่จะมี Routine ที่เรียกใช้เพื่อติดต่อ Device ต่าง ๆ ได้มากชนิด ซึ่ง Primitive Routine จะมีรายละเอียดในโครงสร้างที่แตกต่างกัน ซึ่งผู้ใช้ต้องเขียนเพื่อตอบสนองกับ Device ที่ผู้ใช้ต้องการ แทนที่ Program จะไปเรียกใช้ Device โดยตรง(เมื่อเปลี่ยน device อาจทำให้โปรแกรมใช้งานไม่ได้) เราจึงสามารถสร้างโปรแกรมขึ้นมา และใช้งานได้ในหลายทาง เพียงแต่เราเปลี่ยน Primitive Routine เท่านั้น ก็จะนำไป RUN กับ Device อื่น ๆ ได้ และยังได้แสดงถึง Application ที่นำหลักการเหล่านี้มาประยุกต์ใช้ เพื่อ

ชี้ให้เห็นถึงข้อดีของการจัดทำหนังสือเล่มนี้ขึ้นมา

ในการศึกษาผู้ศึกษาควรมีความเข้าใจในการใช้ภาษา pascal และ ทฤษฎีทางเรขาคณิต พอสมควร และ มีความเข้าใจใน Device ที่จะใช้ เพื่อสร้าง Primitive routine สำหรับ แต่ละ Device ได้ และ ในการสร้างภาพที่ซับซ้อนนั้น ควรจะมีความเข้าใจในเรขาคณิต เป็นอย่างดี และการที่จะเข้าใจหลักการต่าง ๆ ได้ดีนั้นจะต้องหัดเขียน Program ใช้งานให้มากๆ อาจดูเอาจากในเนื้อหาหรือสร้างขึ้นเองก็ได้ จะทำให้มีทักษะดีขึ้น

เนื้อหาของการวิจัย

ใน Computer Graphic Device มีมากมายหลายชนิด และ หลายขนาด เช่น Storage Tubes,Raster Devices,vector refresh displays,flat-bled plotters ฯลฯ ได้มีผู้ที่พยายามเขียน Graphic Software มาตรฐาน หรือ Graphic Packages ขึ้นมาเพื่อติดต่อกับมัน

ในหนังสือเล่มนี้จะอธิบายรายละเอียด ของการกำหนดรูปทรงและ การวาด ให้สี แรงเงา ในระบบ 2 มิติ และ ระบบ 3 มิติ

และเพื่อให้หลักการในหนังสือเล่มนี้ สามารถใช้ใน Graphic Device ได้หลายชนิด เราจึงสร้างโครงสร้างของ Primitive Routine ขึ้นมา เพื่อใช้ติดต่อกับ Graphic Device โดยให้มีoutlineที่เหมือนกันในทุก ๆ Device แต่มีรายละเอียดหรือการเขียน Routine ที่แตกต่างกัน ตาม Graphic Device แต่ละชนิด ที่ผู้ใช้ต้องการ

โปรแกรมในหนังสือเล่มนี้มีจุดประสงค์ ให้สามารถใช้ได้ใน Graphic Device หลายชนิด และ Pascal หลายๆ VERSION จากเหตุผลนี้ โครงสร้างของมันอาจไม่ใช้โครงสร้างที่ดีนัก โปรแกรมที่สร้างในนี้จะสอดคล้อง กับ memory ที่แบ่งเป็น BLOCK block ละ 64 K stack และ heap ก็เป็นลักษณะเดียวกัน ด้วยเหตุผลนี้โปรแกรมนี้ไม่เพียงแต่จะ run ได้ใน main frame เท่านั้น ยังสามารถ RUN ได้ใน micro computer ด้วย

เมื่อท่านอ่านหนังสือเล่มนี้ ท่านจะพบว่า โปรแกรมหลาย ๆ โปรแกรมจะต้องนำมาใช้ร่วมกับโปรแกรมที่เขียนไว้ขึ้นก่อนหน้า และ การกำหนดค่าตัวแปรต่าง ๆ เพื่อ run โปรแกรมนั้นจะมีขนาดใหญ่ขึ้นเรื่อยๆ เนื่องจากเราเขียนโปรแกรมใน *listing* ต่างๆ อาจต้องใช้ *listing* เหล่านี้มาร่วมประกอบกัน เพื่อเป็นโปรแกรม ที่ สมบูรณ์ ดังจะเขียนกำกับไว้ต่อไป

เรากำหนดว่า ส่วนแสดงผล ของ Graphic Device นั้น เรียกว่า viewport(frame) ซึ่งประกอบด้วย rectangular array ของจุด(pixel) หรือ matrix nxpix * nypic โดย nxpix คือ จำนวน pixel ในแนวนอน nypix คือ จำนวน pixel ในแนวแกนตั้ง

แต่ละ pixel ใน viewport นั้น จะถูกอ้างถึงด้วย pixel Co-ordinate ที่เป็นคู่ลำดับของ integer ด้วยตัวแปร RECORD TYPE vixelvector เขียนแทนด้วย (pixel.x, pixel.y) ซึ่งบอกตำแหน่งของ pixel ที่สัมพันธ์กับ มุมล่างด้านซ้าย (หรือ มุมบนด้านซ้าย)ของ viewport ซึ่งจะมีค่า (pixel.x,pixel.y) เป็น (0,0)

ในโทรทัศน์สี และ RGB color monitor จะทำงานด้วยหลักการ ผสมแม่สี 3 สี คือ แดง , เขียว , น้ำเงิน เข้าด้วยกัน จะให้ค่าสีต่าง ๆ ออกมา ในแต่ละ pixel นั้นจะประกอบไปด้วยจุดเล็กๆ 3 จุดของแต่ละสี สีต่าง ๆ ที่เกิดขึ้นนั้นจะเป็นไปตามความสว่างของจุดทั้ง 3 นี้ (รายละเอียดในบทที่ 15) ด้วยเหตุผลนี้ Graphic Device ส่วนมาก จะกำหนดสีในรูปขององค์ประกอบของ 3 สีนี้ เรากำหนดว่า Device ของเรามี colour look-up table ซึ่งกำหนดสีด้วย numcol colours จะเป็นค่า integer ระหว่าง 0 ถึง numcol-1 ในแต่ละค่า integer นี้เราจะเรียกว่า logical Colour ส่วนสีที่อยู่ใน look-up table คือ Actual Colour โดยผู้ใช้กำหนดขึ้นเองก็ได้ สำหรับ pixel ต่าง ๆ นั้นจะถูกแสดงสีด้วย Actual Colour ที่กำหนดด้วย Logical Colour

