



การสร้างภาพ 3 มิติ

3 DIMENSIONAL CONSTRUCTION

โดย

นายขจรศักดิ์ กุศลภินันท์

นางสาวรุ่งมณี แยมรุ่ง

นายสุรเชษฐ์ เสถียรจรรรัตน์

วัน เดือน ปี..... 23.ค.ค. 2541
เลขทะเบียน..... 039133
เลขเรียกหนังสือ..... T 40372 2106ก

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2540

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้ง

039133

การสร้างภาพ 3 มิติ
3 DIMENSIONAL CONSTRUCTION



โดย
นายขจรศักดิ์ กุศลภินันท์
นางสาวรุ่งมณี แี่ยมรุ่ง
นายสุรเชษฐ์ เสถียรจารุรัตน์

อาจารย์ที่ปรึกษา
รศ.ดร. มนัส สังวรศิลป์

ปริญญานิพนธ์สำหรับปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้เรียนเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ปีการศึกษา 2540
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ปริญญานิพนธ์ปีการศึกษา 2540

ภาควิชา วิศวกรรมอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
เรื่อง การสร้างภาพ 3 มิติ

ผู้จัดทำ

1. นายขจรศักดิ์ กุศลภินันท์
2. นางสาวรุ่งมณี แย้มรุ่ง
3. นายสุรเชษฐ์ เสถียนจารุรัตน์



(รศ.ดร. มนัส สังวรศิลป์)

อาจารย์ที่ปรึกษา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การสร้างภาพ 3 มิติ

3 DIMENSIONAL CONSTRUCTION

1. นายขจรศักดิ์ กุศลภินันท์
2. นางสาวรุ่งมณี เข้มรุ่ง
3. นายสุรเชษฐ์ เสถียนจารรัตน์

รายงานฉบับนี้ได้ผ่านการตรวจสอบโดยอาจารย์ที่ปรึกษาแล้ว

ลงชื่อ.....อาจารย์ที่ปรึกษา

(รศ.ดร. มนัส สังวรศิลป์)

วันที่...../...../.....

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

รายงานฉบับนี้สำเร็จลุล่วงลงได้ด้วยความช่วยเหลือจากบุคคลหลายฝ่าย ทางคณะผู้จัดทำขอกราบขอบพระคุณเป็นอย่างสูงไว้ ณ ที่นี้ โดยเฉพาะอย่างยิ่ง รศ.ดร.มนัส สังวรศิลป์ ที่ให้คำปรึกษาและคำแนะนำในการทำโครงการชิ้นนี้ ขอขอบคุณบุคคลอื่น ๆ ซึ่งให้ความช่วยเหลือที่ไม่ได้กล่าวถึงไว้ ณ ที่นี้ ขอบคุณกำลังใจจากเพื่อน ๆ ไว้ ณ ที่นี้ด้วย

รองศาสตราจารย์ กุศลภรณ์

(นายขจรศักดิ์ กุศลภรณ์)

(นางสาวรุ่งมณี เข้มรุ่ง)

(นายสุรเชษฐ์ เสถียนจารรัตน์)

ผู้จัดทำ

การสร้างภาพ 3 มิติ

นายขจรศักดิ์ กุศลภินันท์
นางสาวรุ่งมณี เข้มรุ่ง
นายสุรเชษฐ์ เสถียรจารุรัตน์
อ.มนัส สังวรศิลป์ (อาจารย์ที่ปรึกษา)
ภาคการศึกษาที่ 2 ปีการศึกษา 2540

บทคัดย่อ

ปัจจุบันได้มีการใช้ภาพคอมพิวเตอร์กราฟิก 3 มิติในงานด้านต่าง ๆ มากมาย ประกอบกับความเร็วและความสามารถที่มากขึ้น ราคาที่ถูกลง โปรแกรมสำเร็จรูปที่ใช้งานได้ง่ายขึ้นของคอมพิวเตอร์ ซึ่งปัจจัยเหล่านี้ทำให้การสร้างภาพคอมพิวเตอร์ 3 มิติเป็นงานที่ง่ายขึ้นกว่าแต่ก่อนมาก ทำให้การสร้างภาพคอมพิวเตอร์กราฟิก 3 มิติเป็นที่นิยมมากขึ้น

รายงานฉบับนี้เกี่ยวข้องกับขบวนการได้มาของการสร้างภาพและการเขียนโปรแกรมคอมพิวเตอร์เพื่อใช้ในการสร้างภาพ 3 มิติโดยอาศัยคอมพิวเตอร์ส่วนบุคคลพร้อมทั้งแสดงการประยุกต์ใช้งานในการจำลองสถานการณ์จริง

3 DIMENSIONAL CONSTRUCTION

Mr. Khajohnsak Kusalapinun

Miss Roongmanee Yamroong

Mr. Surachade Stainjarurat

Mr. Manas Sangworasil(Advisor)

2nd Semester, Educational Year 1997

Abstract

In the present time 3 dimensional computer graphics have been a various number of application areas. As a result of the faster and more ability, lower prize and many program computer applications made the 3 dimensional computer graphics is easy and popular to be used.

This report is about the process of construction and programming for 3 dimensional computer graphics by personal computer and shows how to apply in virtual simulation

สารบัญ

เรื่อง	หน้า
กิตติกรรมประกาศ	I
บทคัดย่อ	II
Abstract	III
สารบัญ	IV
สารบัญภาพ	VII
บทที่ 1 บทนำ	1
1.1 ความเป็นมาของโครงการ	1
1.2 วัตถุประสงค์	1
1.3 ขอบเขตของโครงการ	1
บทที่ 2 กราฟิก 2 มิติ และกราฟิก 3 มิติ	2
2.1 กราฟิก 2 มิติ	2
2.1.1 จุด	2
2.1.2 เส้นตรง	2
2.1.3 รูปหลายเหลี่ยม	3
2.1.4 การเปลี่ยนแปลงตำแหน่ง	4
2.4.1.1 การเปลี่ยนแปลงขนาด	4
2.4.1.2 การเคลื่อนย้ายตำแหน่ง	5
2.4.1.3 การเปลี่ยนแปลงตำแหน่งการหมุน	6
2.1.5 หน้าต่าง ช่องมองหน้าต่าง และการตัดขอบภาพ	8
2.1.6 การตัดส่วนของภาพ	10
2.1.6.1 อัลกอริทึมของ Cohen – Suhterland	10
2.1.6.2 วิธีของ Sutherland - Hodgman	12

2.2 กราฟิก 3 มิติ	13
2.2.1 จุดและระนาบ	13
2.2.2 การเปลี่ยนตำแหน่งในระบบ 3 มิติ	14
2.2.3 การหมุนวัตถุรอบแกนใด ๆ	16
2.2.4 การฉายภาพแบบขนานและการฉายภาพแบบเพอสเปคตีฟ	17
2.2.4.1 การฉายภาพแบบขนาน	17
2.2.4.2 การฉายภาพแบบเพอสเปคตีฟ	19
2.2.5 การตัดส่วนของภาพในระบบ 3 มิติ	20
2.2.6 การซ่อนเส้นและพื้นผิว	22
2.2.6.1 การใช้ Z BUFFERS	22
2.2.6.2 อัลกอริทึม SCAN LINE	23
2.2.6.3 อัลกอริทึมของ PAINER	23
บทที่ 3 การแสดงวัตถุ	25
3.1 การแสดงและแบบจำลองของวัตถุพีชคณิต	25
3.2 polygonal representation	25
บทที่ 4 อัลกอริทึมของการเรนเดอร์	29
4.1 บทนำ	29
4.2 การเลือกและการตัด (culling and clipping)	29
4.3 Incremental shading techniques	32
4.3.1 Gouraud shading	32
4.3.2 Phong interpolation	35
4.3.3 เปรียบเทียบวิธีทั้งสอง	37
4.4 Rasterization หรือ scan conversion	37
4.4.1 rasterizing edges	38
4.4.2 rasterizing polygon	38
บทที่ 5 RAY TRACING	40
5.1 การจำลองภาพแบบ Ray Tracing	40
5.1.1 โดยวิธีติดตามแสงจากต้นกำเนิดแสง	40

5.1.2 โดยวิธีติดตามทิศทางการมองของผู้สังเกต	40
บทที่ 6 แสง สี และลำดับความเข้ม	42
6.1 การสะท้อนแสงของวัตถุ	42
6.2 แสงและสี	50
6.2.1 ไดอะแกรม CIE chromaticity	51
6.2.2 แบบจำลองของสี	52
6.2.2.1 แบบจำลองของสีแบบ RGB	53
6.2.2.2 แบบจำลองของสีแบบ CMY	53
6.2.2.3 แบบจำลองของสีแบบ YIQ	54
6.2.2.4 แบบจำลองของสีแบบ HSV	54
6.2.2.5 แบบจำลองของสีแบบ HLS	56
6.2.3 ทฤษฎีการกำหนดความเข้มของแสง	56
6.2.3.1 การกำหนดความเข้มของแสงแบบคงที่	57
6.2.3.2 การกำหนดความเข้มของแสงแบบ GOURAUD	58
6.2.3.3 การกำหนดความเข้มของแสงแบบ PHONG	59
บทที่ 7 OPENGL	62
7.1 เบื้องต้นเกี่ยวกับ OPENGL	62
7.2 หลักการทำงานเบื้องต้น	62
7.3 สถาปัตยกรรมกราฟฟิก	62
บทที่ 8 ผลการทดลอง	64
8.1 ผลการทำงานของโปรแกรมสร้างภาพ 3 มิติ	64
8.2 ผลการทำงานของโปรแกรมควบคุมแกนกล	77
บทที่ 9 สรุปผลและวิจารณ์	80
เอกสารอ้างอิง	
ภาคผนวก	

สารบัญภาพ

ภาพที่	หน้า
2.1 จุดในระบบกราฟิก 2 มิติ	2
2.2 เส้นตรงในระบบกราฟิก 2 มิติ	3
2.3 รูปหลายเหลี่ยม	4
2.4 การขยายรูปหลายเหลี่ยมในแนวแกน X	5
2.5 การเคลื่อนย้ายตำแหน่ง ของวัตถุ	6
2.6 การหมุนของจุด (1,0) รอบจุดศูนย์กลาง	7
2.7 การหมุนจุด (0,1) รอบจุดศูนย์กลาง	7
2.8 การแสดงผลเมื่อมีหน้าต่างที่ต่างกันแต่ช่องมองภาพเดียวกัน	9
2.9 การแสดงผลเมื่อมีหน้าต่างเดียวกันแต่ช่องมองภาพต่างกัน	9
2.10 การขยายส่วนของภาพจำลอง โดยช่องมองภาพ	10
2.11 การตัดส่วนของภาพ	10
2.12 แสดงค่าของตำแหน่งของจุดต่าง ๆ รอบหน้าต่าง	11
2.13 แสดงลักษณะเส้นตรงต่าง ๆ รอบหน้าต่าง	12
2.14 การตัดส่วนของภาพที่บริเวณขอบของหน้าต่าง	12
2.15 แสดงตำแหน่งของจุดในระบบ 3 มิติ	13
2.16 การหมุนรอบแกน Z	15
2.17 การหมุนวัตถุรอบแกนใด ๆ	17
2.18 การฉายภาพแบบขนาน	18
2.19 การฉายภาพแบบเพอสเปคตีฟ	20
2.20 ปริมาตรการมองสำหรับการฉายภาพแบบขนาน	21
2.21 ปริมาตรการมองสำหรับการฉายภาพแบบเพอสเปคตีฟ	21
2.22 อัลกอริทึมของ Painter	24
3.1 การประมาณค่าพื้นผิวโค้ง โดยใช้รูปหลายเหลี่ยม	26

3.2 ลำดับย่อยของการแสดงวัตถุ	27
3.3 แสดงถึงการใช้ขอบร่วมกัน	28
4.1 การเลือกหรือการตัดผิวหลัง	30
4.2 การตัดแบบ Sutherland-Hodgman	30
4.3 การตัดแบบ Sutherland-Hodgman ภายในรูปหลายเหลี่ยม	31
4.4 Dot product ซึ่งใช้หาว่าเส้นอยู่ภายในหรือภายนอกของเขต	32
4.5 แสดงเวกเตอร์ N_A ซึ่งเฉลี่ยมาจากเวกเตอร์ N_1, N_2, N_3 และ N_4	33
4.6 นอร์มอลเวกเตอร์และการตัด	34
4.7 แสดงการใช้สมการหาค่าความเข้มแสง	34
4.8 ขั้นตอนการเรนเดอร์ภาพ 3 มิติ สำหรับ Gouraud shading	35
4.9 เวกเตอร์ความเข้มแสงสร้างขึ้นในวิธีการแบบ Phong	36
4.10 เวกเตอร์ที่สร้างขึ้นเพื่อสร้างเส้นโค้ง	36
4.11 ลำดับจุดที่ใช้ใน (a) การวาดเส้น (b) โพลิกอนฟิลลิง	38
4.12 ตัวอย่างการรักษาการเชื่อมต่อใน polygon rasterization	39
5.1 การจำลองภาพแบบ Ray Tracing โดยวิธีคิดตามแสงจากต้นกำเนิดแสง	40
5.2 การจำลองภาพแบบ Ray Tracing โดยวิธีคิดตามทิศทางการมองของผู้สังเกต	41
6.1 แสดงการตกกระทบของแสงจากต้นกำเนิดแสง	42
6.2 การตกกระทบของแสงจากต้นกำเนิดแสงแบบกระจาย	43
6.3 มุมตกกระทบของแสง	44
6.4 การสะท้อนของแสงแบบกรวย	45
6.5 แสงที่ตกกระทบกับพื้นผิที่มุมต่าง ๆ	46
6.6 ทิศทางของแสงที่ตกกระทบ เมื่อต้นกำเนิดแสงอยู่ใกล้เคียงกับวัตถุ	46
6.7 มุมของแสงตกกระทบ	47
6.8 การสะท้อนของแสงที่พื้นผิวต่าง ๆ	48
6.9 มุมตกกระทบของแสงและมุมมอง	49
6.10 ความยาวของคลื่นแม่เหล็กไฟฟ้าที่สามารถมองเห็นได้	50
6.11 สีต่าง ๆ ที่มองเห็นได้	51
6.12 การสร้างมาตรฐาน CIE	52

6.13 ไคอะแกรม CIE Chromaticity	52
6.14 แบบจำลองสีแบบ RGB	53
6.15 แบบจำลองสีแบบ HSV	55
6.16 แบบจำลองสีแบบ HLS	56
6.17 ทิศทางการสะท้อนของแสง	57
6.18 วิธี Constant Shading	58
6.19 การหาความเข้มแสงโดยวิธี Gouraud	59
6.20 การกำหนดความเข้มแสงแบบ Gouraud	60
6.21 การกำหนดความเข้มแสงแบบ Phong	60
8.1 หน้าจอของโปรแกรมสร้างภาพ 3 มิติ	64
8.2 wireframeของbox	65
8.3 wireframeของtetrahedron	65
8.4 wireframeของoctahedron	66
8.5 wireframeของicosahedron	66
8.6 wireframeของdodecahedron	67
8.7 wireframeของsphere	67
8.8 wireframeของcylinder	68
8.9 wireframeของcone	68
8.10 wireframeของtorus	69
8.11 surfaceของbox	70
8.12 surfaceของtetrahedron	70
8.13 surfaceของoctahedron	71
8.14 surfaceของicosahedron	71
8.15 surfaceของdodecahedron	72
8.16 surfaceของsphere	72
8.17 surfaceของcylinder	73
8.18 surfaceของcone	73
8.19 surfaceของtorus	74

8.20 smooth surfaceของsphere	74
8.21 smooth surfaceของcylinder	75
8.22 smooth surfaceของcone	75
8.23 smooth surfaceของtorus	76
8.24 หน้าจอแสดงแกนกลแบบ wireframe	78
8.25 หน้าจอแสดงแกนกลแบบ surface	78



บทที่ 1

บทนำ

1.1 ความเป็นมาของโครงการ

ปัจจุบันภาพคอมพิวเตอร์กราฟิก 3 มิติ ถูกนำมาใช้งานในด้านต่าง ๆ มากมาย ไม่ว่าจะเป็นทางด้านภาพยนตร์โฆษณา งานจำลองต้นแบบทางอุตสาหกรรม การจำลองภาพทางการแพทย์ ฯลฯ สืบเนื่องมาจากความสามารถที่สูงขึ้นของคอมพิวเตอร์ในปัจจุบัน ทำให้ภาพที่ได้มีความสวยงามคมชัดและดูแปลกตา จึงทำให้การสร้างภาพคอมพิวเตอร์กราฟิก 3 มิติเป็นที่นิยมเพิ่มมากขึ้น แต่เนื่องจากโปรแกรมที่ใช้ในการสร้างภาพคอมพิวเตอร์กราฟิก 3 มิติ ที่นิยมใช้งานในปัจจุบันเป็นโปรแกรมสำเร็จรูป และการใช้งานค่อนข้างยาก โครงการนี้จึงเกิดขึ้นเพื่อศึกษาถึงขั้นตอนการได้มาของภาพคอมพิวเตอร์กราฟิก 3 มิติ ซึ่งจะช่วยให้เข้าใจถึงวิธีการสร้างภาพ 3 มิติ มากขึ้น

1.2 วัตถุประสงค์

- เพื่อศึกษาถึงขั้นตอนและวิธีการต่าง ๆ ที่ใช้ในการสร้างภาพคอมพิวเตอร์กราฟิก 3 มิติ
- เพื่อช่วยให้เข้าใจถึงการใช้งาน โปรแกรมสร้างภาพคอมพิวเตอร์กราฟิก 3 มิติ มากยิ่งขึ้น
- เป็นการส่งเสริมให้เกิดความสนใจในเรื่องการสร้างภาพคอมพิวเตอร์กราฟิก 3 มิติ ซึ่งปัจจุบันกำลังขาดแคลนผู้เชี่ยวชาญในสาขานี้

1.3 ขอบเขตของโครงการ

- ศึกษาถึงกรรมวิธีและขั้นตอนต่าง ๆ ที่ใช้ในการสร้างภาพคอมพิวเตอร์กราฟิก 3 มิติมาแสดงให้เห็นจริงโดยการเขียนโปรแกรมสร้างภาพ 3 มิติ
- นำความรู้ที่ได้จากการศึกษาเกี่ยวกับการสร้างภาพคอมพิวเตอร์กราฟิก 3 มิติ มาแสดงให้เห็นจริงโดยการเขียนโปรแกรมสร้างภาพ 3 มิติ
- ทำการประยุกต์ใช้งานภาพ 3 มิติ

บทที่ 2

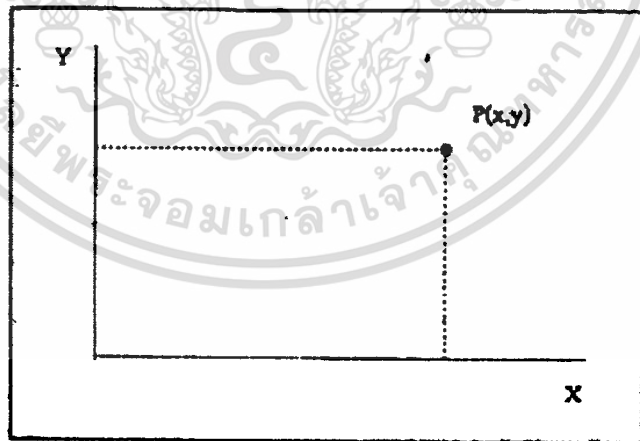
กราฟิก 2 มิติ และกราฟิก 3 มิติ

2.1 กราฟิก 2 มิติ

ระบบที่เป็นพื้นฐานของระบบคอมพิวเตอร์กราฟิก คือ ระบบกราฟิก 2 มิติ เพราะภาพที่ได้จากระบบกราฟิก 2 มิติ นี้สามารถนำไปใช้ประโยชน์ได้มากมาย เช่น รูปภาพต่าง ๆ ในระบบกราฟิก 2 มิติ นี้ ตำแหน่งของข้อมูลต่าง ๆ จะถูกแทนที่ได้โดยตัวแปร 2 ตัวด้วยกันคือ (x,y) เมื่อ x คือ ระยะทางในแนวนอนจากจุดกำเนิด (origin) ใด ๆ และ y คือ ระยะทางในแนวตั้งจากจุดกำเนิดเดียวกัน สิ่งที่สำคัญในระบบกราฟิก 2 มิติ คือ จุด (point) เส้นตรง (line) และรูปหลายเหลี่ยม (polygon) เนื่องจากสามารถที่จะนำสิ่งเหล่านี้ไปประกอบเพื่อให้เกิดเป็นภาพขึ้นมาตามความต้องการ

2.1.1 จุด

จุดในระบบกราฟิก 2 มิติจะถูกแสดงโดยพิกัด (coordinate) ของมันเองโดยค่า 2 ค่าด้วยกัน คือ $P(x,y)$ ดังแสดงในรูปที่ 2.1

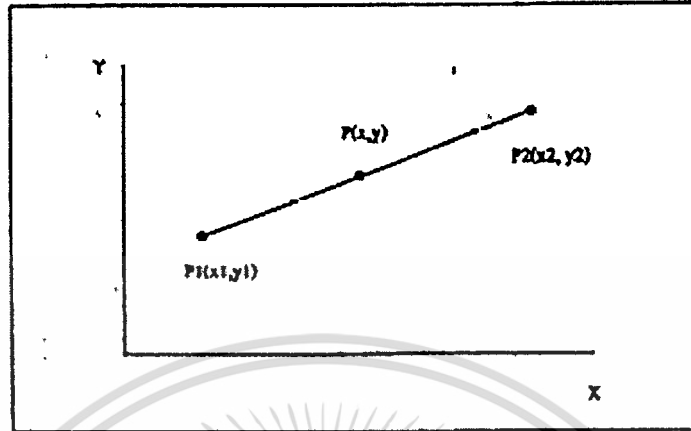


รูปที่ 2.1 จุดในระบบกราฟิก 2 มิติ

2.1.2 เส้นตรง

เส้นตรงในระบบกราฟิก 2 มิตินั้นเกิดจากระยะทางระหว่างจุด 2 จุด ดังรูปที่ 2.2

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.2 เส้นตรงในระบบกราฟิก 2 มิติ

สมการของเส้นตรงสามารถอธิบายได้ดังนี้ ถ้าจุด $P(x,y)$ เป็นจุดใด ๆ บนเส้นตรงที่เกิดจากจุด $P_1(x_1, y_1)$ และ $P_2(x_2, y_2)$ ใด ๆ แล้ว สมการของเส้นตรงคือ

$$y = mx + b \quad (1)$$

โดยที่

$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

$$b = y_1 - mx_1$$

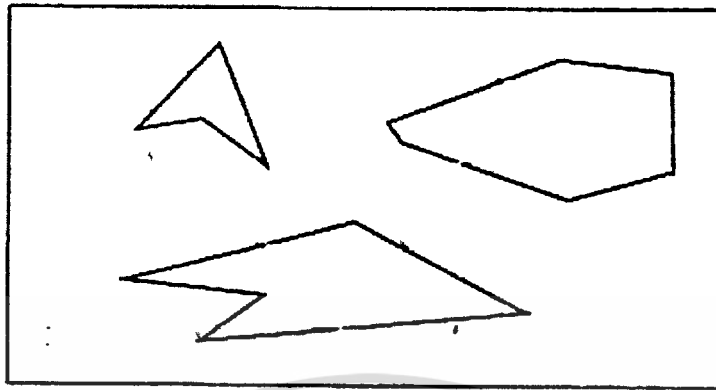
สมการที่ 1 เรียกว่าสมการเส้นตรงแบบ Slope-intercept

m = ความชัน (Slope)

b = จุดตัดแกน y ของสมการเส้นตรง ณ ตำแหน่งที่ x มีค่าเป็นศูนย์ $(0,b)$

2.1.3 รูปหลายเหลี่ยม

รูปหลายเหลี่ยม คือ รูปที่เกิดจากการรวมของจุดและเส้นตรงที่เชื่อมติดกันที่บริเวณปลายทั้งสองข้างดังรูปที่ 2.3 จะเห็นว่ารูปหลายเหลี่ยมที่เกิดขึ้นนั้นจะมีลักษณะเป็นภาพปิด (close figure) เรียกด้านที่ประกอบกันของรูปหลายเหลี่ยมว่าด้าน (side) หรือขอบ (edges) ของรูป ส่วนจุดปลายสุดของแต่ละด้านของรูปหลายเหลี่ยมจะเรียกว่าจุดรวม (vertices)



รูปที่ 2.3 แสดงรูปหลายเหลี่ยม

2.1.4 การเปลี่ยนแปลงตำแหน่ง

ในระบบคอมพิวเตอร์กราฟิกนั้นในบางครั้งมีความจำเป็นที่จะต้องทำการเปลี่ยนแปลงลักษณะของภาพที่ได้เพื่อนำผลที่ได้ให้สามารถนำไปใช้ประโยชน์ในด้านอื่น หรือ เพื่อความสะดวกแก่ผู้ใช้งาน โดยไม่ต้องสร้างภาพขึ้นมาใหม่ เช่น การขยายภาพ การลดขนาดภาพ หรือ การเปลี่ยนทิศทางของภาพเดิม แต่จะนำสมการทางคณิตศาสตร์บางอย่างมาช่วยในการเปลี่ยนแปลงลักษณะของภาพจำลอง

2.1.4.1 การเปลี่ยนแปลงขนาด

กำหนดให้ $P_1 = [x_1 \ y_1]$ เป็นจุดใด ๆ ที่มีขนาด 1×2 เมตริกซ์ (matrix) ถ้าคูณเมตริกซ์ P_1 ด้วยเมตริกซ์ T ที่มีขนาด 2×2 จะทำให้ได้เมตริกซ์ใหม่ที่มีขนาด 1×2 ด้วย ซึ่งสามารถให้แทนตำแหน่งอื่น ๆ ได้

$$[x_2 \ y_2] = P_2 = P_1 T \quad (2)$$

จากสมการที่ 2 จะเห็นว่าเมตริกซ์ T สามารถให้เป็นเมตริกซ์ที่เป็นตัวเชื่อมความสัมพันธ์ของตำแหน่งเดิม P_1 และตำแหน่งใหม่ P_2 ได้ หรือเมตริกซ์ T สามารถใช้เป็นตัวกำหนดขนาดของตำแหน่ง P_2 เมื่อเทียบกับตำแหน่ง P_1 ได้ซึ่งเมตริกซ์ T สามารถเขียนได้ดังนี้

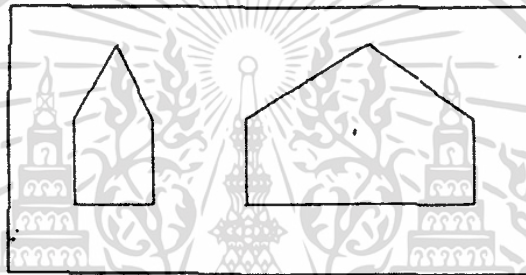
$$T = S = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \quad (3)$$

เมื่อ S = เมตริกซ์ของการขยายขนาด

S_x = สัดส่วนการขยาย (Scale factor) ในแนวแกน X

S_y = สัดส่วนการขยายในแนวแกน Y

ดังนั้นเมื่อนำมาใช้กับข้อมูลของรูปหลายเหลี่ยม จะทำให้สามารถที่จะลดหรือขยายสัดส่วนของภาพได้ตามต้องการ ดังแสดงให้เห็นในรูปที่ 2.4



รูปที่ 2.4 การขยายรูปหลายเหลี่ยมในแนวแกน X

2.1.4.2 การเคลื่อนย้ายตำแหน่ง

การเคลื่อนย้ายภาพจากตำแหน่งเดิมไปสู่ตำแหน่งใหม่นั้นเรียกว่า การเคลื่อนย้ายตำแหน่ง (translation) ดังนั้นเป็นการง่ายถ้าต้องการเคลื่อนย้ายตำแหน่งของวัตถุไป (t_x, t_y) หน่วย จากตำแหน่งเดิม (x_1, y_1) ไปสู่ตำแหน่งใหม่ (x_2, y_2) ดังแสดงในรูปที่ 2.5

$$\left. \begin{aligned} x_2 &= x_1 + t_x \\ y_2 &= y_1 + t_y \end{aligned} \right\} \quad (4)$$

แต่เนื่องจากในระบบคอมพิวเตอร์กราฟิกนั้น จะใช้สมการคณิตศาสตร์ที่เป็นระบบเมตริกซ์เป็นส่วนมาก ดังนั้นสมการที่ 4 จึงไม่เหมาะสมที่จะนำมาใช้ในระบบคอมพิวเตอร์การฟิกมากนัก เพื่อที่จะแก้ปัญหานี้จึงได้มีการนำเอาวิธีการของพิกัดแบบ homogeneous (homogeneous coordinate) มาใช้ซึ่งมีวิธีการดังนี้

ในระบบพิกัดแบบ homogeneous นั้นจะใช้เมตริกซ์ขนาด 3×3 แทนเมตริกซ์ขนาด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

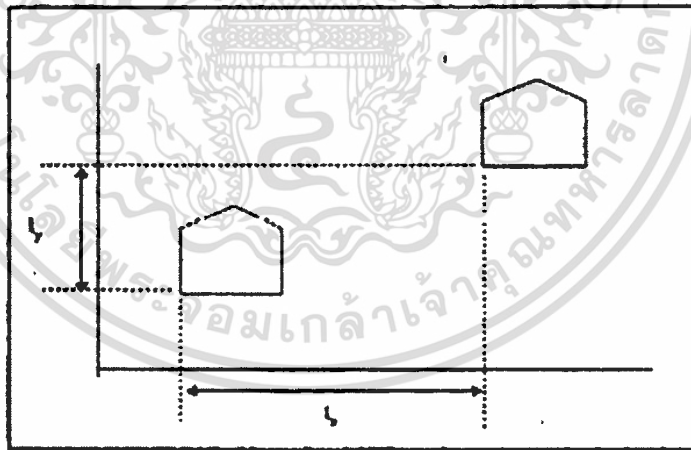
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2x2 โดยเพิ่มสมาชิกที่เรียกว่า พิกัดตัวผาก(dummy coordinate) คือ w ขึ้นมา จะต่าง ๆ จะถูกกำหนดโดยตัวเลข 3 ตัวแทนแบบเดิมที่ใช้ตัวเลข 2 ตัว สมาชิกตัวแรกของระบบนี้จะเป็นผลคูณของ x และ w สมาชิกตัวที่ 2 จะเป็นผลคูณของ y และ w ส่วนสมาชิกตัวที่ 3 คือ w ดังนั้นจุด (x,y) จะถูกแทนที่เป็น (xw,yw,w) โดยกำหนดให้ w มีค่าเท่ากับ 1 ดังนั้นในระบบนี้เมตริกซ์ของการเปลี่ยนแปลงตำแหน่งด้วยขนาด t_x, t_y คือ

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix} \quad (5)$$

และเมตริกซ์ของการเปลี่ยนแปลงขนาดคือ

$$S = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (6)$$

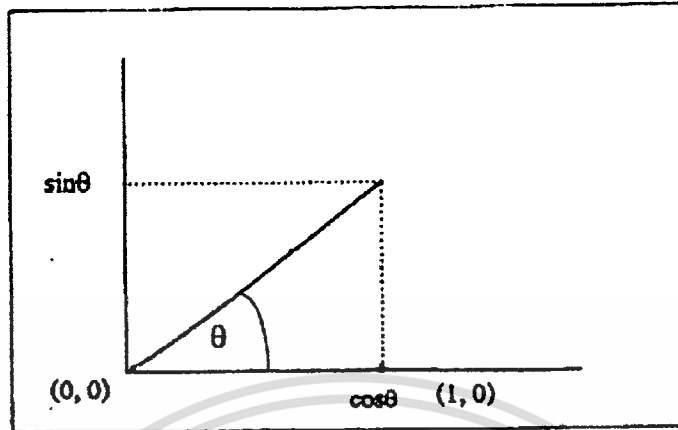


รูปที่ 2.5 การเคลื่อนย้ายตำแหน่งของวัตถุ

2.1.4.3 การเปลี่ยนแปลงตำแหน่งโดยการหมุน

ในการหาเมตริกซ์ของการเปลี่ยนแปลงตำแหน่งโดยการหมุน (Rotation) นั้นสามารถกระทำได้โดยพิจารณาจุดที่ตำแหน่ง $(1,0)$ ถ้าหมุนตำแหน่งของจุดนี้ในทิศทวนเข็มนาฬิกาด้วยมุม θ ดังรูปที่ 2.6 จะพบว่าที่ตำแหน่งใหม่จะมีค่า $(\cos\theta, \sin\theta)$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.6 การหมุนของจุด (1,0) รอบจุดศูนย์กลาง