เมื่อเราสมมุติให้ cursor นั้น เลื่อนตำแหน่งไปใน viewport Pixel Co-ordinate ที่ตำแหน่งของ Cursor เราจะเรียกว่า Current position ภาพต่างๆจะถูกวาดโดยการเคลื่อนที่ของ Cursor ไปบน viewportตามตำแหน่งที่ถูกกำหนด และ reset ค่า bit-map ที่ current position ให้เป็นค่าของ logical colour

Primitive routine

prepit;

เป็น Routine ที่ใช้ในการเตรียม viewport เพื่อใช้ในการแสดงภาพ

finish;

Routineหลังจากภาพถูกวาดเสร็จแล้ว ในบาง device ต้องการคำสั่งนี้เพื่อจบ frame

setcol(col);

เป็นการเปลี่ยน current colour ด้วย logical colour; 0 <= col <= numcol

erase;

เป็นการลบ pixel ใน viewport ด้วย current colour

(ถ้าเป็นใน micro film คำสั่งนั้น คือ การเลื่อนไปยังเฟรมต่อไป)

setpix(pixel);

เป็นการให้สีแก่ current pixel ที่กำหนดด้วย pixel TYPE pixelvector ด้วย

current colour

movepix(pixel);

เป็นการเลื่อน graphic cursor ไปยัง current position ที่กำหนดโดย pixel โดยไม่เปลี่ยน สี

linepix(pixel);

เป็นการวาดเส้นด้วย current colour จาก Current position ไปถึงตำแหน่งใหม่ที่กำหนดด้วย pixel และ ก็จะกลายเป็น current position ใหม่

polypix (n,poly);

ใช้ในการวาดรูปหลายเหลี่ยม n เหลี่ยมด้วย n pixel vector ที่กำหนดโดย poly[i]

i = 1,..,n

rgblog(i,r,g,b);

เป็นการกำหนด Actual colour ใน colour look-up table

ซึ่งจะมีวิธีการกำหนดได้หลายวิธี แต่ในที่นี้table จะถูกกำหนดด้วย logical colour i ด้วยองค์ประกอบของสี แดง(r) เขียว(g) และน้ำเงิน(b)

ความสามารถในการกำหนด Actual Colour ขึ้นกับ device ในแต่ละชนิด ผู้อ่านควรจะศึกษาวิธีการกำหนดสี และ ข้อจำกัดในแต่ละ Device ที่ท่านใช้ เมื่อกำหนด routine นี้

ในการเขียน Primitive ผู้ใช้อาจเขียน routine ของผู้ใช้ขึ้นมาเอง เพิ่มเติมได้เพื่อใช้กับ program ที่ท่านเขียนขึ้น เช่น ใน Raster Device บางแบบ จะมีการวาดเส้นด้วยเส้นแบบต่างๆ ในการนี้ นั้นไม่สามารถวาดเส้นนั้นด้วยวิธีปกติ แต่ละ pixel บนเส้น นั้นจะถูกให้สีด้วยวิธี bit-wise boolean binary opertion(เช่น exclusive OR) ในบทที่ 5ของหนังสือจะแสดงถึง Primitive Routine line styles และคำอธิบาย หลักการเหล่านี้

ในหนังสือเล่มนี้นั้นเราไม่ได้ศึกษาถึง computer graphic ในทุกๆแขนง เราจะมุ่งศึกษาในด้านการสร้างภาพด้วยวิธีทางเรขาคณิต เท่านั้น (Geometric modelling)

ในการกำหนดจุดต่างๆ เพื่อการวาดภาพด้วย pixel vector โดยตรงนั้น มีข้อจำกัดมาก

มาย เราสามารถแก้ข้อจำกัดนั้นด้วยการใช้ Real Co-ordinate ช่วยในการกำหนด แต่ใน primitive ที่ติดต่อกับ Graphic device นั้น เรากำหนดจุดต่าง ๆ ด้วย pixel vector เพราะฉะนั้นเราจำเป็นต้องหา routine ที่มีความสัมพันธ์กัน เพื่อแปลง real Co-ordinate ที่กำหนดนี้เป็น pixel vector

เราอาจมองภาพของ Two-Dimension space เป็นระนาบของกระดาษที่เป็น infinity ในทุกทิศทาง ในการกำหนดตำแหน่งของจุดต่าง ๆ ในระนาบนี้ เราใช้อ้างด้วย ระบบพิกัด cartersian บนระนาบ เริ่มต้นเราเลือกจุดใดจุดหนึ่งบน space ขึ้นมา และกำหนดให้เป็น Co-ordinate Origin หรือ จุด Origin แล้วลากเส้นแนวนอนโดยผ่านจุด Origin เรียกว่า แกน x โดยค่าทางขวาของจุด Origin จะเป็นบวก ทางซ้ายจะเป็นลบ และลากเส้นตรงในแนวตั้งฉาก(แนวตั้ง)กับแกน x ผ่านจุด Origin เราจะเรียกว่า แกน y โดยเหนือจาก Origin จะเป็นบวก ใต้ Origin ลงไปจะเป็นลบ และจากนั้นก็กำหนด scale ลงไป โดยไม่จำเป็นว่า scale ในแนวแกน x กับแกน y จะต้องเท่ากัน

จากนี้เราสามารถกำหนด ตำแหน่งของจุด(p)ต่างๆใน space นี้ได้ โดยอาศัยคู่ลำดับ (x,y) x คือ ระยะทางตามแนวแกน x ที่วัดจาก Origin ไปยังเส้นตรง ที่ลากจากจุด p มาตั้งฉาก และตัดกับแกน x (projection) และ y คือ ระยะทางจาก Originไปยังจุดที่เส้นตรงที่ลากผ่านจากจุด p มาตั้งฉากและตัดกับแกน y (projection)

เป็นที่ตระหนักอย่างหนึ่งว่า ค่าคู่ลำดับที่ใช้แสดงจุดใดๆบน space นั้น ค่าของมันจะขึ้นกับ Co-ordinate system ที่เราเลือกด้วย ในการศึกษาทาง Computer Graphic Algorithm นั้น เราไม่ได้ใช้ Co-ordinate system เดียวในการแสดง รูปทรงต่างๆใน space ตัวอย่างเช่น ถ้าเรามี Co-ordinate system อยู่ 2 ระบบ โดยแกนอ้างอิงขนานกัน (x,y) แต่มีจุด Origin ต่างกัน เป็นระยะจากแกน x = 1 แกน y = 2 เมื่อนั้น จุด Origin ของระบบหนึ่ง (0,0) ในอีกระบบหนึ่งจะเป็น (1,2) เพราะฉะนั้นจุด P ใดๆ ใน space ก็จะมี ค่า vector Co-ordinate (คู่ลำดับ) ที่แตกต่างกันใน 2 ระบบนี้