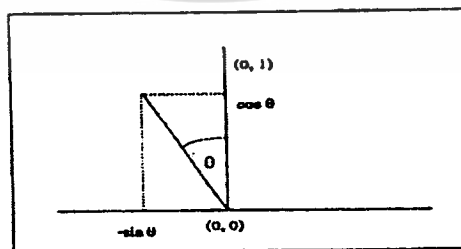
ดังนั้น

$$\begin{bmatrix} \cos \theta & \sin \theta \end{bmatrix} = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} a & b \end{bmatrix} \quad (7)$$

ในกรณีเดียวกันถ้าหมุนจุด (0,1) ในลักษณะเดียวกันตำแหน่งใหม่ที่ได้คือ $(-\sin \theta, \cos \theta)$ ดังรูปที่ 2.7

ดังนั้น

$$\begin{bmatrix} -\sin \theta & \cos \theta \end{bmatrix} = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} c & d \end{bmatrix} \quad (8)$$



รูปที่ 2.7 การหมุนจุด (0,1) รอบจุดศูนย์กลาง

จากสมการที่ 7 และ 8 สามารถที่จะกำหนดเมตริกซ์ของการเปลี่ยนแปลงตำแหน่งโดยการหมุนในทิศทางทวนเข็มนาฬิกาเป็นมุม θ รอบจุดศูนย์กลางได้ คือ

$$R = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} \quad (9)$$

และในระบบพิกัดแบบ homogeneous คือ

$$R = \begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (10)$$

2.1.5 หน้าต่าง, ช่องมองหน้าต่างและการตัดขอบภาพ

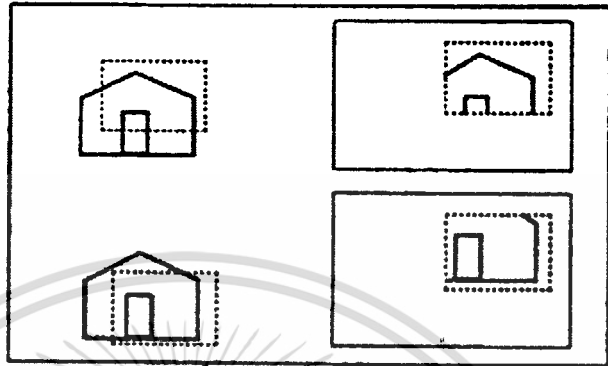
ภาพต่าง ๆ ที่ได้จากการจำลองภาพในระบบคอมพิวเตอร์กราฟิกนั้นบางครั้งจะเป็นภาพที่มีขนาดใหญ่และมีรายละเอียดในภาพมาก ซึ่งทำให้การมองภาพจำลองบนจอภาพของคอมพิวเตอร์ไม่สามารถที่จะบอกรายละเอียดของส่วนต่าง ๆ ของภาพได้ทั้งหมดภายในครั้งเดียว ดังนั้นจึงต้องมีการนำเอาวิธีการบางอย่างเข้ามาช่วยในการมองของภาพจำลองที่ได้ เพื่อให้สามารถแสดงรายละเอียดบางส่วนของการจำลองได้ วิธีการที่นำมาช่วยในการมองภาพจำลองนั้นเรียกว่าหน้าต่าง (Window)

หน้าต่างนั้นสามารถทำให้การแสดงผลภาพในส่วนที่ต้องการได้ไม่ว่าจะเป็นการแสดงผลภาพจำลองทั้งภาพ หรือเฉพาะบางส่วนของภาพที่ต้องการแสดงรายละเอียดเป็นพิเศษ ขนาดของหน้าต่างสามารถที่จะกำหนดขนาดได้ตามต้องการ ไม่ว่าจะเป็นหน้าต่างที่สามารถแสดงผลภาพทั้งหมดของภาพจำลองหรือการแสดงผลเพียงบางส่วนของภาพจำลองก็ได้ การนำส่วนของภาพจำลองที่เลือกไว้มาแสดงผลบนจอภาพนั้นยังสามารถที่จะกำหนดได้ว่าสามารถที่จะแสดงที่ส่วนใดของจอภาพ และมีขนาดเท่าใดของจอภาพ ช่องการแสดงผลบนจอภาพนั้นเรียกว่า ช่องมองภาพ (viewport)

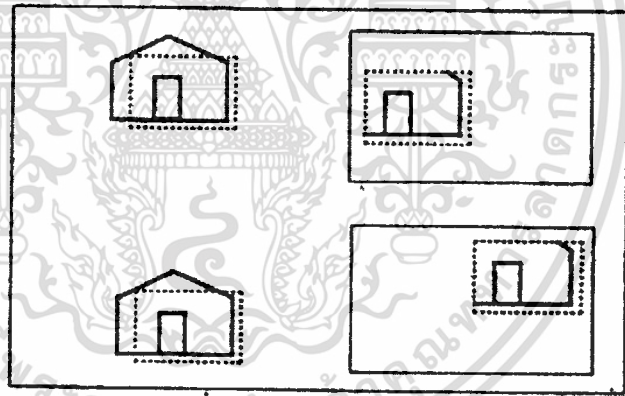
ส่วนของภาพจำลองที่เลือกโดยขนาดของหน้าต่างนี้จะถูกนำมาแสดงในช่องมองภาพและถึงแม้ว่าขนาดของภาพที่ถูกกำหนดโดยหน้าต่างและช่องมองภาพจะมีขนาดแตกต่างกันหรือตำแหน่งต่างกัน เมื่อทำการแสดงผลบนจอภาพ ส่วนของภาพจำลองที่ถูกกำหนดโดยหน้าต่างจะถูกนำมาคำนวณใหม่เพื่อเปลี่ยนแปลงขนาดและตำแหน่งให้เหมาะสมกับตำแหน่งและขนาดของช่องมองภาพโดยสมการทางคณิตศาสตร์ เช่น การขยายภาพหรือ การหมุนภาพ จากผลของการนำเรื่องหน้าต่างและช่องมองภาพมาใช้งานทำให้สามารถที่จะนำบางส่วนของภาพมาย่อหรือขยายตัด

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้เผยแพร่โดยไม่ขออนุญาต
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

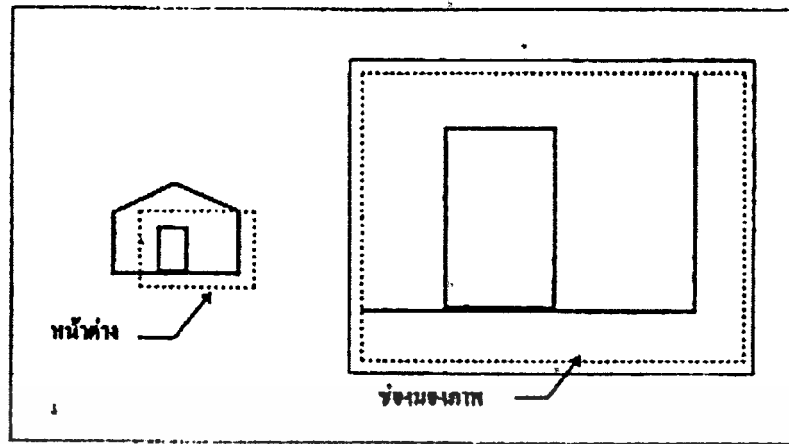
ส่วนได้ตามต้องการ การแสดงผลของช่องมองภาพ และหน้าต่างแสดงไว้ในรูปที่ 2.8 , รูปที่ 2.9 และรูปที่2.10



รูปที่ 2.8 การแสดงผลเมื่อมีหน้าต่างที่ต่างกันแต่ช่องมองภาพเดียวกัน



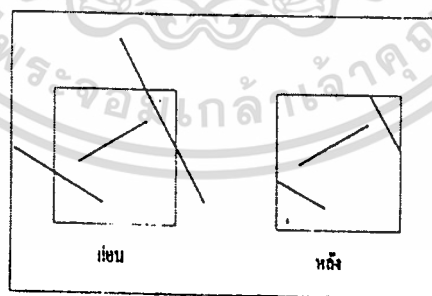
รูปที่ 2.9 การแสดงผลเมื่อมีหน้าต่างเดียวกันแต่ช่องมองภาพต่างกัน



รูปที่ 2.10 การขยายส่วนของภาพจำลองโดยช่องมองภาพ

2.1.6 การตัดส่วนของภาพ

การตัดส่วนของภาพ (Clipping) เป็นการกำจัดส่วนของภาพจำลองที่ไม่ต้องการหรือส่วนของภาพจำลองที่อยู่ภายนอกหน้าต่างที่กำหนด ดังนั้นในการตัดส่วนของภาพจำลองนั้นจะต้องพิจารณาว่าส่วนใดส่วนหนึ่งของภาพจำลองมีความสัมพันธ์อย่างไรกับขนาดและตำแหน่งของหน้าต่าง เช่น ภาพจำลองอยู่ภายในหน้าต่างทั้งหมด ภาพจำลองอยู่ภายนอกหน้าต่างทั้งหมดหรือบางส่วนของภาพจำลองอยู่ภายนอกหน้าต่าง รูปที่ 2.11 แสดงการตัดส่วนของภาพจำลอง ในการตัดส่วนของภาพมีวิธีการทำได้หลายวิธีด้วยกัน เช่น วิธีของ Cohen - Sutherland และวิธีของ Sutherland - Hodgman



รูปที่ 2.11 การตัดส่วนของภาพ

2.1.6.1 อัลกอริทึมของ Cohen - Sutherland

วิธีการตัดส่วนของเส้นตรงที่นิยมให้กันมากคือ วิธีของ Cohen - Sutherland ซึ่งวิธีการตัดส่วนของเส้นตรงวิธีนี้สามารถทำได้รวดเร็วมาก เมื่อเส้นตรงที่ต้องการตัดอยู่เฉพาะด้านใดด้านหนึ่งของเอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

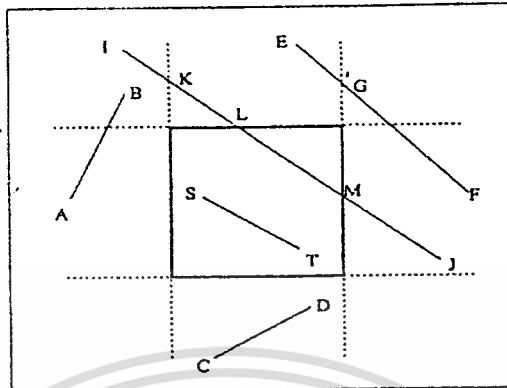
ของบริเวณที่ต้องการตัด วิธีการตัดส่วนของเส้นตรงวิธีนี้อาศัยวิธีการของตัวดำเนินการแบบบิต หรือ bit operator หรือ outcode มาใช้ โดยกำหนดจุดจบ (endpoint) ของเส้นตรงมีค่า 4 ค่าด้วยกัน โดยแทนที่ด้วยเลขฐาน 2 บิตที่มีลำดับสูงสุดจะถูกกำหนดให้มีค่าเป็น 1 เมื่อจุดนั้นอยู่นอหน้าต่าง บิตลำดับถัดมาจะถูกกำหนดให้มีค่าเป็น 1 เมื่อจุดนั้นอยู่ต่ำกว่าหน้าต่างบิตลำดับที่ 3 และ 4 จะถูกกำหนดให้มีค่าเป็น 1 เมื่อจุดนั้นอยู่ด้านขวาและด้านซ้ายของหน้าต่างตามลำดับดังรูปที่ 2.12

1001	1000	1010
0001	0000	0010
0101	0100	0110

รูป 2.12 แสดงค่าของตำแหน่งของจุดต่าง ๆ รอบหน้าต่าง

จากรูปที่ 2.12 เมื่อเส้นตรงอยู่ในหน้าต่างทั้งเส้น จุดปลายทั้งสองเส้นตรงจะมีค่า outcode เป็น “0000” เช่น เส้นตรง ST ในรูปที่ 2.13 หรือในกรณีที่เส้นตรงเส้นนั้นอยู่นอกหน้าต่างด้านใดด้านหนึ่ง ดังนั้นที่จุดปลายทั้งสองของเส้นตรงจะมีค่า outcode เป็น 1 ในตำแหน่งที่กำหนดของหน้าต่างด้านนั้น การตรวจสอบว่าเส้นตรงเส้นใดอยู่นอกหน้าต่างสามารถกระทำได้โดยใช้วิธีการทางตรรก (logic) เข้ามาช่วยวิธีการนั้นคือ การนำเอาผลของการใช้ตัวดำเนินการ “AND” มาให้กับตำแหน่งของจุดปลายของเส้นตรงนั้น ถ้าผลของการ “AND” มีค่าไม่เท่ากับศูนย์ แสดงว่า เส้นตรงเส้นนั้นอยู่นอกหน้าต่าง เช่น เส้นตรง AB และ CD ในรูปที่ 2.13

โดยวิธีการนี้ส่วนที่มีความยุ่งยากมากที่สุดในการหา คือ ในกรณีที่เส้นตรงนั้นมีบางส่วนอยู่ในหน้าต่าง เช่น เส้นตรง EF และ IJ ในรูปที่ 2.13 สำหรับในกรณีนี้จุดตัดของเส้นตรงและขอบของหน้าต่างจะถูกนำมาใช้ช่วยในการแยกเส้นตรงเส้นเดิมออกให้เป็นเส้นตรงเส้นใหม่ ส่วนของเส้นตรงใหม่จะถูกนำมาทดสอบโดยวิธีการเดิมเพื่อตรวจสอบว่าเส้นตรงเส้นใหม่อยู่นอกหน้าต่างหรือไม่ เช่น เส้นตรง EF จะถูกแยกออกเป็นเส้นตรง EG และ GF และพบว่าเส้นตรงทั้งสองอยู่นอกหน้าต่าง แต่เส้นตรง IJ จะถูกแบ่งออกเป็น IK, KL, LM และ MJ ซึ่งจะทำให้ได้ส่วนของเส้นตรง LM อยู่ในหน้าต่างที่ต้องการ

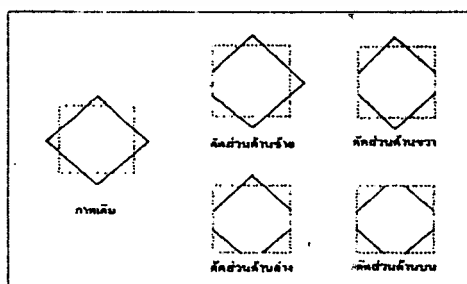


รูปที่ 2.13 แสดงลักษณะเส้นตรงต่าง ๆ รอบหน้าต่าง

2.1.6.2 วิธีของ Sutherland - Hodgman

วิธีการของ Cohen - Sutherland นั้นสามารถนำมาใช้ได้กับการตัดส่วนของเส้นตรง แต่ในบางกรณีภาพต่าง ๆ ที่ได้ไม่ใช่ภาพของเส้นตรงแต่เป็นภาพของรูปหลายเหลี่ยม วิธีการของ Sutherland - Hodgman เป็นวิธีการตัดส่วนของภาพจากเส้นรอบรูปของหน้าต่างทั้ง 4 ด้าน โดยไม่ขึ้นต่อกัน หลักการของวิธีการนี้คือ สามารถที่จะทำให้ตัดภาพจากด้านใดด้านหนึ่งของหน้าต่าง โดยไม่ขึ้นกับด้านอื่น ๆ

วิธีการตัดส่วนของภาพที่บริเวณขอบเขตของหน้าต่างทำได้โดยพิจารณาที่จุดปลายของเส้นตรงทุกจุดว่าจุดนั้นอยู่ในหน้าต่างหรืออยู่ที่ขอบเขตของหน้าต่างหรือไม่ ถ้าอยู่ในหน้าต่างที่ต้องการ จุดนั้นจะถูกรักษาไว้หรือคำนวณใหม่ ในกรณีที่จุดอยู่ที่บริเวณจุดตัดของเส้นตรงและขอบเขตของหน้าต่าง ดังนั้นเมื่อได้จุดทั้งหมดที่อยู่ในหน้าต่างแล้วจะทำให้ได้ส่วนของภาพที่ในบริเวณของหน้าต่างที่ต้องการ ดังรูปที่ 2.14



รูปที่ 2.14 การตัดส่วนของภาพที่บริเวณขอบของหน้าต่าง

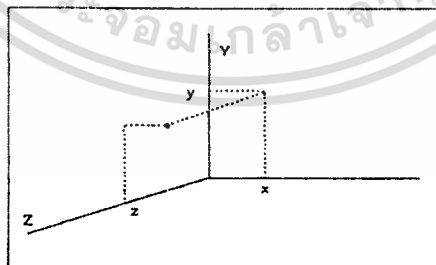
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.2 กราฟฟิก 3 มิติ

ภาพกราฟิกบางชนิดจำเป็นต้องใช้ระบบ 2 มิติเพื่อแสดงผลเช่น ภาพของกราฟต่าง ๆ แผนที่ หรือแม้กระทั่งภาพที่ถูกสร้างจากจิตรกรต่าง ๆ ก็เป็นภาพกราฟิกในระบบ 2 มิติ แต่ในงานบางอย่างก็จำเป็นต้องใช้ภาพในระบบ 3 มิติในการแสดงผลเพื่อให้เกิดความง่ายในการที่จะทำให้เข้าใจในภาพนั้น เช่น ภาพของโครงสร้างต่าง ๆ ซึ่งถ้าเป็นภาพจำลองที่สร้างขึ้นมาในระบบ 3 มิติจะสามารถทำให้ผู้ใช้สามารถมองเห็นภาพรวมในจุดมอง (viewpoint) ที่ต่าง ๆ กัน

2.2.1 จุดและระนาบ

สิ่งที่ง่ายที่สุดในการที่จะนำมาใช้ในการอธิบายระบบ 3 มิติคือ จุด เนื่องจากในระบบ 2 มิติได้อธิบายจุดโดยที่บอกตำแหน่งของจุดโดยอาศัย ค่า 2 ค่า แต่ในระบบ 3 มิตินี้จะต้องเพิ่มแกนอีก 1 แกนเพื่อที่จะสามารถนำมาใช้บอกตำแหน่งของจุดในระบบ 3 มิติ ซึ่งจะทำให้ตัวเลขที่ใช้ในการบอกตำแหน่งของจุดมี 3 ค่าด้วยกัน คือ ค่าแรกจะแทนค่าความสูง (height) ของจุด ส่วนค่าที่ 2 และ 3 จะแทนค่าความกว้าง (width) และความลึก (depth) ตามลำดับ โดยที่ทั้ง 3 แกนที่ใช้ในการอธิบายตำแหน่งในระบบ 3 มิติ จะมีทิศทางที่ตั้งฉากซึ่งกันและกัน ในระบบ 2 มิตินั้นใช้ค่าในแนวแกน X และแกน Y แทนค่าตำแหน่งในด้านความกว้างและความยาวส่วนในระบบ 3 มิตินั้นจะใช้แนวแกน Z แทนค่าของตำแหน่งในด้านความลึก ส่วนในแนวแกน X และแกน Y นั้นเหมือนกับในระบบ 2 มิติทุกประการ ดังนั้นการกำหนดตำแหน่งของจุดในระบบ 3 มิติจะสามารถกำหนดได้โดยตัวแปร 3 ตัวคือ $P(x, y, z)$ ดังแสดงในรูปที่ 2.15



รูปที่ 2.15 แสดงตำแหน่งของจุดในระบบ 3 มิติ

และสมการของเส้นตรงในระบบ 3 มิติคือ

$$\begin{aligned}\frac{y-y_1}{x-x_1} &= \frac{y_2-y_1}{x_2-x_1} \\ \frac{z-z_1}{x-x_1} &= \frac{z_2-z_1}{x_2-x_1}\end{aligned}\quad (11)$$

ส่วนสมการของระนาบในระบบ 3 มิติคือ

$$Ax + By + Cz + D = 0 \quad (12)$$

เมื่อ A, B, C และ D คือค่าคงที่หรือ

$$x + B_1y + C_1z + D_1 = 0 \quad (13)$$

โดยที่ $B_1 = \frac{B}{A}$, $C_1 = \frac{C}{A}$, และ $D_1 = \frac{D}{A}$

และระยะทางระหว่างจุด (x, y, z) ใด ๆ กับระนาบถูกกำหนดโดย

$$L = \left| \frac{A_2x + B_2y + C_2z + D_2}{d} \right| \quad (14)$$

โดยที่ $A_2 = \frac{A}{d}$, $B_2 = \frac{B}{d}$, $C_2 = \frac{C}{d}$ และ $D_2 = \frac{D}{d}$

2.2.2 การเปลี่ยนตำแหน่งในระบบ 3 มิติ

ในระบบพิกัดแบบ homogeneous เมตริกซ์ขนาด 4×4 ของเมตริกซ์การเปลี่ยนแปลงขนาดคือ

$$S = \begin{vmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \quad (15)$$

เมื่อ S คือ สัดส่วนการขยายในแนวแกน X, Y และ Z ตามลำดับ

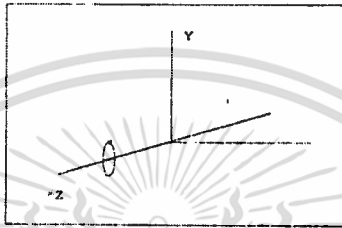
สำหรับการเคลื่อนย้ายตำแหน่งนั้น เมตริกซ์ของการเปลี่ยนแปลงตำแหน่งแกน x ไป t

หน่วย และเคลื่อนย้ายตำแหน่งไป และ ในแกน y และ z ตามลำดับ คือ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t_x & t_y & t_z & 1 \end{bmatrix} \quad (16)$$



รูปที่ 2.16 การหมุนรอบแกน Z

ส่วนการหมุนภาพรอบแกนต่าง ๆ สามารถแสดงเมตริกซ์ของการเปลี่ยนแปลงของการหมุนได้ดังนี้
สำหรับการหมุนรอบแกน Z ด้วยมุม θ ในทิศทางเข็มนาฬิกาแสดงในรูปที่ 2.16 คือ

$$R_z = \begin{bmatrix} \cos \theta & \sin \theta & 0 & 0 \\ -\sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (17)$$

และเมตริกซ์ของการเปลี่ยนแปลงของการหมุนรอบแกน X แกน Y คือ ตามลำดับ

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta & \sin \theta & 0 \\ 0 & -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (18)$$

$$R_y = \begin{vmatrix} \cos \theta & 0 & -\sin \theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin \theta & 0 & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \quad (19)$$

2.2.3 การหมุนวัตถุรอบแกนใด ๆ

การหมุนวัตถุรอบแกนใด ๆ สามารถกระทำได้โดยอาศัยเมตริกซ์ของการเปลี่ยนแปลงตำแหน่งต่าง ๆ เช่น เมตริกซ์ของการเคลื่อนย้ายตำแหน่งและเมตริกซ์ของการหมุนรอบแกนใด ๆ โดยทั่วไปแล้วเส้นตรงต่าง ๆ ในระบบ 3 มิติ สามารถถือได้ว่าเป็นแกนของระบบได้ ดังนั้นการหมุนวัตถุรอบแกนใด ๆ เป็นมุม θ นั้น สามารถหาเมตริกซ์ของการเปลี่ยนแปลงได้โดยในขั้นแรกนั้นต้องเคลื่อนย้ายแกนใด ๆ ให้จุดกำเนิดของระบบมาอยู่บนแกนที่ต้องการ จากนั้นจะต้องหมุนวัตถุรอบแกน X และ แกน Y เพื่อให้แกนที่ต้องการอยู่บนแกน Z จากนั้นจึงทำการหมุนวัตถุรอบแกน Z ตามมุมที่ต้องการ ในขั้นตอนสุดท้ายจะต้องทำการเลื่อนแกนที่ต้องการกลับไปสู่ตำแหน่งเดิมในตอนแรกโดยอาศัยส่วนกลับ (inverse) ของเมตริกซ์ของการเปลี่ยนแปลงที่ใช้ในครั้งแรก คือการหมุนรอบแกน Y และแกน X ตามลำดับ จากนั้นจึงทำการเลื่อนแกนของวัตถุกลับสู่ตำแหน่งเดิม รูปที่ 2.23 แสดงขั้นตอนการหมุนวัตถุรอบแกนใด ๆ และเมตริกซ์ของการหมุนวัตถุรอบแกนใด ๆ เป็นมุม θ คือ

$$R_\theta = TR_X R_Y R_Z R_Y^{-1} R_X^{-1} T^{-1} \quad (20)$$

เมื่อ T = เมตริกซ์ของการเปลี่ยนแปลงตำแหน่ง

R_X = เมตริกซ์ของการเปลี่ยนแปลงการหมุนรอบแกน X

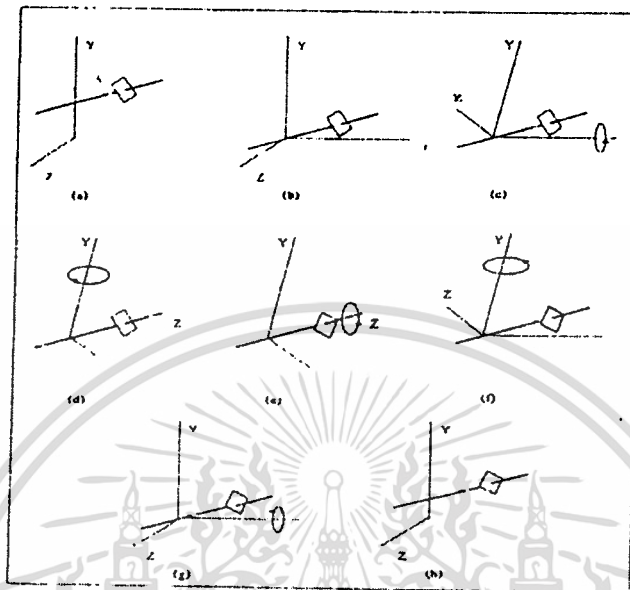
R_Y = เมตริกซ์ของการเปลี่ยนแปลงการหมุนรอบแกน Y

R_Z = เมตริกซ์ของการเปลี่ยนแปลงการหมุนรอบแกน Z

T^{-1} = ส่วนกลับของเมตริกซ์ของการเปลี่ยนแปลงตำแหน่ง

R_X^{-1} = ส่วนกลับของเมตริกซ์ของการเปลี่ยนแปลงการหมุนรอบแกน X

R_Y^{-1} = ส่วนกลับของเมตริกซ์ของการเปลี่ยนแปลงการหมุนรอบแกน Y



รูปที่ 2.17 การหมุนวัตถุรอบแกนใด ๆ

2.2.4 การฉายภาพแบบขนานและการฉายภาพแบบเพอสเปคตีฟ

วัตถุต่าง ๆ ในระบบ 3 มิติ นั้นประกอบด้วยคุณสมบัติของความกว้าง, ความยาวและความลึกซึ่งต่างจากคุณสมบัติของวัตถุระบบ 2 มิติ ซึ่งมีเพียงความกว้างและความยาว อุปกรณ์การแสดงผลของภาพต่าง ๆ นั้นจึงถือว่าเป็นอุปกรณ์ในระบบ 2 มิติ เพราะสามารถที่จะกำหนดค่าได้เพียงความกว้างและความยาวเท่านั้น ดังนั้นการที่จะนำภาพในระบบ 3 มิติ มาแสดงผลบนอุปกรณ์แสดงผลหรือจอภาพจึงต้องใช้วิธีการบางอย่างเข้ามาช่วยในการแสดงผลเพื่อให้สามารถแสดงผลของความลึกของวัตถุวิธีการนี้เรียกว่า การฉายภาพ(projection)

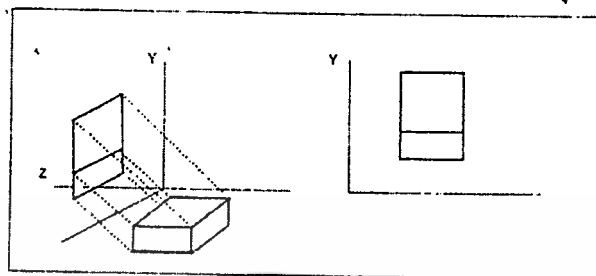
การฉายภาพของวัตถุในระบบ 3 มิติลงบนจอภาพสามารถที่จะแยกออกได้เป็น 2 วิธีด้วยกัน คือ

1. การฉายภาพแบบขนาน (Parallel Projection)
2. การฉายภาพแบบเพอสเปคตีฟ (Perspective Projection)

2.2.4.1 การฉายภาพแบบขนาน

วิธีการฉายภาพแบบขนานสามารถกระทำได้โดยการลากเส้นสมมุติจากจุดยอดของวัตถุทุก ๆ จุดมาตัดกับระนาบของการมองหรือระนาบของจอภาพ จุดตัดของเส้นขนานจากจุดยอดของวัตถุและระนาบของจอภาพคือ จุดของการฉายภาพของจุดยอดของวัตถุที่ระนาบของจอภาพ ดังนั้นถ้า

ลากเส้นเชื่อมจุดของการฉายภาพเหล่านี้ จะได้ภาพของการฉายภาพแบบขนานของวัตถุ ดังแสดงในรูปที่ 2.18



รูปที่ 2.18 การฉายภาพแบบขนาน

จากวิธีการฉายภาพแบบขนานนี้จะเห็นว่าความลึกของวัตถุในแนวแกน Z จะไม่มีผลกับภาพที่ได้ ในกรณีที่ระนาบของการมองอยู่ในระนาบ XY เนื่องจากเส้นของการฉายภาพจะขนานกับแกน Z ดังนั้น ถ้าเคลื่อนที่ภาพในแนวของเส้นของการฉายภาพ ค่าที่มีการเปลี่ยนแปลงคือค่าของวัตถุในแนวแกน Z เท่านั้น แต่ค่าของวัตถุในแนวแกน X และแกน Y ยังคงมีค่าเท่าเดิม ซึ่งทำให้จุดตัดของการฉายภาพที่ระนาบของการมองมีค่า x และ y เท่ากับค่าของ x และ y ของจุดยอดของวัตถุ ดังนั้นภาพที่ถูกฉายที่ระนาบของการมองจะถูกสร้างจากค่าของ x และ y เท่านั้น โดยที่ค่าในแนวแกน Z จะไม่มีผลใดๆ กับภาพในการฉายภาพแบบขนานนั้นจะต้องกำหนดทิศทางของการฉายภาพ เช่น $[x,y,z]$ และการฉายภาพจะเกิดที่ระนาบ xy ถ้าจุด (x,y,z) เป็นจุดใดๆ บนวัตถุ สามารถจะทราบตำแหน่งที่จะถูกฉาย (x,y) ได้โดย

สมการเส้นตรงในรูปสัมการพาราเมตริกซ์ที่มีทิศทางของการฉายภาพและผ่านจุด (x, y, z) ใดๆ คือ

$$\begin{aligned} X &= X_1 + X_p U \\ Y &= Y_1 + Y_p U \\ Z &= Z_1 + Z_p U \end{aligned} \quad (21)$$

ที่จุดตัดของทิศทางของการฉายภาพ และระนาบ XY คือ $z = 0$ ดังนั้น



$$u = \frac{z_p}{z_p}$$

(22)

ดังนั้น

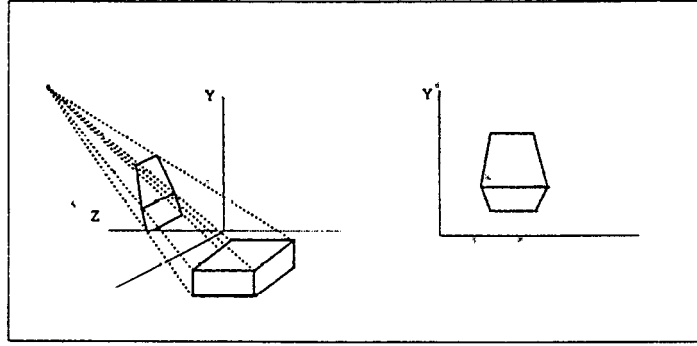
$$\begin{aligned} x_2 &= x_1 - z_1 \frac{x_p}{z_p} \\ y_2 &= y_1 - z_1 \frac{y_p}{z_p} \end{aligned} \quad (23)$$

สามารถเขียนในรูปของเมทริกซ์ในระบบพิกัดแบบ homogeneous ได้คือ

$$[x_2 y_2 z_2 1] = [x_1 y_1 z_1 1] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{x_p}{z_p} & -\frac{y_p}{z_p} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (24)$$