ถึงแม้ว่าใน Computer Graphic นั้นจะมีหลาย Co-ordinate system ก็ตาม แต่ทุกๆ System นั้น ต้องมีความสัมพันธ์กับ ABSOLUTE Co-ordinate system (หรือ เรียกว่า World Co-ordinate System) ซึ่งก็คือระบบพิกัด Cartesian กล่าวคือ

เรากำหนด พื้นที่สี่เหลี่ยมผืนผ้า rectangular area หรือ window ที่เท่ากับ horiz * vert หน่วย ซึ่งมี Origin และ แกนสัมพันธ์กับ ASOLUTE system ขึ้นมา ซึ่ง window นี้ก็คือ viewport เพื่อใช้ในการวาดภาพลงบน graphic device เราสามารถย้าย window นี้ในบริเวณต่างบน Two-Dimension space ได้ ใน window นั้นเรากำหนด Co-ordinate

system ขึ้นมา เรียกว่า window system ซึ่งมีจุด Origin อยู่ตรงกลาง window และแกน x กับแกน y จะขนานกับขอบของ window และ scale ในแนวแกน x และ แกน y จะเท่ากัน แต่โดยที่เรากำหนด จุด, เส้นตรง,หรือรูปหลายเหลี่ยม ด้วย ABSOLUTE SYSTEM เรามีจึงมีความจำเป็นต้องรู้ ความสัมพันธ์ ระหว่าง ABSOLUTE และ WINDOW SYSTEM ว่าเป็นอย่างไร เมื่อเรารู้ความสัมพันธ์แล้ว เราสามารถแสดงคู่ลำดับของจุดใน ABSOLUTE Co-ordinate ด้วย ABSOLUTE co-ordinate แล้ว เราสามารถแสดงจุดเหล่านั้นด้วย pixel ใน viewport ได้

ส่วนในบทที่สองของหนังสือจะเป็นการแนะนำเกี่ยวกับโครงสร้างของข้อมูล เพื่อที่จะเข้าใจโครงสร้างของ Algorithm ที่ซับซ้อนได้ดีขึ้น

ในบทที่สามจะกล่าวถึงเรขาคณิตในระบบ Co-ordinate 2 มิติ การกำหนด curve ด้วย Analytic form กับ Parametric form ซึ่งทฤษฎีทางคณิตศาสตร์ นั้นมีความสำคัญต่องานด้านคอมพิวเตอร์มาก โดยเฉพาะทางด้าน Computer Graphic บทนี้จะกล่างถึง ทฤษฎีทางคณิตศาสตร์เบื่องต้นที่ใช้ในบทต่อๆไป

ในบทที่สี่ จะกล่างถึง matrix กับการแปลง Co-ordinate 2 มิติ โดยใช้ matrix, ความสัมพันธ์ของ SETUP กับ ACTUAL position และ OBSERVER system

เราสามารถย้าย Window ไปในบริเวณต่างๆของ ABSOLUTE System ได้ จากเหตุผลนี้ Co-ordinate ของ Window กับ Absolute จะไม่เหมือนกัน Co-odinate Origin ของ Window จะสามารถย้ายไปในบริเวณต่างๆ ของ ABSOLUTE System ได้

เมื่อ Co-odinate system เปลี่ยนไปนั้นเราจะต้องคอยศึกษาว่าจะมีผลต่อคู่ลำดับที่แทนจุด เส้นตรง Curve ต่างๆ ใน Co-ordinate system อย่างไร

ใน Two-dimensional และ Three-dimensional space นั้น เราจะใช้หลักการเบื้องต้น 3 แบบ ในการเปลี่ยน Co-ordinate system คือ การย้ายจุด Origin การเปลี่ยน Scale และการหมุนแกน ในการแปลงแบบอื่นนั้น เราสามารถนำการแปลงทั้ง 3 มา ประกอบกัน เพื่อสร้างการแปลงนั้นขึ้นมาได้ หลักการแปลงต่างๆในบทนี้จะเป็นการแปลงในระบบ 3 มิติ ส่วนในระบบ 3 มิติ จะกล่าวถึงในบทที่ 7 ต่อไป

ในการแปลงต่าง ๆ นั้น ส่วนมากเรามักจะแปลงที่ละหลาย ๆ จุด ซึ่งวิธีที่ง่ายที่สุด ในการแปลงก็คือ สร้างเมตริกซ์ ของการแปลงขึ้นมา ซึ่งจะมีลักษณะเป็น Sguare matrix 3x3

และใน Three-Dimensional space เมตริกซ์ของการแปลงจะเขียนเป็น matrix 4x4 และเราจะเขียนคู่ลำดับของจุดต่าง ๆ ด้วย Column Vector เราจะนำหลักการเหล่านี้ไปเขียน Routine ขึ้นมา เมื่อต้องการแปลงจุดใด ๆ ใน space นั้น เรานำคู่ลำดับของจุดนั้นที่เขียนในรูป Column vector ไปคูณกับ Transformation matrix เราก็จะได้คู่ลำดับใหม่ที่สัมพันธ์กับ Co-ordinate System ที่เปลี่ยนไป

เราสามารถแสดงคู่ลำดับต่างๆ ของรูปทรงต่างๆ (อาจแสดงมุมของมัน) ได้โดยใช้ Co-ordinate System ที่ง่ายต่อการกำหนด โดยเราเรียกว่า SETUP position

แต่ละมุมของรูปนั้นจะถูกแปลงจาก SETUP position ไปยังตำแหน่งที่ต้องการใน Space โดยใช้ matrix ของการแปลงคูณกับ Vector Column ของคู่ลำดับนั้นๆ ตำแหน่งใหม่ที่ได้นี้เรียกว่า ACTUAL position

Matrix ของการแปลงที่แสดงถึงความสัมพันธ์ของ SETUP กับ ACTUAL position เราเรียกว่า matrix p ซึ่งอาจเป็นเมตริกซ์ของการเปลี่ยน Orgin,การเปลี่ยน Scale, การหมุนแกน หรือ เป็น Combination ของ matrix ดังกล่าวก็ได้

ข้อสังเกตุ คือ Co-ordinate ของรูปทรงต่างๆใน SETUP และ ACTUAL position นั้นจะต้องอยู่ ABSOLUTE system (บทที่ 1)

ภาพที่จะถูกวาดลงใน viewport นั้น จะเป็นภาพที่ถูกกำหนดจากจุดที่สังเกตุ ที่ซึ่งสัมพันธ์กับ ABSOLUTE system Concept คือ เหมือนกับคนยืนดูภาพที่อยู่บนกำแพง โดยกำแพงนี้จะแทน Cartesian Plane ที่ใช้วาดภาพ 2 มิตินี้ จุดของตาที่สังเกตุนั้นเรากำหนดให้เป็น(eye.x,eye.y) ใน ABSOLUTE system และ ตาจะเอียงไปในทิศทางทวนเข็มนาฬิกา จะถูกแทนด้วยมุม α จาก Concept ของจุดสังเกตนี้ ภาพที่เห็นจากตาของผู้สังเกต ก็คือภาพที่ปรากฏบน Viewport นั้นเอง