2.2.4.2 การฉายภาพแบบเพอสเปคตีป

การฉายภาพวัตถุแบบเพอสเปคตีปนี้ ภาพของวัตถุที่ได้จะมีขนาดขึ้นอยู่กับตำแหน่งของผู้สังเกต ถ้าผู้สังเกตอยู่ไกลภาพที่ได้จะมีขนาดเล็กและภาพของวัตถุจะมีใหญ่ขึ้นเมื่อผู้สังเกตอยู่ใกล้กับวัตถุ ดังนั้นการฉายภาพแบบนี้จะสามารถทำให้ภาพที่ได้มีขนาดต่าง ๆ กันขึ้นอยู่กับระยะห่างของผู้สังเกตและวัตถุ ในการฉายภาพแบบเพอสเปคตีปนี้เส้นของการฉายภาพ (lines of projection) จะไม่ใช่เส้นที่ขนานกัน แต่จะรวมกันที่จุด ๆ หนึ่งเรียกว่า จุดศูนย์กลางของการฉายภาพ (center of projection) จุดตัดของเส้นของการฉายภาพและระนาบของการมองเห็นเหมือนกับภาพที่ผู้สังเกตมองวัตถุที่ตำแหน่งจุดศูนย์กลางของการฉายภาพหรือทิศทางของเส้นของการฉายภาพจะมีลักษณะเหมือนกับทิศทางของแสงที่มาจากวัตถุมาสู่ดวงตา รูปที่ 2.19 แสดงวิธีการฉายภาพแบบเพอสเปคตีป



รูปที่ 2.19 การฉายภาพแบบเพอสเปคตีฟ

ถ้ากำหนดจุดศูนย์กลางของการฉายภาพอยู่ที่ตำแหน่ง (x_c, y_c, z_c) และจุดบนวัตถุอยู่ที่ตำแหน่ง (x, y, z) ดังนั้นทิศทางของการฉายภาพคือ เส้นตรง

$$\left. \begin{aligned} x &= x_c + (x_1 - x_c)u \\ y &= y_c + (y_1 - y_c)u \\ z &= z_c + (z_1 - z_c)u \end{aligned} \right\} \quad (25)$$

ตำแหน่งของจุด (x, y) ที่ถูกฉายบนระนาบการมอง XY เมื่อ $z=0$ คือ

$$\left. \begin{aligned} x_2 &= \frac{x_c z_1 - x_1 z_c}{z_1 - z_c} \\ y_2 &= \frac{y_c z_1 - y_1 z_c}{z_1 - z_c} \end{aligned} \right\} \quad (26)$$

สามารถเขียนการฉายภาพแบบเพอสเปคตีฟในรูปของการเปลี่ยนแปลงในระบบของ homogeneous coordinate คือ

$$P = \begin{vmatrix} -z_c & 0 & 0 & 0 \\ 0 & -z_c & 0 & 0 \\ x_c & y_c & -1 & 1 \\ 0 & 0 & 0 & -z_c \end{vmatrix} \quad (27)$$

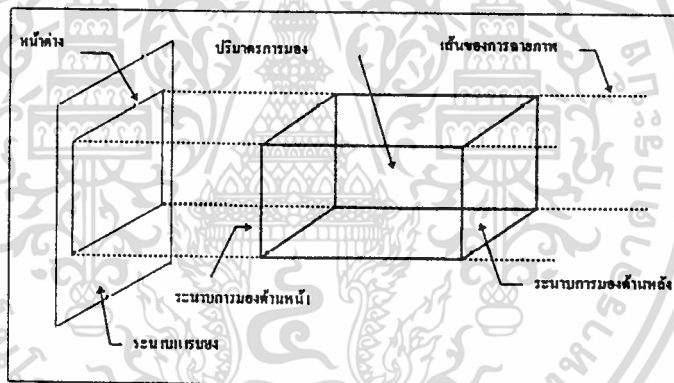
2.2.5 การตัดส่วนของภาพในระบบ 3 มิติ

ในระบบ 2 มิติหน้าต่างถูกใช้เป็นขอบเขตของการมองภาพ เช่นเดียวกันในระบบ 3 มิติ

หลักการของหน้าต่างยังถูกนำมาใช้กำหนดคปริมาตรของการมอง (view volume) วัตถุที่อยู่ภายนอกเอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

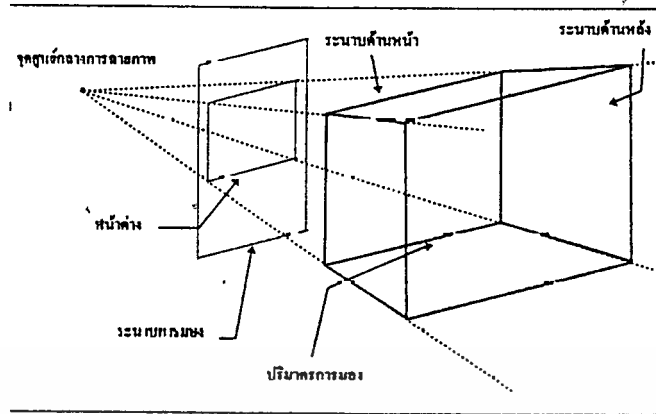
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บริเวณปริมาตรของการมองเห็นจะถูกตัดออก ดังนั้นส่วนต่าง ๆ ของวัตถุที่อยู่ภายในปริมาตรของการมองเห็นนั้นที่จะถูกมองเห็น เนื่องจากเป็นระบบ 3 มิติ ปริมาตรของการมองเห็นจึงถูกกำหนดโดยระนาบของการมองเห็น 6 ระนาบด้วยกัน คือ ด้านหน้า, ด้านหลังและด้านข้าง 4 ด้าน หลักการของการตัดส่วนของภาพในระบบ 3 มิติ ยังคงเหมือนกับการตัดส่วนของภาพในระบบ 2 มิติคือ จะต้องตรวจสอบว่าตำแหน่งของวัตถุอยู่ในปริมาตรของการมองเห็นหรือไม่ แต่ในระบบ 3 มิติจะต้องตรวจสอบว่าจุดต่าง ๆ อยู่ภายในระนาบหรือไม่ แทนการตรวจสอบว่าวัตถุอยู่บนเส้นตรงเหมือนกับในระบบ 2 มิติ ปริมาตรของการมองเห็นจะถูกแยกออกเป็น 2 แบบด้วยกันตามชนิดของการฉายภาพสำหรับการฉายภาพแบบขนาน ระนาบที่อยู่ในทิศเดียวกับทิศทางการฉายภาพ จะกำหนดทอรูปเหลี่ยมที่มีขนาดเท่ากับขนาดของหน้าต่าง ดังรูปที่ 2.20 และระนาบของการตัดภาพของด้านหน้าและด้านหลังจะถูกกำหนดขึ้นมาซึ่งจะทำให้ทอรูปเหลี่ยมเปลี่ยนให้มีลักษณะเหมือนกล่องรูปลูกบาศก์ วัตถุต่าง ๆ ที่อยู่ภายในบริเวณกล่องรูปลูกบาศก์เท่านั้นที่จะสามารถมองเห็นได้



รูปที่ 2.20 ปริมาตรการมองเห็นสำหรับการฉายภาพแบบขนาน

สำหรับการฉายภาพแบบเพอสเปกตีฟนั้น ขอบเขตของหน้าต่างที่ระนาบการมองเห็นและจุดศูนย์กลางของการฉายภาพ จะทำให้เกิดลักษณะที่เป็นพีระมิด (Pyramid) ขึ้น ดังรูปที่ 2.20 รูปพีระมิดจะถูกแบ่งโดยระนาบของการตัดด้านหน้าและระนาบของการตัดด้านหลัง จะทำให้เกิดปริมาตรของการมองเห็นจากสมการที่ 1.12 ซึ่งเป็นสมการของระนาบใด ๆ ทำให้สามารถนำไปประยุกต์ในการทดสอบว่าจุดต่าง ๆ อยู่บนระนาบหรือไม่หรืออยู่ด้านใดของระนาบที่กำหนด ซึ่งจะทำให้สามารถนำไปคำนวณหาภาพที่อยู่ในบริเวณของปริมาตรของการมองเห็นได้



รูปที่ 2.21 ปริมาตรการมองสำหรับการฉายภาพแบบเพอสเปคตีฟ

2.2.6 การซ่อนเส้นและพื้นผิว

ในการมองภาพของวัตถุที่มีความเหมือนจริง รายละเอียดของวัตถุจะไม่สามารถมองเห็นได้ทั้งหมดภายในครั้งเดียว เนื่องจากการบดบังซึ่งกันและกัน ส่วนประกอบส่วนใดของวัตถุที่สามารถมองเห็นได้และส่วนใดของวัตถุจะถูกบดบังไว้ จะขึ้นอยู่กับตำแหน่งที่มองภาพเช่น จากการมองจากด้านหน้า ส่วนของวัตถุที่สามารถมองเห็นได้คือบริเวณด้านหน้า และส่วนด้านหลังของวัตถุจะเป็นส่วนที่ถูกบดบังไว้ ในทางตรงกันข้ามถ้ามองวัตถุจากด้านหลังส่วนของวัตถุที่ถูกบดบังไว้คือด้านหน้าของวัตถุ หรือในกรณีที่มีมองวัตถุจากภายนอกจะทำให้ไม่สามารถที่จะมองเห็นรายละเอียดภายในของวัตถุได้เลย เนื่องจากถูกบดบังด้วยพื้นผิวของวัตถุภายนอกจากการสร้างแบบจำลองของวัตถุในระบบ 3 มิติ นั้น จะต้องสร้างและกำหนดรายละเอียดทุกจุดของภาพทั้งรายละเอียดภายในหรือภายนอก ดังนั้นในการแสดงภาพวัตถุแต่ละครั้งจะทำให้รายละเอียดที่กำหนดไว้แสดงออกมาด้วยซึ่งจะทำให้เป็นการยากที่จะกำหนดว่ารายละเอียดส่วนใดเป็นของบริเวณใดของวัตถุ วิธีการที่ช่วยในการแสดงผลของวัตถุให้สามารถแยกรายละเอียดของพื้นผิวของวัตถุได้เรียกว่า การซ่อนเส้นและพื้นผิว (Hidden lines and Surface) การซ่อนเส้นและพื้นผิวสามารถกระทำได้หลายวิธีด้วยกัน ซึ่งแต่ละวิธีนั้นมีขั้นตอนการทำงานที่ต่างกัน

2.1.6.1 การใช้ Z BUFFERS

การซ่อนเส้นและพื้นผิวโดยวิธีการใช้ Z Buffers อาศัยคุณสมบัติของอุปกรณ์ต่าง ๆ มาช่วยในการทำงานคือ จอภาพ เนื่องจากจอภาพที่เรียกว่า Raster display นั้นจะอาศัย frame buffer เป็นตัวช่วยในการแสดงผลคือ ค่าของจุดต่าง ๆ ที่จะถูกนำมาแสดงผลจะถูกนำเก็บไว้ในหน่วยความจำที่ใช้เฉพาะกับจอภาพก่อนแล้วจึงแสดงผลทางจอภาพ Z Buffers หมายถึง ชุดของหน่วยความจำที่กำหนดขึ้นมาเพื่อใช้ในการเก็บข้อมูลของทุก ๆ จุดบนจอภาพ คล้ายกับ frame buffer หน่วย

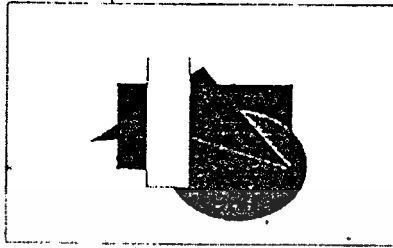
ความจำนี้จะเก็บข้อมูลของตำแหน่งในแกน Z ของพื้นผิวที่จะถูกแสดงผล ภาพหรือพื้นผิวต่าง ๆ จะถูกเก็บเข้าสู่หน่วยความจำของ frame buffer ตามลำดับ จากนั้นตำแหน่งต่าง ๆ ของภาพหรือพื้นผิวจะถูกฉายลงบนส่วนการแสดงผล ดังนั้นจึงสามารถที่จะเปรียบเทียบค่าของตำแหน่งในแกน Z ของพื้นผิวกับค่าอ้างอิงที่เก็บใน Z Buffer จะทำให้ทราบว่าพื้นผิวที่ถูกฉายนั้นอยู่ที่ตำแหน่งใดเมื่อเทียบกับพื้นผิวเดิม โดยที่ถ้าพื้นผิวใหม่ที่มีค่า Z มากกว่าค่าที่เก็บใน Z Buffer แสดงว่าพื้นผิวนั้นอยู่ที่บริเวณด้านหน้าของพื้นผิวเดิมและค่าความเข้มของจุดนั้นจะถูกนำไปเก็บใน Z Buffer จะหมายถึงว่าพื้นผิวใหม่อยู่ที่บริเวณด้านหลังของพื้นผิวที่เก็บค่าใน Z Buffer ซึ่งพื้นผิวใหม่จะเป็นพื้นผิวที่ถูกบังไว้จากวิธีการดังที่กล่าวมาของ Z Buffer จะเห็นว่าวิธีการนี้ใช้เวลามาก เนื่องการใช้จุดทุกจุดในการคำนวณ และต้องใช้หน่วยความจำเป็นจำนวนมากเพื่อใช้เก็บข้อมูล แต่วิธีนี้เป็นวิธีที่ค่อนข้างง่าย ดังนั้นจึงสามารถนำไปพัฒนาในส่วนของ Hardware เพื่อให้มีความเร็วเพิ่มขึ้นในการประมวลผลและในปัจจุบันนี้ค่าใช้จ่ายในการผลิต Hardware ยังมีราคาไม่สูงมากด้วยอีกประการหนึ่ง

2.2.6.2 อัลกอริทึม SCAN LINE

วิธีการซ่อนเส้นและพื้นผิวในวิธีของอัลกอริทึม Scan line นั้นจะคล้ายวิธีการเปรียบเทียบความลึกของพื้นผิวทั้งหมดในครั้งเดียวที่เส้น Scan line หนึ่ง เมื่อได้พื้นผิวที่อยู่ด้านหน้าสุดแล้ว จึงจะทำการหาพื้นผิวที่อยู่หน้าสุดที่เส้น Scan line อื่น ๆ ส่วนหลักการทั่วไปยังคงเหมือนกับหลักการของ Z-Buffer จากวิธีการนี้จะเห็นว่าหน่วยความจำบางส่วนที่ใช้ในการเก็บค่าความลึกของพื้นผิวจะน้อยลงเพราะไม่ต้องการเก็บข้อมูลทุกครั้ง แต่จะเก็บข้อมูลเฉพาะในส่วนของ Scan line เท่านั้น เมื่อมีการเปลี่ยนแปลง Scan line เป็นเส้นใหม่ ค่าของ Z-Buffer จะถูกกำหนดใหม่ให้เป็นค่าเริ่มต้นแล้วจึงทำการคำนวณหาค่าใหม่ จึงเป็นการใช้หน่วยความจำน้อยกว่า

2.2.6.3 อัลกอริทึมของ PAINTER

วิธีการซ่อนเส้นและพื้นผิวโดยวิธีอัลกอริทึมของ PAINTER นี้อาศัยลำดับความลึกของพื้นผิวเป็นตัวกำหนดในการแสดงผล พื้นผิวที่มีความลึกที่สุดจะถูกแสดงก่อน ส่วนพื้นผิวที่อยู่ใกล้กับผู้สังเกตมากที่สุดจะถูกแสดงทีหลัง ดังแสดงในรูปที่ 2.22 จากหลักการที่ผ่านมามีการซ่อนพื้นผิวโดยวิธีนี้นั้นเหมือนกับการวาดภาพของนักเขียนภาพโดยที่จะเริ่มจากส่วนด้านหลัง (background) ก่อน จากนั้นจึงวาดส่วนด้านหน้าของภาพ (foreground) ซึ่งภาพของส่วนหน้าที่วาดจะเกิดการทับกับพื้นด้านหลัง ซึ่งทำให้ไม่มีการแก้ไขพื้นด้านหลังเลย แต่วิธีการนี้มีข้อเสียคือการกำหนดพื้นผิวต่าง ๆ ต้องเรียงลำดับความลึกด้วย หรือในกรณีที่มีการเรียงลำดับความลึกนั้นจะต้องทำการเปรียบเทียบค่าความลึกของพื้นผิวทุกครั้งที่มีการกำหนด



รูปที่ 2.22 อัลกอริทึมของ Painter

สรุป การที่จะแสดงผลของภาพจำลองที่สร้างขึ้นให้ได้ดีนั้นจำเป็นจะต้องนำสมการทางคณิตศาสตร์เข้ามาช่วยในการคำนวณต่าง ๆ ดังนั้นเครื่องคอมพิวเตอร์ที่มีคุณสมบัติที่ดีขึ้นด้านการประมวลผลและการแสดงผลจึงเป็นที่ต้องการมากขึ้น วิธีการอีกส่วนที่ทำให้ภาพจำลองมีคุณสมบัติดีขึ้นคือ การนำเรื่องสีและความเข้มของแสงเข้ามาช่วยในการจำลองภาพ ซึ่งจะได้กล่าวถึงรายละเอียดต่อไป

บทที่ 3

การแสดงผลวัตถุ

3.1 การแสดงและแบบจำลองของวัตถุพีชคณิต

มีการพัฒนารูปแบบการแสดงผลและแบบจำลองของวัตถุ 3 มิติในงาน computer graphics เป็นจำนวนมาก ในบางรูปแบบจะเหมาะสมกับงานเฉพาะด้าน เช่น CSG ซึ่งเป็น solid modeling ที่ใช้ในงานด้านวิศวกรรมที่จะอำนวยความสะดวกในการสร้างชิ้นงานจริง บางรูปแบบสร้างขึ้นมาเพื่อการ rendering การที่นิยม sphere ในงาน ray-traced images เพราะเป็นวัตถุที่ง่ายที่สุดในการทำ ray trace รูปแบบ patch ที่ใช้ใน radiosity method เพราะการใช้ method นี้ถ้าใช้ patch จะทำให้การหาผลเฉลยทำได้ดีขึ้น

องค์ประกอบที่รูปแบบจะต้องนิยามขึ้นมาคือ

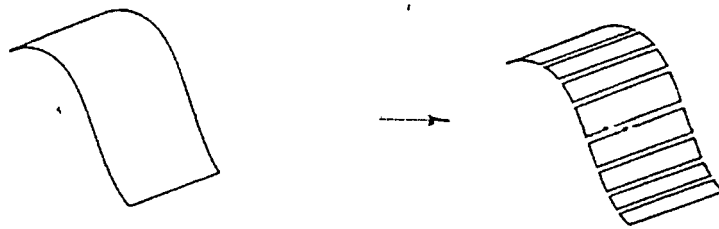
- โครงสร้างข้อมูล (data structure) ชนิดของ process algorithm รูปแบบ hardware ที่ใช้
- ค่าใช้จ่ายในการ process วัตถุในกระบวนการสร้าง 3 มิติ
- รูป 3 มิติที่ได้ เพราะบางแบบจะใกล้เคียงที่ต้องการมากกว่า
- ความง่ายหรือองค์ประกอบในการแก้ไขรูปร่างของวัตถุ

ตัวอย่างในการแสดงรูป 3 มิติ เฉพาะที่ใช้บ่อยและมีความสำคัญ

1. polygon คือ วัตถุที่ประมาณค่าโดยโครงข่ายของแผ่นระนาบ polygon
2. bicubic parametric patch คือวัตถุที่ถูกแทนด้วยโครงข่ายของ patch
3. constructive solid geometry (CSG) คือวัตถุที่ถูกแทนด้วยการรวมกันของทรงกลม ทรงกระบอก กล้อง
4. space subdivision tech คือการฝังวัตถุใน space ที่แต่ละจุดใน space มีการระบุค่าไว้อยู่แล้ว

3.2 polygonal representation

เป็นวิธีดั้งเดิม โดยจะแสดงวัตถุด้วยโครงข่ายของระนาบรูป polygon ในกรณีทั่วไปวัตถุจะถูกประมาณผิวโค้งด้วยระนาบดังในรูปที่ 3.1



รูปที่ 3.1 การประมาณค่าพื้นผิวโค้งโดยใช้รูปหลายเหลี่ยม

การแสดงด้วยโครงข่าย polygon จะเรียกเป็นทางการว่า boundary representation (B rep) เพราะเป็นคำจำกัดความของขอบเขตของผิววัตถุ

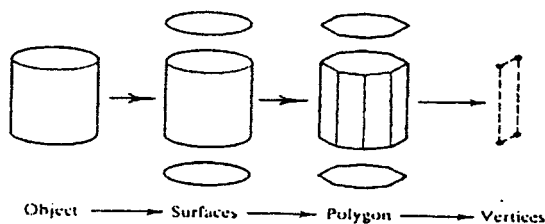
การแสดงผลแบบ polygon มีอยู่ทั่วไปในงาน computer graphics เพราะ 2 เหตุผล คือ การจำลองหรือการสร้างรูป polygon เป็นการทำให้ข้างหน้าถึงแม้ในทางปฏิบัติจะทำได้ยาก ความแม่นยำที่มีมาก คุณภาพของแบบที่สร้างขึ้นกับขนาดของแต่ละ polygon ว่าเหมาะกับรูปต้นแบบเพียงใด เช่น ถ้าโค้งมากก็จำเป็นต้องใช้ polygon ในอัตราที่เพิ่มขึ้น (จำนวน polygon / พื้นที่) ซึ่งปัจจัยนี้ก็คือ กระบวนการสร้างนั่นเอง เช่น การสร้าง 3 มิติด้วย digitizer หลังจากได้ coordinate แล้ว ตัวปฏิบัติการจะคำนวณขนาดของ polygon เพื่อให้พอดีกัน

การพัฒนาทางด้าน 3 D ที่สำคัญคือ การพัฒนา shading algorithm เป็นสิ่งที่เกี่ยวข้องกับวัตถุ polygon ในขณะเดียวกันก็เกี่ยวข้องกับ interpolation scheme ซึ่งทำให้ใช้ piece linearity ในการแสดงน้อยลง

โครงข่าย polygon ไม่ใช่เพียงเป็นโครงสร้างข้อมูลหุ่นจำลอง สามารถสร้างเป็นโครงสร้างข้อมูลแบบอื่นได้

ตัวอย่างโครงข่าย polygon ที่ง่ายคือ โครงสร้างข้อมูลที่ประกอบด้วยลำดับของพิกัด (x, y, z) ซึ่งแต่ละจุดคือ มุม vertex ของรูป polygon ซึ่งจะหมายถึงการอธิบายวัตถุด้วยลำดับของ vertex นอกจากนี้ยังสามารถเก็บรายละเอียดของวัตถุเป็นส่วนๆ หรือข้อมูลพีชคณิตอื่นๆที่ใช้ในการประมวลผล เช่น เวกเตอร์พวก polygon normal และ vertex normal ซึ่งพวกนี้จะคำนวณเพียงครั้งเดียวแต่จะมีการแปลงเชิงเส้นด้วยเมื่อวัตถุมีการแปลงเกิดขึ้น

เป็นการสะดวกที่จะจัดลำดับ polygon แบบสืบทอดเป็นลำดับชั้น ดังแสดงในรูปที่ 3.2



รูปที่ 3.2 ลำดับย่อยของการแสดงวัตถุ

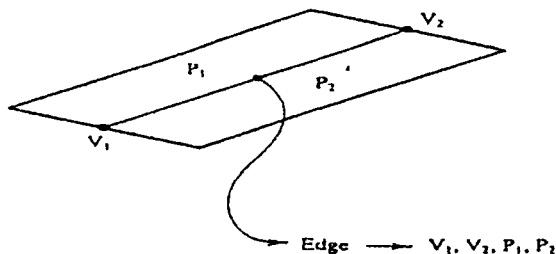
หลายๆ polygon รวมเป็นพื้นผิว แต่ละพื้นผิวรวมกันเป็นวัตถุ เช่น ทรงกระบอกมี 3 พื้นผิว คือ ฝาบน ฝาล่าง ซึ่งเป็นพื้นผิวแบบขอบโค้ง กับตัวผิวด้านข้าง ทั้งนี้เพื่อแยกความแตกต่างระหว่างขอบของวัตถุจริงกับขอบที่เกิดจากการประมาณค่าเส้นโค้งบนพื้นผิวเพราะการทำ render จะมีวิธีที่ต่างกัน

ลักษณะสำคัญอีกอย่างของการแสดงแบบ polygon คือความเป็นอิสระ ซึ่งหมายถึง วิธี render ที่นิยมต้องทำต่อ polygon แบบอิสระ ประมวลผลมันในหน่วยความจำโดยใช้กระบวนการของ Z buffer hidden surface removal ในการทำรวมกับการ shading

ในบางครั้งกล่าวกันว่า คำนิยามหลักของการแสดงรูปแบบนี้จะเป็นความสัมพันธ์ระหว่าง polygon ที่ถูกฝังตัวกันอยู่ เช่น ไม่ง่ายที่จะชี้ชัดว่า polygon อันใดใช้ vertex ร่วมกัน ซึ่งข้อมูลนี้จะใช้ใน shading algorithm ซึ่งความสำคัญของจุดนี้ขึ้นกับการนำรูปจำลองไปใช้

สำหรับการ render มาตรฐานส่วนใหญ่ รูปจำลองจะถูกกำหนดคดาบตัวในขั้นตอนการสร้างแล้ว จากนั้น จึงทำการแปลงเชิงเส้น

การแสดงวัตถุในรูป polygon อีกแบบคือ รวบรวมขอบที่ใช้ร่วมกัน (ไม่ใช่ขอบจริง) ดังในรูปที่ 3.3 ซึ่งผิวนบนสุดจะถูกแบ่งเป็น set ของขอบ ซึ่งแต่ละขอบจะประกอบไปด้วยรายการ 2 จุดและ 2 รูป polygon ที่ร่วมกันใช้ขอบนั้นๆ



รูปที่ 3.3 แสดงถึงการใช้ขอบร่วมกัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาก็ได้ แต่ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สิ่งกระตุ้นให้มีรูปแบบขอบเพราะประสิทธิภาพในขบวนการ render เมื่อทำการ render รูป polygon แล้ว สามารถทำขบวนการอื่นได้อีกที่แต่ละขอบ โดยเฉพาะการ clipping รูป polygon เพื่อแสดงใน view volume และการหา pixel ต่างๆที่เส้นขอบตัดผ่าน ถ้าแยกทำทีละรูป polygon ทำให้ขอบที่ใช้ร่วมกันต้องผ่านกระทำ 2 รอบ ซึ่งถือเป็นค่าใช้จ่ายรวมที่เพิ่มขึ้นซึ่งจะสำคัญมากหากรูปวัตถุนั้นประกอบไปด้วย polygon ที่มากและซับซ้อน



บทที่ 4

อัลกอริทึมของการเรนเดอร์

4.1 บทนำ

การกระทำการสร้างพื้นผิว (rendering) ตามขั้นตอนจำเป็นต้องใช้ลำดับขั้นตอน (algorithm) ที่ไม่ใช่พีชคณิต (non-geometric) ในที่นี้จะอ้างอิงถึงวัตถุที่แสดงด้วยโครงข่าย polygon

rendering คือ การ map ภาพ 3 มิติไปเป็นภาพ 2 มิติ ที่มีการไล่สีและ projection บน view surface โดยการ rendering ต้องทำตามขั้นตอนดังนี้

1. สร้างแบบจำลองวัตถุแบบ polygon ซึ่งมีข้อมูลที่ต้องใช้ในกระบวนการให้แสงเงา (shading)
2. ทำการแปลงเชิงเส้น (linear transformation) รูปจำลองโครงข่าย polygon เช่น การแปลงตำแหน่งต่างๆของวัตถุภายในหน้าจอ การแปลง viewing
3. ทำการซ่อนผิววัตถุที่ถูกบัง (culling)
4. ตัดรูป (clipping) polygon ที่อยู่ภายนอกส่วนที่มองเห็น (view volume)
5. ทำการ converting หรือ rasterizing รูป polygon ซึ่งจะเปลี่ยนแบบจำลองแบบจุดมุม (vertex)
6. ประยุกต์ลำดับขั้นตอน hidden surface removal
7. ให้แสงเงาแต่ละ pixel โดยใช้หลักการให้แสงเงาแบบ interpolative หรือ incremental

4.2 การเลือกและการตัด (culling and clipping)

คือ การซ่อนผิวด้านหลัง จะเปรียบเทียบรูป polygon รวม กับ view point หรือจุดศูนย์กลางการ projection จากนั้นทำการกำจัด polygon ข้อยที่มองไม่เห็น ถ้า polygon รูปเดียว การตรวจสอบว่าส่วนใดเป็นส่วนที่มองเห็นหรือเป็นส่วนที่มองไม่เห็น โดยการคำนวณดังนี้

$$\text{visibility} := N_p N > 0$$

เมื่อ N_p คือ polygon normal vector (พุ่งออกจากวัตถุ) และ

N คือ line of sight vector (จาก view point)

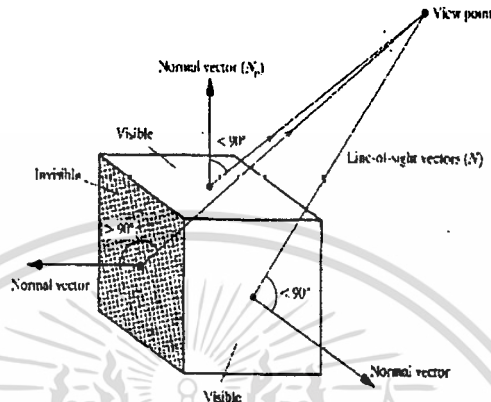
เรากำหนดขอบเขตของ coordinate ดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

$$-w \leq x \leq w$$

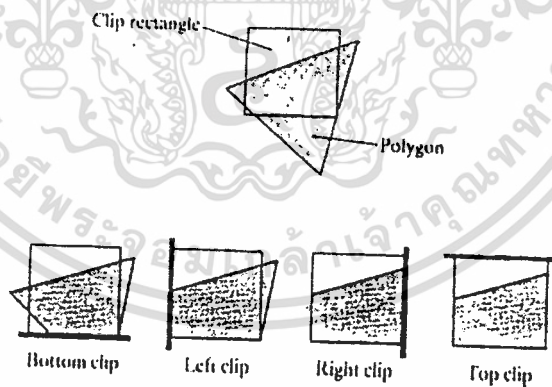
$$-w \leq y \leq w$$

$$0 \leq z \leq w$$



รูปที่ 4.1 การเลือกหรือการตัดผิวหลัง

object ที่ต้องการทดสอบว่าอยู่ภายในขอบเขตการ clip หรือไม่ จะถูกทดสอบโดยนำแต่
ละขอบของ polygon มาทดสอบกับขอบเขตของการ clip ที่กำหนดไว้

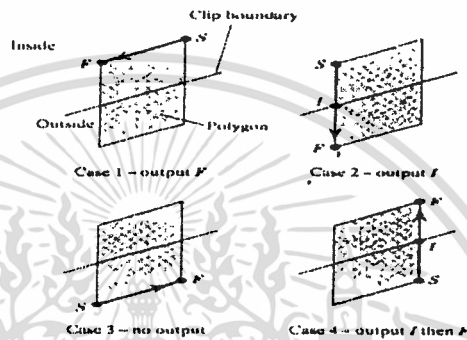


รูปที่ 4.2 การตัดแบบ sutherland-hodgman

การทดสอบมี algorithm คือนำขอบของ polygon มา 1 เส้น ทดสอบกับขอบเขตของ
การ clip 1 ด้าน จากนั้นนี้ได้ผลลัพธ์เป็น vertex แต่ละจุดที่จะถูกรวมไว้ในรายการ vertex ของ
polygon ที่อยู่ภายในพื้นที่ที่กำหนดเป็น clipping volume ดังในรูป 4.3 แสดงกรณีที่เป็นไปได้ 4
กรณีขอบกำหนดด้วยจุด vertex S และ F

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในกรณีแรกขอบจะอยู่ภายในขอบเขตและ vertex F จะเป็น output ที่รวบรวมไว้ ในกรณีที่สองขอบได้ข้ามขอบเขตจึงทำให้เกิด vertex I เป็น output ที่รวบรวม ในกรณีที่สามขอบอยู่ภายนอกอย่างสมบูรณ์ซึ่งไม่เกิด output ใดๆ และในกรณีสุดท้ายก็ทำให้เกิด vertex เป็น output ที่รวบรวมเช่นกัน