ในกรณีเรากำหนด Co-ordinate System ขึ้นมาคือ OBSERVER system ซึ่ง Co-ordinate ของจุดต่างที่แปลงมาอยู่ ในระบบของการสังเกต (OBSERVER system) นี้ จะเรียกว่า OBSERVER Co-ordinate ใน OBSERVER system นั้น เราจะให้เมตริกซ์ของการแปลงที่สร้างจากหลักการคือ เราย้ายจุด Origin ใน ABSOLUTE system มายังจุด (eye.x, eye.y) จากนั้นก็จะหมุนแกนไปในทิศทางทวนเข็มนาฬิกาด้วยมุม α เมตริกซ์ ของการแปลงนี้จะเรียกว่า Matrix Q ซึ่งโปรแกรมของการหา matrix Q นี้จะอยู่ใน Routine find Q และ look2 ใน *listing 4.2*

และ routine ของการแปลงจาก ABSOLUTE system ไปยัง OBSERVER system คือ Routine observe

ในการแสดงภาพบน graphic viewport ภาพที่ถูกแสดงบน viewport นั้นคือภาพที่ได้จากการสังเกตุ ในบทที่ 1 เรากำหนดให้ viewport คือ rectangular window ใน two-dimensional space เราสามารถให้ window OBSERVER system ได้ ดังนั้น คู่ลำดับต่าง ๆ ของภาพใน window กับ OBSERVER นั้นมีค่าเดียวกัน และจาก window ไปยัง viewport นั้น เราให้หลักการดังแสดงในบทที่ 1

ภาพของระบบ 2 มิติ จะถูกวาดลงบน viewport ด้วย Rountine drawit

ในบทที่ห้า จะกล่าวถึง เทคนิคต่างในการจัดภาพในระบบ 2 มิติ เช่น Clipping กับการ blanking , เทคนิคในการระบายภาพด้วยวิธี hacthing, การ intersect กัน ของ polygon และ Animation

วิธีเบื้องต้นที่ได้แนะนำมาก่อนแล้ว ทำให้เราสามารถสร้างและวาดภาพ 2 มิติ ง่าย ๆ ที่ประกอบด้วย กลุ่มของมุม,ด้านของรูปหลายเหลี่ยม ในบทนี้เราจะพิจารณาเทคนิคต่างๆ ซึ่งอาจจะนำไปใช้ในการสร้างภาพ 2 มิติที่มีความซับซ้อน ซึ่งสามารถปรับปรุงไปใช้ในระบบ 3 มิติได้ routine เหล่านี้จะต้องนำมาทำเป็นโปรแกรมที่สมบรูณ์สำหรับการใช้งาน

ในบทที่ 4 การวาดวัตถุทำได้โดย เปรียบ viewport เหมือน rectangular window ที่จุดศูนย์กลางอยู่บน origin ของ OBSERVER system ใน OBSERVER system ประกอบด้วยแกน 2 ที่แทนระนาบ cartesian ที่มีขนาดเป็น infinity แต่สำหรับ viewportไม่ใช่ แน่นอนเราไม่ประสงค์ที่จะแสดงถึงระนาบ cartesian ทั้งหมด เราต้องการเพียงส่วนที่ประกอบด้วยมุมเท่านั้น จะไม่มีปัญหาเกิดขึ้นถ้า window ที่ถูกกำหนดด้วย viewport นั้นสามารถบรรจุมุมต่างๆได้ทั้งหมด(รวมทั้งเส้นและด้านต่างๆ) ถ้ามีส่วนของวัตถุปรากฏอยู่ภายนอก window ดังนั้นเราต้องสามารถที่แสดงส่วนต่างๆนี้ เพื่อจะได้ถูกตัดออกไป ในที่นี้เราจะมุ่งความสนใจในการจัดการเกี่ยวกับเส้นต่างๆ ปัญหาเกี่ยวกับรูปหลายเหลี่ยม ในส่วนที่อยู่นอก window จะพิจารณาในตอนท้ายของบท สำหรับการพล็อตจุดนั้นไม่มีปัญหา ถ้าจุดนั้นอยู่ภายใน window ก็สามารถที่จะพล็อตได้ แต่ถ้าอยู่ภายนอก window ก็ไม่สามารถที่จะพล็อตได้(Clipping)

ในบทที่หก จะกล่าวถึง เรขาคณิตระบบ Co-ordinate 2 มิติ , Analytic form , การเรียงตัวของ Convex polygon ในระบบ 2 มิติ และ 3 มิติ เช่น การกำหนดเส้นตรง มุมระหว่าง 2 Direction vector การกำหนดระนาบ จุดตัดของเส้นตรงกับระนาบ ระยะทางของจุดวัดจากระนาบ การสะท้อนของจุดในระนาบ จุดตัดระหว่างเส้นตรงสองเส้น Vector Product ระยะทางที่สั้นที่สุดระหว่างเส้นตรงสองเส้น ระนาบของจุด 3 จุดที่ไม่ได้อยู่บนเส้น

เดียวกัน จุดตัดของระนาบ 3 ระนาบ เส้นตรงที่แสดงการตัดกันของระนาบ 2 ระนาบ Analytic Representation การเรียงกันของ Convex polygon ใน three-dimensional space การเรียงกันของ Convex polygon ใน Two-dimensional space

ก่อนที่เราจะศึกษาถึงการแสดงภาพกราฟฟิกใน three-dimensinal space(3 มิติ) นั้น จำเป็นต้องทราบเกี่ยวกับระบบของเรขาคณิต 3 มิติ ในระบบ Cartesian และวิธีการที่จะคำนวณเกี่ยวกับเรื่องต่างใน Three-dimension space (3 มิติ) ได้เข้าใจอย่างง่าย ๆ