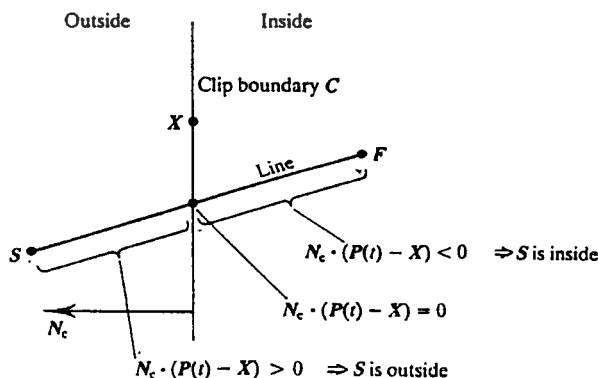


รูปที่ 4.3 การตัดแบบ sutherland-hodgman ภายในรูปหลายเหลี่ยม

การคำนวณว่า vertex จุดนั้นอยู่ภายในหรือภายนอกขอบเขตการ clip ใช้การทดสอบด้วย dot product รูปที่ 4.4 แสดง clip boundary c พร้อมด้วย outward normal N_c และเส้นที่มีจุดเริ่มและจุดปลายคือ S และ F ดัง สมการคือ

$$P[t] = S + [F - S] t$$

เมื่อ $0 \leq t \leq 1$



รูปที่ 4.4 dot product ซึ่งใช้หาว่าเส้นอยู่ภายในหรือภายนอกขอบเขต

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เรานิยามจุดบนขอบเขตเป็น X และเวกเตอร์จากจุด X ไปจุดใด ๆ บนเส้น ผลการคำนวณ dot product ของเวกเตอร์นี้กับ normal ทำให้แยกแยะ ได้ว่าจุดบนเส้นนั้นอยู่ภายใน ภายนอก หรือบนขอบเขต ตามรูปที่ 4.4 จะได้สมการคือ

$$N_c \cdot (S - X) > 0 \Rightarrow S \text{ อยู่ภายนอกบริเวณ clip}$$

$$N_c \cdot (F - X) < 0 \Rightarrow F \text{ อยู่ภายในบริเวณ clip}$$

และ

$$N_c \cdot (P(t) - X) = 0$$

นิยามให้เป็นจุดที่อยู่ทั้งบนเส้นและบนขอบเขต

4.3 incremental shading techniques

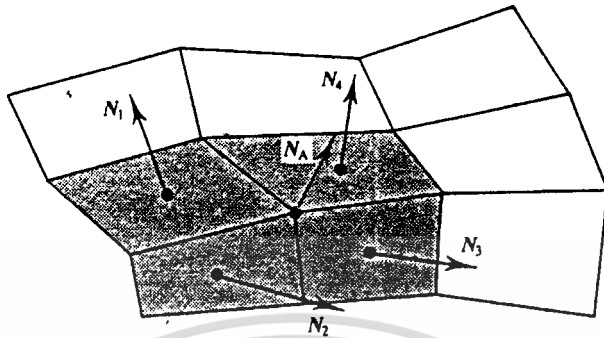
ในรูปจำลองโครงข่าย polygon จะมีข้อมูลทางพีชคณิตเฉพาะที่ vertices แต่เราสามารถทำการ shading ส่วนอื่นๆ ได้โดยทำการคำนวณความเข้มแสงที่จุด vertices แล้วทำการ interpolate ไปยังจุดต่างๆ ภายใน polygon ตามลำดับเส้น scanline (จากซ้ายไปขวา) ซึ่งเป็นหลักการวิธีของ incremental หรือ interpolative shading โดยหลักการดังกล่าวนี้สามารถนำไปใช้ร่วมกับ Z buffer scanline hidden surface removal algorithm

incremental shading techniques แบ่งเป็น 2 วิธีคือ Gouraud interpolation และ Phong interpolation วิธีแรกจะเหมาะสมสำหรับคุณภาพมากกว่าและคำนวณง่ายกว่า แต่ Phong จะให้ความแม่นยำกว่า

4.3.1 Gouraud Shading

เป็นวิธีที่แก้ไขข้อเสียของหลักการ constant shading ที่ใช้การ interpolation intensity แบบ bilinear วิธี gouraud จะไม่คำนวณทั้งหมดของส่วนที่มองเห็น

ในระบบการมองเห็นของมนุษย์จะไวต่อความเข้มแรกที่ได้มาเพื่อที่จะตรวจจับแล้วขยายไปขอบ วิธี Gouraud จะเข้มงวดในการแพร่ส่วนประกอบของ reflection model เพราะการ shade ด้วยวิธีนี้ขึ้นกับโครงข่าย polygon อย่างมาก

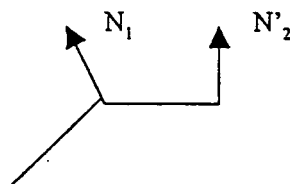
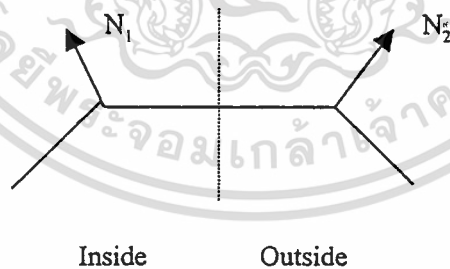


รูปที่ 4.5 แสดงเวกเตอร์ N_A ซึ่งเฉลี่ยมาจากเวกเตอร์ N_1, N_2, N_3 และ N_4
ขั้นแรกคำนวณ intensity ของ vertex ในแต่ละ polygon โดยใช้สมการ

$$I_r K_d (L \cdot N) / (r + K)$$

เมื่อ N คือ vertex normal จะใช้การคำนวณนี้หาค่าเฉลี่ยจาก normal ของแต่ละ polygon ที่ใช้ vertex ร่วมกัน (ดูรูป 4.5) นี่เป็นส่วนสำคัญของวิธีนี้ ค่า vertex normal เป็นการคำนวณหาค่าประมาณของ true normal ของพื้นผิวที่จุดนั้น

เมื่อคำนวณ normal vertex แล้วเก็บเป็น database ของ object เมื่อวัตถุถูก clip จะต้องมีการคำนวณใหม่อีกครั้ง (ค่า normal vertex และ vertex intensity) โดย normal vertex ภายนอกจะถูกคำนวณร่วมกับ normal vertex ภายใน จะได้เป็น normal vertex ที่ขอบนอก

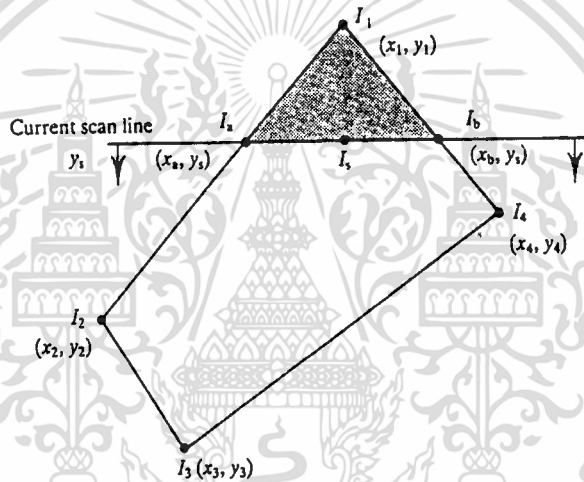


รูปที่ 4.6 นอร์มอลเวกเตอร์และการตัด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากนั้นใช้ข้อมูลมาคำนวณ intensity, I , ของแต่ละ vertex แล้วกระจายเข้าไปในพื้นที่ผิว โดยรวมการคำนวณ scan conversion ใน algorithm ซึ่งจะคำนวณตำแหน่งขอบของ polygon จาก vertex ใน data structure

ค่า intensity ที่ขอบรูปในแต่ละ scan line ถูกคำนวณจาก vertex intensity และค่า intensity ตามเส้น scan line การกระจายตามสมการ



รูปที่ 4.7 แสดงการใช้สมการหาค่าความเข้มแสง

$$I_a = [I_1(Y_i - Y_2) + I_2(Y_1 - Y_i)] / [Y_1 - Y_2]$$

$$I_b = [I_1(Y_i - Y_4) + I_4(Y_1 - Y_i)] / [Y_1 - Y_4]$$

$$I_s = [I_a(X_b - X_s) + I_b(X_s - X_a)] / [X_b - X_a]$$

จากสมการสุดท้ายข้างต้นเป็นการคำนวณทุก pixel ถ้าให้ ΔX เป็นระยะทางตามเส้น scan line และ ΔI เป็นการเปลี่ยนแปลงความเข้มแสงระหว่าง pixel

$$\Delta I_s = \Delta X (I_b - I_a) / (X_b - X_a)$$

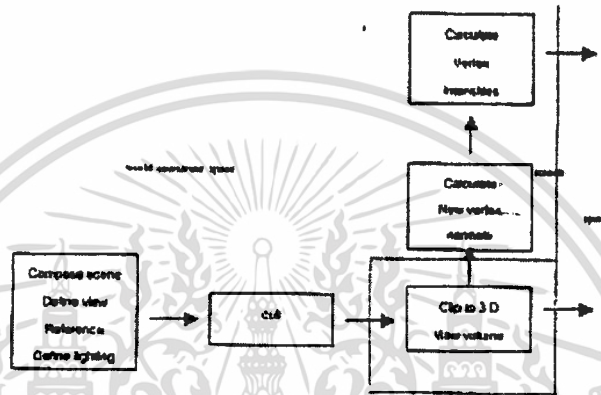
$$I_{s,n} = I_{s,n-1} + \Delta I_s$$

ความผิดพลาดที่อาจเกิดขึ้นได้

1. Anomalies เกิดขึ้นในการแสดงภาพ animate เพราะการคำนวณการกระจายความเข้มแสงนั้นทำในคู่ลำดับในจอภาพ แต่ค่าความเข้มแสงนำมาจากคู่ลำดับในความเป็นจริง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2. ในการหาค่าเฉลี่ยของ surface normal เพื่อหา vertex normal ในการคำนวณความเข้มแสง อาจเกิดความผิดพลาดในบางพื้นผิว เช่น ผิวแบบลูกฟูก ซึ่งค่า vertex normal ที่ได้จะมีทิศทางเดียวกัน ทำให้มองเห็นเป็นผิวเรียบ
- ขั้นตอนต่าง ๆ สามารถแสดงดังรูปต่อไปนี้

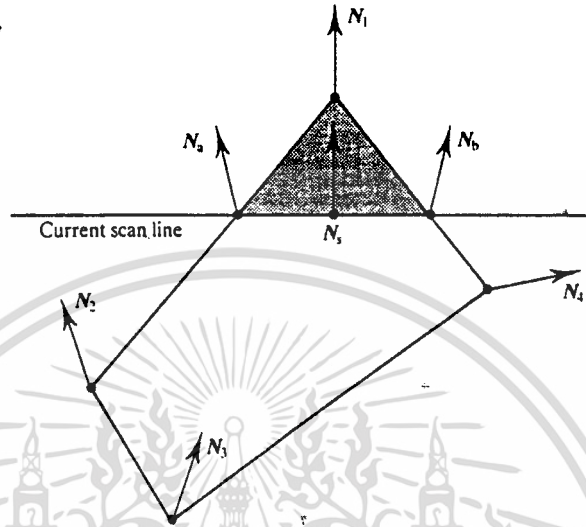


รูปที่ 4.8 ขั้นตอนการเรนเดอร์ภาพ 3 มิติ สำหรับ Gouraud Shading

4.3.2 Phong Interpolation

จะทำให้การ shading ที่จุดกึ่งกลางของ polygon แม้ว่าบาง vertex normal จะทำมุมในองศาที่ไม่เกิดเงาได้ สาเหตุสำคัญของวิธีนี้คือ

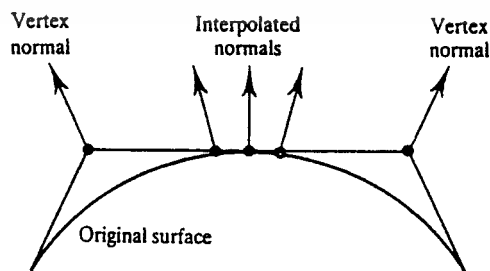
1. ยังคงใช้หลักการ bilinear interpolation ทำให้จุดภายในคำนวณได้อย่างมากขึ้น
2. การ interpolate ใช้ vertex intensity มากกว่าใช้ vertex normal แต่ยังคงหาค่าเฉลี่ยของ vertex normal สำหรับพื้นผิว
3. separate intensity จะใช้คำนวณสำหรับแต่ละ pixel จาก interpolate normal
4. ยังคงใช้จุดมองและแหล่งกำเนิดแสงอยู่ที่จุด ∞ เพื่อให้ความเข้มแสงที่จุดเป็นผลมาจาก interpolated normal



รูปที่ 4.9 เวกเตอร์ความเข้มแสงที่สร้างขึ้นในวิธีการแบบ Phong

ขั้นแรกในขั้นตอนการจะเหมือนกับวิธี Gouraud Method ที่จะหา vertex normal ของแต่ละ polygon สำหรับแต่ละเส้น scan line ใน polygon จะทำการคำนวณ linear interpolation ของจุดปลายทั้งสอง ดังแสดงในรูปที่ 4.9 ใช้ N_4 และ N_5 มา interpolate N_1 จากนั้นทำการหา normal vector ของทุก pixel ใน polygon ซึ่งเป็นการประมาณ normal จริงของผิวโค้งลงบนรูป polygon

จาก N_1 ที่ได้จะนำมาใช้คำนวณความเข้มแสง การ interpolate ค่า vector ทำในคู่ลำดับจริง ทำให้ยังคงความผิวโค้งของผิวจริงได้ดังรูป 4.10



รูปที่ 4.10 เวกเตอร์ที่สร้างขึ้นมาเพื่อสร้างเส้นโค้ง

จากรูปที่ 4.7 และ 4.9 จะได้สมการคือ

$$N_x = [N_1(Y_3 - Y_2) + N_2(Y_1 - Y_3)] / [Y_1 - Y_2]$$

$$N_y = [N_1(Y_3 - Y_4) + N_4(Y_1 - Y_2)] / [Y_1 - Y_4]$$

$$N_z = [N_3(Y_6 - Y_5) + N_6(Y_3 - Y_2)] / [Y_6 - Y_4]$$

สมการแต่ละอันจะนำไปใช้กับอีก 3 สมการเพื่อให้เป็นเวกเตอร์ในความจริงจึงทำให้การคำนวณใช้เวลามากกว่าวิธี Gouraud shading 3 เท่า ถ้าหากคำนวณความเข้มแสงทุก pixel ก็สามารถทำได้

$$N_{sx,n} = N_{sx,n-1} + \Delta N_{sx}$$

$$N_{sy,n} = N_{sy,n-1} + \Delta N_{sy}$$

$$N_{sz,n} = N_{sz,n-1} + \Delta N_{sz}$$

เมื่อ N_{sx}, N_{sy}, N_{sz} คือส่วนประกอบของ N_z

4.3.3 เปรียบเทียบวิธีทั้งสอง

วิธี Gouraud Shading จะเหมาะสมสำหรับผิวที่สะท้อนอย่างแพร่ออกไป แต่ถ้าใช้กับผิวที่สะท้อนเป็นรูปเฉพาะ การสะท้อนนั้นจะขึ้นกับตำแหน่งของ polygon วิธีนี้มีข้อดีคือ ใช้เวลาในการคำนวณน้อย ต้องการเพียงการคำนวณ intensity ของ vertex แล้ว interpolation แบบ bi linear เป็นค่า intensity ของแต่ละ pixel

วิธี Phong Shading จะให้การสะท้อนที่ไม่ขึ้นกับ polygon พื้นฐาน แต่จะใช้เวลามากกว่า เพราะต้อง interpolation ค่า surface normal แล้วจึงคำนวณ intensity แต่ละ pixel

การจะทำให้ Phong Shading ทำงานได้เร็วขึ้นต้องทำงานร่วมกับ Gouraud Shading

4.4 Rasterization หรือ scan conversion

ตามที่ได้ศึกษามาแล้วว่าจุดโดยทั่วไปในรูป polygon สามารถกำหนดความเข้มแสงโดยหาจากค่าของ vertex ตอนนี้มาดูว่าเราจะหา pixel ที่แท้จริงที่เราต้องการความเข้มแสงอย่างไรเรียกการให้ค่า intensity ของ pixel ใน polygon โดยใช้ vertex มาคำนวณ ว่า Rasterization หรือ scan conversion จะแบ่งการพิจารณาปัญหานี้เป็น 2 ขั้นตอน

1. หาว่า pixel เป็นปลายของ polygon หรือเปล่า

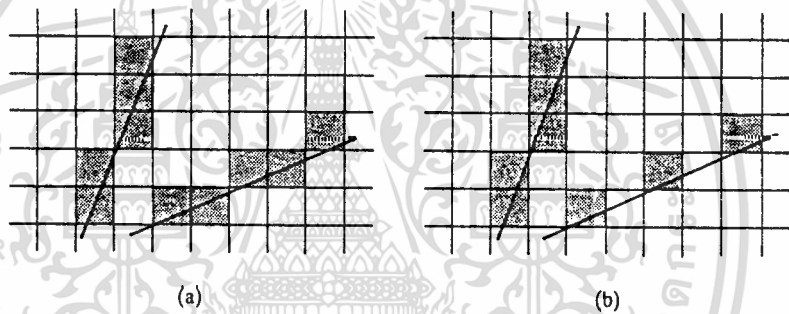
2. จัดข้อมูลนี้เพื่อหาจุดภายใน ได้อย่างไร

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ไว้สำหรับใช้ในการเรียนการสอนเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.4.1 Rasterizing Edges