ในระบบ 2 มิติ จะใช้ Co-ordinate Origin ส่วนในระบบ 3 มิติ จะประกอบไปด้วยเส้นตั้งฉากกัน 3 เส้น ถือเป็นเส้นหลักและมีความยาวเป็น infinity เส้นเหล่านี้ให้เป็นแกน x ,y และ z จะมีค่าเป็นบวกและลบอย่างละครึ่ง โดยจุดกึ่งกลางให้เป็นจุดกำเนิด ระยะทางจากจุดกำเนิดถึงจุดใดๆ บนแต่ละแกนจะมีค่าเป็นบวก และ ในทิศทางที่ตรงข้ามกันจะให้เป็นลบ ในแกน x และ y จะมีลักษณะคล้ายคลึงกับระบบ Two-Dimensional space โดยแกน x จะวางอยู่ในแนวนอน ส่วนแกนจะ y อยู่ในแนวตั้ง ค่าที่เป็นบวกของแกน x จะอยู่ทางขวามือของจุดกำเนิด ในลักษณะเดียวกันค่าที่เป็นบวกของแกน y จะอยู่เหนือจุดกำเนิดขึ้นไปข้างบน ส่วนแกน z จะตั้งฉากกับแกนทั้งสอง ค่าบวกของ z จะอยู่ด้านที่พุ่งเข้าไปยังระนาบแกน x,y (สำหรับมือซ้าย) และพุ่งออกจากระนาบx,y(สำหรับมือขวา) ซึ่งเราจะเห็นลักษณะของแกนต่างๆโดย ทดลองนำมือของเราเองทั้งซ้ายและขวามาลอง กำหนดให้นิ้วต่างๆ แทนแกนต่างๆ ซึ่งตั้งฉากกัน ตามรูป 6.1 โดยจะใช้นิ้วโป้งชี้ไปทางแนวแกน x ส่วนนิ้วชี้ชี้ไปตามแนวแกน y และนิ้วนางชี้ไปทางแกน z ซึ่งพอที่จะทำให้เราเกิด imagine ต่างของแกนต่างพอเข้าใจ

ในบทที่เจ็ด จะกล่าวถึง เรื่องของ matrix การแปลง Co-ordinate ความสัมพันธ์ของ SETUP กับ ACTUAL position และ OBSERVER system ซึ่งภายในเนื้องหาจะประกอบด้วย การย้ายจุด Origin,การเปลี่ยน Scale,การหมุนแกน,Combining Transformation,การหมุนของแกนด้วยมุม ϕ รอบแกนใดๆ ($b_u + \mu d$),SETUP and ACTUAL position,โครงสร้างของ scene construction

หลักการแปลงต่าง ๆ ใน Two-Dimensional space ที่ได้อธิบายไว้ในบทที่ 4 สามารถจะนำมาใช้ใน Three-Dimension space ได้ และเป็นพื้นฐานที่สำคัญในการทำการวาดภาพในระบบ 3 มิติ โดยใช้ projection ลงบนระบบ 2 มิติ (device ต่าง ๆ นั้นอยู่ในระบบ 2 มิติ) หลักการแปลงต่าง ๆ เช่น การย้ายจุด Origin , การแปลง Scale และการหมุนแกนในระบบ 3 มิติ นั้นใช้หลักการเดียวกับในระบบ 2 มิติ แต่ matrix ที่ใช้ในการแปลงใน Three-Dimension space นั้น จะใช้ Matrix (4x4) และจุดต่าง ๆ ใน space จะถูก

แทนด้วย Column Vector

ในเรื่องของ SETUP and ACTUAL position นั้น เราสามารถแสดงคู่ลำดับต่างๆ ของรูปทรงต่างๆ (อาจแสดงด้วยมุมของมัน) ได้โดยใช้ Co-ordinate System ที่ง่ายต่อการกำหนด โดยเราเรียกว่า SETUP position

แต่ละมุมของรูปนั้นจะถูกแปลงจาก SETUP position ไปยังตำแหน่งที่ต้องการใน Space โดยใช้ matrix ของการแปลงคูณกับ Vector Column ของคู่ลำดับนั้นๆ ตำแหน่งใหม่ที่ได้นี้เรียกว่า ACTUAL position

Matrix ของการแปลงที่แสดงถึงความสัมพันธ์ของ SETUP กับ ACTUAL position เราเรียกว่า matrix p ซึ่งอาจเป็นเมตริกซ์ของการเปลี่ยน Orgin,การเปลี่ยน Scale, การหมุนแกน หรือ เป็น Combination ของ matrix ดังกล่าวก็ได้

ข้อสังเกตุ คือ Co-ordinate ของรูปทรงต่างๆใน SETUP และ ACTUAL position นั้นจะกำหนดใน ABSOLUTE system (บทที่ 1)

ในบทที่แปด เราจะพิจารณาถึงปัญหาของการแสดงภาพ 3 มิติบน viewport ในบทที่ 7 นั้นเราได้กล่าวถึงการกำหนดรูปทรง 3มิติใน ACTUALposition ในบทนี้จะกล่าวถึงจุดสังเกต (observer) และภาพที่เกิดจากการมองจากจุดสังเกตว่าจะเป็นอย่างไร และจะแสดงบน view port ได้อย่างไร ซึ่งจะประกอบไปด้วยเรื่อง Projection ของ the view plane , the window and the viewport , the orthgraphic projection , Drawing a scene

เราจะกำหนดว่า ข้อมูลของภาพที่จะแสดงนั้นจะกำหนดในรูปของ Co-ordinate ใน ACTUAL position ซึ่งสัมพันธ์กับ ABSOLUTE co-ordinate system

ในการแสดงภาพบน viewport นั้นเปรียบเสมือนตาเรามองไปยังวัสดุ ซึ่งเมื่อเรายืนในตำแหน่งที่ต่างกันและเอียงศรีษะด้วยมุมที่ต่างกัน เราก็จะได้ภาพที่มีลักษณะแตกต่างกันออกไป ซึ่งเราสามารถเรียกว่า OBSERVE system แต่เรากำหนดวัตถุต่างๆกันใน ABSOLUTE system ฉะนั้นเราจึงต้องหาความสัมพันธ์ของ 2 ระบบ เพื่อสร้าง matrix ของการแปลง Co-ordinate จาก ABSOLUTE มายัง OBSERVER Co-ordinate(matrix Q) ซึ่งเรากำหนดให้ จุดสังเกตนั้นเป็น origin ของ OBSERVER ซึ่งมีคู่ลำดับที่แทนด้วยตัวแปร eye(vector3) ใน ABSOLUTE system ซึ่งจะมี direction vector จากจุดสังเกตไปยัง origin ของ ABSOLUTE ด้วยตัวแปร direct คือ direct.x = -eye.x , direct.y = -eye.y , direct.z = -eye.z มีค่าเป็นลบ เนื่องจากเราบอกจากจุด eye ไปยัง origin ตามทฤษฎีเรขาคณิต ในการแปลง Co-ordinate ของจุดต่างใน ABSOLUTE system มาเป็น OBSERVER system เราจะใช้หลัก

การของการหมุนรอบแกน b+μd ใด ในบทที่ 7 มาใช้ โดย b = (eye.x,eye.y,eye.z) และ d = (-direct.x,-direct.y,-direct.z)

ในบทที่เก้า ซึ่งในบทที่แปดเราได้อธิบายวิธีเบื้องต้นต่างๆ ใน rotuine ใช้ในการสร้างข้อมูลข ติ ให้กับ Constrution routine ซึ่งบางที่ก็นำข้อมูลมาจาก file จากนั้นข้อมูลของจุดต่างๆ จะถูกแปลงไปอยู่ใน OBSERVED position เพื่อให้ routine drawit เรียกไปใช้เพื่อแสดงภาพ ในบทต่อๆไปก็จะกล่าวถึง drawit แบบต่างๆ Position ชนิดต่างๆ แบบ line drawinh หรือ colour display และอื่นๆ

สิ่งที่สำคัญก่อนที่เราจะศึกษาในบทต่อไป เราต้องต้องคำนิงก็คือ การสร้างข้อมูล ซึ่งจะกล่าวในบทนี้ เนื้อหาประกอบด้วย การใช้ file input,การใช้ routine datain,การกำหนดตำแหน่งแบบสัมพัทธ์ของวัตถุ,วิธี Extrusion,วิธี Body of Revolution,Three dimension Animation

การใช้ Routine Datain

โดยมากข้อมูลที่ใช้ในการแสดงภาพมักถูกเรียกใช้หลายครั้งหลายคราว ดังนั้นเราสามารถสร้างข้อมูลขึ้นเพียงครั้งเดียวและเก็บข้อมูลไว้เพื่อเรียกใช้ในครั้งต่อไป แทนที่จะสร้างข้อมูลขึ้นใหม่ทุกครั้งที่ใช้งาน โดยเราสามารถกำหนดข้อมูลได้จากคีย์บอร์ด และใช้ routine dataout เพื่อเก็บข้อมูลไว้ใน file และเรียก file นั้นออกมาใช้ด้วย routine datain โดยปกติข้อมูลที่เก็บไว้ใน file มักจะเป็นข้อมูลที่กำหนดด้วย SETUP position เมื่อเราต้องการแสดงภาพจากข้อมูลเหล่านี้ เราจะใช้ routine scene เพื่อเอาข้อมูลนี้ไปแสดง ซึ่งเราสามารถใช้ matrix ในการแปลงข้อมูลให้อยู่ใน ACTUAL position ได้ตามที่เรากำหนด

การกำหนดตำแหน่งแบบสัมพัทธ์ของวัตถุ

เราสามารถกำหนดตำแหน่งของวัตถุใดด้วยการกำหนดข้อมูลแต่ละส่วนแยกจากกันได้ และนำไปประกอบกัน วิธีนี้มักมีการนำไปใช้บ่อยครั้งเพื่อความสะดวกในการกำหนดข้อมูล

วิธี Extrusion

เราสามารถนำกำหนดภาพขึ้นในระบบ 2 มิติ และนำไปแสดงในระบบ 3 มิติได้ ซึ่งมีอยู่ 2 วิธี วิธีแรกคือ *Extrusion* โดยการนำ convex polygon n เหลี่ยม ที่เก็บค่าอยู่ใน vector2array v,{(v[i].x,v[i].y) | i=1..n} ที่กำหนดขึ้นในระบบ 2 มิติ จากนั้นเราจะกำหนดความหนาให้กับมันมีค่าเท่ากับ d ผลที่ได้ก็คือ จะได้รูปทรงในระบบ 3 มิติ ที่มีจำนวนด้านเท่ากับ 2+n (ประกอบด้วยด้านหน้าและด้านหลังของวัตถุสองด้าน และด้านข้างของวัตถุ n ด้าน) ข้อสังเกตุในการกำหนดภาพในระบบ 2 มิตินั้น รูปหลายเหลี่ยมจะต้องมีการเรียงตัว

แบบทวนเข็มนาฬิกาและเมื่อเราแสดงภาพโดยมองจากภายนอกวัตถุ ด้านต่างๆก็จะต้องมีการเรียงตัวแบบทวนเข็มนาฬิกาด้วย

วิธี Body of Revolution

เป็นวิธีที่สองในการกำหนดภาพในระบบ 2 มิติ และนำไปแสดงในระบบ 3 มิติ โดยวิธีนี้เราจะกำหนดส่วนหนึ่งของภาพขึ้นในระนาบ 2 มิติ (ระนาบ x/y ที่ผ่าน origin) และเราจะหมุนทุกๆส่วนของลายเส้นที่เรากำหนดนี้ไปในทิศทวนเข็มนาฬิกาด้วยมุม 2*i/nhoriz เรเดียน (1<=i<=nhoriz) ไปรอบๆแกน y จนถึงตำแหน่ง nhoriz Routine bodyrev ใน *listing 9.7* จะสร้างข้อมูลสำหรับวิธี body revolution ซึ่งรูปหลายเหลี่ยมที่เรากำหนดขึ้นในระบบสองมิติจะต้องมีการเรียงตัวแบบทวนเข็มนาฬิกา เพื่อได้ด้านต่างๆของวัตถุที่มีการเรียงตัวแบบทวนเข็มนาฬิกาด้วย

Three Dimension Animation

เราสามารถใช้เทคนิคในบทที่ 5 มาสร้างภาพเคลื่อนไหวในระบบ 3 มิติได้ ด้วยการสร้าง loop เพื่อแสดงภาพใน routine scene โดยการใช้ matrix เปลี่ยนค่า ACTUAL ให้แตกต่างกัน หรือจะใช้การเปลี่ยนจุดสังเกตุก็ได้

ในบทที่สิบ เราสามารถเขียน wire diagram ของรูปทรงได้แล้ว แต่ในความเป็นจริงวัตถุนั้นจะเป็นรูปทรงตันซึ่งเหลี่ยมของภาพทางด้านหน้าจะบังภาพที่อยู่ทางด้านหลังม้นออกไป ซึ่งในการสร้างภาพนั้นเราจะต้องสร้าง routine ที่แสดงว่า พื้นผิวหรือเส้นใดสามารถมองเห็นหรือไม่จากจุดสังเกต ซึ่งจะเรียกว่า hidden surface หรือ hidden line algorithm ซึ่งมีมากมายหลายรูปแบบ ในที่นี้เราจะพิจารณาจากรูปแบบที่ง่ายที่สุดไปจนถึงที่นำเอาไปใช้งานได้

Algorithm ในบทนี้จะทำให้เราสร้างรูปสีใน 3 มิติได้ เป็นอันดับแรกโดย routine seefacet(k) จะใช้แสดง facet k หรือพื้นผิวอื่นๆ โดย routine นี้จะไปเรียก facetfill ซึ่งใช้ในการให้สี (ในบทต่อไปจะแสดงผิวที่ซับซ้อนขึ้น) routine drawit จะทำหน้าที่ เรียกใช้ routine เหล่านี้รวมคือ hidden เพื่อใช้กำจัด hidden surface หรือ hidden line จากนั้นจะไปเรียก seefacet ต่อไป

ซึ่งภายในเนื้อหาจะประกอบด้วยเรื่อง ของ hidden surface algorithm สำหรับ single Closed convex Body , the painter's algorithm หรือหลักการ(back-to-front) , การแสดงผิวแบบพิเศษในระบบ 3 มิติ

ในบทที่สิบเอ็ด เป็นเรื่องเกี่ยวกับภาพ perspective,การคำนวนตำแหน่งของ prespec

tive transformation,การใช้ระยะของ viewplane,Stereoscopic views

ในบทที่สิบสอง พูดถึงเรื่อง Hidden line algorithm

ในบทที่สิบสาม เป็นเรื่องของ More General Hidden surface Algorithm

มีวิธีการมากมายในการสร้าง hidden line หรือ hidden surface algorithm วิธีหนึ่งที่จะกล่าวถึงก็คือ การใช้ array สองมิติ ในการกำหนด pixel ทั้งหมดของจอภาพ จากนั้นเราก็สมมุติว่า แสงที่ผ่านมาสู่ตานั้นสะท้อนมาจาก pixel ของจอภาพ ซึ่งแสงนี้ก็จะผ่านวัตถุบนจอภาพด้วย เราสามารถหาจุดตัดของแสงกับวัตถุได้ โดยที่จะเก็บค่า 'z co-Ordinate ' ของจุดตัดที่ใกล้ตาที่สุด ดังนั้นเราสร้างภาพใดๆบนจอขึ้นมาหรือสร้างผ่านเพิ่มขึ้นมาอีกได้ ด้วยการหาจุดตัดเหล่านี้ เทคนิคนี้นำไปใช้ได้ดีกับกรรมวิธีการไล่ Shade แต่อย่างไรก็ตาม Algorithm นี้ต้องอาศัยคอมพิวเตอร์ที่มีประสิทธิภาพสูง ในบทนี้เราจะแนะนำถึง Algorithm แบบอื่นที่เหมาะสมสำหรับใช้กับคอมพิวเตอร์ทั่วๆไป และใช้ร่วมกับจอภาพแบบ Raster-scan Display devices โดยใช้หลักการของ 'Back To Front' ที่ได้กล่าวถึงในบทที่ 10

เมื่อถึงตอนนี้เราสามารถสร้างภาพ 3 มิติได้อย่างสมบูรณ์แล้ว ในบทต่อๆไปจะเป็นวิธีการช่วยในการสร้างภาพให้เหมือนจริงขึ้น เช่น วิธีการ three dimensional clipping, shading,shadow,reflection เป็นต้น ซึ่งเราสามารถนำ routine ต่างมาที่จะกล่าวถึงในบทต่อๆไปใส่เข้าไปในโปรแกรมของเราได้ โดยไม่ต้องเปลี่ยนแปลงโปรแกรมมากเกินไป ซึ่งยังคงเป็นหลักการหลักของหนังสือเล่มนี้

ในบทที่สิบสี่ เป็นเรื่อง Three-dimensional Clipping ซึ่งในบทที่ 5 เราได้พิจารณาถึงการ clip ในระบบ 2 มิติ ซึ่งวิธีนี้เราจะนำมาประยุกต์ใช้ในระบบ 3 มิติในแบบ Orthographic project โดยเราจะ Clip จากภาพที่ถูก projection แล้ว ในระนาบ 2 มิติแทน ส่วนใน perspective projection เราจะต้องใช้กรรมวิธีที่ซับซ้อนกว่า เมื่อเราพิจารณา viewplane มีระยะทางจาก eye ไปตามแกน z ด้านลบ d ใน OBSERVER system rectangular window(horiz*vert) ก็คือขนาดของ viewplane ที่กำหนดให้เท่ากับ graphic viewport นั้น ซึ่งหากเมื่อเป็นภาพใหญ่(เป็นภาพ landscape) และเราต้องการมองเพียงบางส่วน หรือเมื่อเราให้จุด eye อยู่ในภาพนั้น จุดต่างๆที่อยู่ทางด้านบวกของแกน z จะต้องไม่ถูก project ซึ่งกรณีเหล่านี้ไม่สามารถใช้วิธีปกติได้ เราจึงคิดวิธีการ Three-

dimensional clipping ขึ้นมา เพื่อกำหนดว่าจุดไหนเส้นไหนๆสามารถถูก project ลงบน viewplane ได้ ก่อนที่จะทำการ projection

ในที่นี้การทำ Three-dimension clipping หลักการใหญ่ๆก็คือ

''''*The Pyramid of Vision*

Polygon Clipping in Three Dimension

ในบทที่สิบห้า เป็นเรื่องของ Shading

บทนี้และบทต่อไปจะแสดงถึงวิธีการในการแสดงภาพให้ดูสมจริงมากขึ้น ในบทนี้จะกล่าวถึงการแรงเงาวัตถุ ซึ่งหมายถึงความแตกต่างของสี ที่เกิดจากแสงตกกระทบวัตถุแล้วสะท้อนมาสู่ตา ซึ่งจะทำให้เกิดความแตกต่าง โดยขึ้นอยู่กับคุณสมบัติต่างๆของผิววัตถุ และคุณสมบัติของแสงที่ตกกระทบ ในทางปฏิบัติเราจะกำหนดให้แสงนั้นประกอบด้วย เส้นรังสีจำนวนมาก(ray หรือ beam) ซึ่งจะกำหนดด้วยเวคเตอร์ ในการแรงเงาจะพิจารณาถึงแหล่งกำเนิดแสงที่มีสองลักษณะ คือแหล่งกำเนิดแสงที่มาจากจุดๆเดียว(point source) โดยรังสีของแสงจะพุ่งออกมาจากแหล่งกำเนิดแสงมีลักษณะเป็นรัศมี เช่น แสงที่เกิดจากหลอดไฟ หรือ แสงที่มาจากดวงอาทิตย์ ในกรณีของแสงที่มาจากดวงอาทิตย์นั้น เราอาจจะพิจารณาได้ว่า รังสีของแสงที่มาจากดวงอาทิตย์จะมีลักษณะขนานกัน เนื่องจากดวงอาทิตย์นั้นอยู่ไกลจากเรามาก นั่นก็คือลักษณะของแหล่งกำเนิดแสงแบบที่สอง ซึ่งจะสมมุติให้แหล่งกำเนิดแสงอยู่ห่างจากวัตถุ infinity มันจะให้แสงที่มีลักษณะเป็นลำแสงขนาน(parallel beam illumination)

ทั้งในกรณีของลำแสงขนานและลำแสงที่เกิดจากจุดกำเนิดจุดเดียว(point source) จะถูกแทนด้วยเวคเตอร์ที่สัมพันธ์กับ OBSERVER system ซึ่งตำแหน่งของจุดกำเนิดแสงจะถูกแทนด้วยเวคเตอร์ s และทิศทางของลำแสงในแบบของลำแสงขนานจะถูกกำหนดด้วยเวคเตอร์ -l ในการคำนวนหาแสงที่สะท้อนจากจุด p ใดๆบนพื้นผิวของวัตถุ เราจะต้องกำหนด normal ของพื้นผิวที่จุด p นั้นด้วย n และเวคเตอร์บอกทิศทางจากจุด p ไปยังจุดกำเนิดแสง ในกรณีของลำแสงขนานเวคเตอร์บอกทิศทางจากทุกๆจุด p ก็คือเวคเตอร์ l นั้นเอง ในกรณีของลำแสงที่เกิดจากจุดกำเนิดจุดเดียวนั้นเวคเตอร์บอกทิศทางก็คือ s-p ในการคำนวนการสะท้อนแสงนี้เราจำเป็นต้องรู้จุดสังเกตุ(eye) ซึ่งก็คือจุด origin ของ OBSERVER system นั้นเอง

ซึ่งประกอบด้วยเรื่อง สีและความเข้มของแสง

สีของพื้นผิว

การสะท้อนของแสง

การพิจารณาการไล่ shade

- ambient reflection
- diffuse reflection
- specular reflection
- transperent surface

การนำวิธีการ shading ไปใช้ในโปรแกรม

- วิธีการสุ่มตัวอย่าง (random sampling)
- Pixel patterns
- RGB colour
- Constant shading
- Gouraund shading
- Phong interpolation

ในบทที่สิบหก จะเป็นเรื่องเกี่ยวกับ Shadows,Transparent Surfaces และ Refections ซึ่งเนื้อหาประกอบด้วยการคำนวนหา shadow polygon , การแสดงภาพของ shadow polygon , การหารูปเหลี่ยมที่เกิดจากเงาตกทอด , พื้นผิวที่โปร่งใส(Transperent surface) , การสะท้อน(reflection)

นอกจากทฤษฎีต่างที่ได้จัดทำเป็นหนังสือแล้ว ผู้จัดทำยังได้นำทฤษฎีต่างมาทดลองให้เห็นผลจริงและนำมาทดลองประยุกต์ใช้เป็นตัวอย่างต่างๆไว้ด้วย

สรุปผลการทดลอง

ข้อสรุปของการทำ THESIS นี้ หลังจากศึกษาและทำความเข้าใจแล้วเรียบเรียงขึ้นมาเป็นถ้อยคำ ที่ผู้จะนำไปศึกษาสามารถทำความเข้าใจได้โดยง่ายแล้ว ยังรวมไปถึงการนำ routine ต่าง ๆ ไปเขียนขึ้นเป็นโปรแกรมตัวอย่าง เพื่อแสดงผลสนับสนุนหลักการต่าง ๆ และการทำงานของ routine (โปรแกรมต่าง ๆ เหล่านี้ จะนำไปแสดงในวัน present เพื่อป้องกันการสูญหาย) พบว่า routine ที่เขียนขึ้นสอดคล้องและทำให้เกิดความเข้าใจในหลักการได้เป็นอย่างดี

ปัญหาที่พบ

ปัญหาที่เกิดขึ้นคือ

- หาหนังสืออ้างอิงเป็นภาษาไทยที่สอดคล้องกับอัลกอริทึมนี้ได้ยาก
- ความยากในการทำความเข้าใจในบาง routine ทำให้เวลานำไปใช้งานและเกิด BUG ขึ้นในโปรแกรม แล้วต้องใช้เวลาในการแก้ไขมาก
- เนื่องจากหนังสือที่ใช้เป็นภาษาอังกฤษ ต้องใช้เวลาในการทำความเข้าใจ และเรียบเรียงเป็นภาษาไทยเพื่อให้ผู้อื่นเข้าใจได้ด้วยตามจุดประสงค์ที่ต้องการให้เป็นแนวทางการศึกษาเบื้องต้นนั้น การจัดทำต้องระมัดระวังอย่างมาก เพื่อให้เกิดข้อผิดพลาดน้อยที่สุด
- ปัญหาในการเขียนโปรแกรม และการกำหนด database ในการ link routine ต่างๆเข้าด้วยกัน

ข้อเสนอแนะ

- การทำความเข้าใจ สำหรับผู้ที่จะนำ THESIS นี้ไปเป็นแนวทาง เพื่อความเข้าใจที่ดียิ่งขึ้นควรศึกษาและทดลองควรนำ routine ต่าง ๆ ในหนังสือเล่มนี้ ไปทดลองเขียนโปรแกรม และสังเกตผลลัพธ์ที่ได้
- ในการประยุกต์ใช้จะเกิดปัญหาในการ link routine ต่างๆเข้าด้วยกัน จะต้องให้ความสำคัญในการผ่านค่าของ database และการเขียนโปรแกรมที่เรียกใช้ routine ต่างๆเหล่านั้นให้ถูกต้องและเหมาะสม และเพื่อใช้งานตามจุดประสงค์ที่ต้องการต้องอาศัยความระมัดระวังและความเข้าใจในการทำงานของมันเป็นอย่างดี

เอกสารอ้างอิง

- Ian O.angell and Gareth Griffith(1988),High-resolution Computer Graphics Using Pascal , Macmillan Education.

- David F. Rogers and J. Alan Adams (1990) , Mathematical Elements for Computer Graphic , McGRAW-HILL.

- Erwin KreyZig (1988) . Advance Engineering Mathematics. Wie Wieley

- มงคล อัศวโกวิทกรณ์ (1992), การเขียนโปรแกรมกราฟิกส์ , บริษัท ดวงกมล จำกัด

- ชัชวาล ยนต์หงส์ , แนะนำภาษา Pascal โดย Turbo pascal , โอเดียนสโตร์

- บุญนิศ เอี่ยมทัศนา,เรียนรู้ ภาษาปาสคาลด้วยภาษาเทอร์โบปาสคาล 4.0-4.5,ซีเอ็ด

- TURBO PASCAL version 6.0 , TURBO LIBRARY REFERENCE

กิตติกรรมประกาศ

การที่ Project สามารถดำเนินไปด้วยดีนั้น ต้องขอขอบคุณ อาจารย์ที่ปรึกษา และอาจารย์ทุก ๆ ท่าน ที่คอยประสิทธ์ประสาทวิชาความรู้ ตลอดจนทุกท่านที่ให้ความช่วยเหลือ และข้อแนะนำต่างๆ

อานิสงส์ ที่ได้จากทำ Project นี้ ขออุทิศให้กับเด็กผู้ด้อยโอกาสทางการศึกษาทุกคน