แบ่งเป็นสองวิธีที่จะทำ rasterizing ขอบ ขึ้นกับว่าเป็นเส้นวาดหรือแผ่นเต็ม ในที่นี้จะไม่กล่าวถึงเกี่ยวกับลายเส้น (algorithm ของการวาดลายเส้นเป็นเพียงการสร้างลำดับของ pixel ที่ไม่มีช่องว่าง) สำหรับรูปแผ่นเต็มจะมีข้อเข้มน้อยกว่า เราสามารถเติม polygon โดยใช้เส้นในแนวนอนซึ่งคล้ายกับการ intersect กันระหว่างรูป polygon กับ เส้น scan line โดยเฉพาะ ดังนั้นจากการที่ทุกเส้น scan line มีขอบเขตทั้งด้านซ้ายและขวา ทำให้การ intersect กับ polygon จะได้ขอบด้านซ้ายและด้านขวาของรูป polygon ใน 1 เส้น scan line ดังนั้นที่ขอบแต่ละด้านจะได้ผลลัพธ์เป็นลำดับของ pixel ที่มีการ intersect ขึ้น ซึ่งอาจเกิดช่องว่างได้ดังรูป



รูปที่ 4.11 ลำดับจุดที่ใช้ใน (a) การวาดเส้น (b) โพลีกอนฟิลลิ่ง

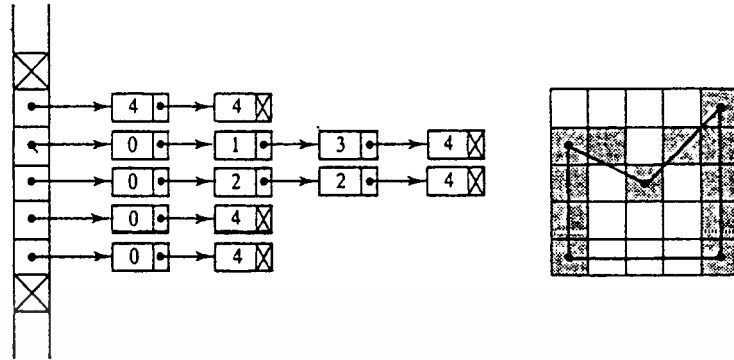
ปกติการหา pixel ดังกล่าวจะใช้ digital differential analyzer หรือเรียก DAA

4.4.2 Rasterizing Polygon

หลังจากที่เราหา pixel ตามขอบของ polygon ได้แล้ว เราจะเติม polygon ให้เต็มซึ่งในงานการ shading หมายถึงการหาค่าลำดับของ pixel ที่อยู่ภายใน polygon แล้วกำหนดค่าได้จากกระบวนการ shading ให้แต่ละจุด

ขั้นแรกเราจะหาจุดปลายแต่ละ segment แล้วเติมให้เต็มในแนวราบระหว่างจุดทั้งสอง ซึ่งสามารถได้มาจากการสร้าง edge list สำหรับแต่ละ polygon ทำได้โดยใช้อาร์เรย์ linked list โดยมีสมาชิกแต่ละตัวเป็นเส้น scan line 1 เส้น

เริ่มแรกเซตให้ทุกสมาชิกเป็น null แล้วนำแต่ละพิกัดของ pixel ในขอบที่ได้จากการ rasterizing มาลงใน list สมาชิกแต่ละตัวเรียงตามลำดับค่า y และในสมาชิก 1 ตัวเรียงตามค่า x ที่เพิ่มขึ้น ดังแสดงในรูป 4.12



รูปที่ 4.12 ตัวอย่างการรักษาการเชื่อมต่อใน polygon pasteurization

การเติม polygon ให้เต็มในแต่ละเส้น scan line โดยเลือกคู่ของ X มาแล้วเติมในช่องว่างระหว่างคู่ นั้น

ในทางปฏิบัติ การเรียงลำดับใน list จะวางในตำแหน่งที่ถูกต้องในครั้งแรกเลย ไม่มาเรียงใหม่ทั้งหมดอีก นอกจากนี้ภายหลังจากได้ค่า X มา 1 ค่าของ pixel ในเส้นขอบจะทำการคำนวณค่า shading ไปด้วยกัน

บทที่ 5

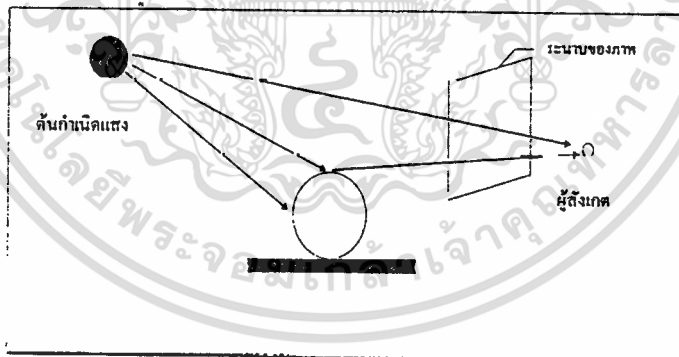
RAY TRACING

5.1 การจำลองภาพแบบ Ray Tracing

Ray Tracing เป็นวิธีการเลียนแบบการส่องสว่าง-การสะท้อนของภาพจำลองในคอมพิวเตอร์กราฟิก โดยจากการให้ต้นกำเนิดแสงมีระยะทางไกลมากจนทำให้มีลักษณะเป็นตรงในตัวเองเดียวกัน ซึ่งหลักการพื้นฐานคือ ผู้สังเกตจะมองเห็นจุดบนพื้นผิวได้เนื่องจากแสงจากต้นกำเนิดแสงที่มากกระทบกับวัตถุแล้วเกิดการสะท้อนเข้าตาผู้สังเกต การจำลองภาพโดยวิธี Ray Tracing นี้แบ่งได้เป็น 2 วิธี คือ

5.1.1 การจำลองภาพแบบ Ray Tracing โดยวิธีติดตามแสงจากต้นกำเนิดแสง

ถ้ากำหนดว่าแสงจากต้นกำเนิดแสงใดๆ เป็นอนุภาคหนึ่งที่สามารถเคลื่อนที่ได้ในบรรยากาศและภาพที่มองเห็นจะเกิดจากอนุภาพเหล่านี้กระทบวัตถุแล้วสะท้อนมาสู่ผู้สังเกต เนื่องจากอนุภาคที่สะท้อนมาสู่ผู้สังเกต ได้นั้นมีเพียงส่วนน้อยเมื่อเทียบกับอนุภาคที่มาจากต้นกำเนิดแสง จึงเป็นการเสียเวลามากในการพิจารณาทุก ๆ อนุภาคจากต้นกำเนิดแสง

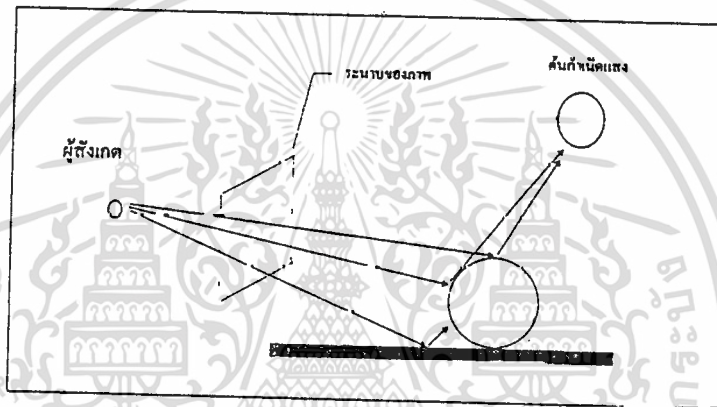


รูปที่ 5.1 การจำลองภาพแบบ Ray tracing โดยวิธีติดตามแสงจากต้นกำเนิดแสง

5.1.2 การจำลองภาพแบบ Ray Tracing โดยวิธีติดตามทิศทางมุมมองของผู้สังเกต

โดยเริ่มต้นพิจารณาดำแหน่งผู้สังเกตไปสู่ต้นกำเนิดแสง โดยอาศัยขนาดของระนาบของภาพเป็นตัวช่วยในการสร้างภาพซึ่งบริเวณภายนอกจะไม่มีมีการพิจารณา วิธีการจำลองภาพแบบนี้ทำได้โดยสร้างเวกเตอร์จากตำแหน่งผู้สังเกตผ่านทุก ๆ จุดบนระนาบของภาพ ถ้าเวกเตอร์จากตำแหน่งของผู้สังเกตผ่านตำแหน่งใด ๆ บนระนาบของภาพแล้วตกกระทบกับวัตถุแสดงว่าที่เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปเผยแพร่โดยไม่ผ่านการอนุญาต ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตำแหน่งนี้บนระนาบของภาพจะสามารถมองเห็นวัตถุที่ตำแหน่งที่ตกกระทบได้ จากนั้นจึงหาความเข้มแสงของแสงและสีที่ตกกระทบนั้น และที่ตำแหน่งของจุดตกกระทบจำเป็นที่จะต้องคำนวณหาจุดตกกระทบกับวัตถุแรก (first sit point) โดยอาศัยเวลาที่ใช้ในการตกกระทบกับวัตถุต่าง ๆ ในระบบเป็นตัวตัดสิน วัตถุในระบบที่มีเวลาตกกระทบน้อยที่สุดจะเป็นวัตถุแรกที่เวกเตอร์นี้ไปตกกระทบ เนื่องจากผู้สังเกตเห็นวัตถุอยู่ใกล้จุดสังเกตมากที่สุดเสมอ การหาค่าความเข้มแสงสามารถหาได้โดยวิธี Phong Shading



รูปที่ 5.2 การจำลองภาพแบบ Ray tracing โดยวิธีติดตามทิศทางการมองของผู้สังเกต

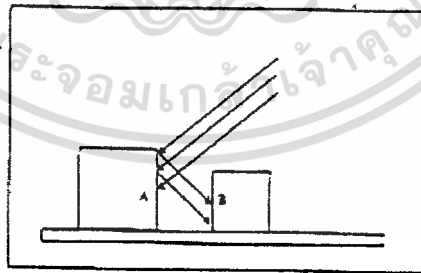
บทที่ 6

แสง สี และ ลำดับความเข้ม

6.1 การสะท้อนแสงของวัตถุ

วิธีการพื้นฐานในการสร้างภาพจำลองของวัตถุที่กล่าวไว้ในบทก่อนหน้านี้ไม่สามารถที่จะสร้างภาพจำลองที่มีความเหมือนจริง (realistic) ได้ แต่เป็นการสร้างภาพจำลองโดยกำหนดลักษณะกว้างๆเท่านั้น ในการจำลองภาพของวัตถุใดๆให้มีความเหมือนจริงนั้นสิ่งสำคัญที่สุดที่ต้องนำมาพิจารณา คือ แสงบนพื้นผิวของวัตถุ วิธีการนี้เรียกว่า rendering วัตถุต่างๆที่สามารถมองเห็นได้ในธรรมชาตินั้น โดยมากเกิดจากการสะท้อนของแสง (reflection of light) จากพื้นผิวของวัตถุมาสู่ดวงตาของผู้สังเกต ซึ่งจะทำให้ผู้สังเกตสามารถบอกความแตกต่างของวัตถุได้ เช่น สี รูปร่าง หรือลวดลายต่างๆของพื้นผิวของวัตถุ (texture) ได้ แสงที่มากระทบกับวัตถุนั้นสามารถมาจากหลายทิศทางและด้วยความเข้ม (intensity) ที่แตกต่างกัน ซึ่งเป็นการยากที่จะจำลองภาพของวัตถุให้มีความเหมือนจริงโดยการจำลองแสงต่างๆที่มากระทบกับวัตถุเพื่อให้สามารถมองเห็นวัตถุนั้น

วัตถุต่างๆที่สามารถมองเห็นได้นั้นเป็นผลมาจากแสงที่มาจากต้นกำเนิดแสงที่มีทิศทางต่างกัน เช่น แสงจากดวงอาทิตย์ หรือ แสงที่สะท้อนมาจากพื้นผิวของวัตถุอื่นๆ จากรูปที่ 6.1 จะเห็นว่าพื้นผิว A จะกระทบกับแสงจากต้นกำเนิดแสงโดยตรงในขณะที่พื้นผิว B จะถูกกระทบโดยแสงที่สะท้อนมาจากพื้นผิว A

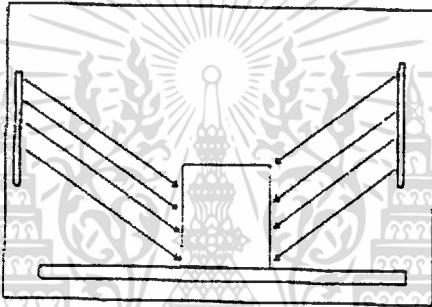


รูปที่ 6.1 แสดงการตกกระทบของแสงจากต้นกำเนิดแสง

แสงที่ตกกระทบกับวัตถุสามารถแยกออกได้เป็น แสงที่มาจากต้นกำเนิดแสงโดยตรง (emitted light) และแสงที่เกิดจากการสะท้อนจากพื้นผิวต่างๆ เรียกว่า แสงโดยรอบ (ambient light, background light) แสงโดยรอบสามารถที่จะเกิดได้จากต้นกำเนิดแสงหลายชนิดด้วยกันไม่

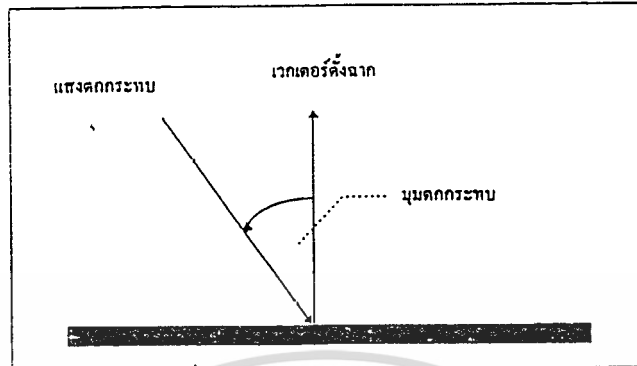
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ว่าจะเห็นต้นกำเนิดแสงจุดเดียวหรือว่าต้นกำเนิดแสงจำนวนมากก็ได้ แสงที่ส่องสว่างจากพื้นผิวขนาดเล็กๆ เรียกว่า แสงจากต้นกำเนิดแสงแบบจุด (point light source) แสงที่ส่องสว่างจากต้นกำเนิดแสงแบบจุดนี้จะมีคุณสมบัติพิเศษ คือ ทิศทางของแสงจากต้นกำเนิดจะมีทิศทางเดียวและมีค่าความเข้มของแสงเท่ากัน เมื่อสังเกตจากตำแหน่งใดตำแหน่งหนึ่งในระบบ ส่วนแสงที่เกิดมาจากพื้นผิวใดๆ จะเรียกต้นกำเนิดแสงนั้นว่าต้นกำเนิดแสงแบบกระจาย (distributed light source) ลำแสงต่างๆ ที่มาจากต้นกำเนิดแสงแบบกระจายจะกระทบจุดที่กำหนดจากหลายทิศทางและยังมีความเข้มต่างกันด้วย ดังรูปที่ 6.2



รูปที่ 6.2 การตกกระทบของแสงจากต้นกำเนิดแสงแบบกระจาย

ถึงแม้ว่าดวงอาทิตย์จะเป็นต้นกำเนิดแสงที่มีขนาดใหญ่ แต่เนื่องจากระยะทางระหว่างโลกกับดวงอาทิตย์มีค่ามาก จึงสามารถทำให้คิดว่าดวงอาทิตย์สามารถแสดงลักษณะเป็นต้นกำเนิดแสงแบบจุด ดังนั้นสามารถที่จะกำหนดได้ว่า ลำแสงจากต้นกำเนิดแสงที่มีระยะห่างจากวัตถุมากๆ จะมีลักษณะเป็นลำแสงที่ขนานกัน และจะตกกระทบกับทุกตำแหน่งบนพื้นผิวด้วยมุมที่เท่ากัน เรียกมุมนี้ว่า มุมตกกระทบของแสง (angle of incident) ดังแสดงในรูปที่ 6.3 เนื่องจากจุดทุกจุดบนพื้นผิวจะมีระยะทางที่เกือบจะเท่ากันจากต้นกำเนิดของแสงที่อยู่ที่ระยะไกล จึงสามารถทำให้กำหนดได้ว่าความเข้มของแสงที่มากตกกระทบจะมีค่าเท่ากัน



รูปที่ 6.3 มุมตกกระทบของแสง

ปริมาณของแสงที่สะท้อนมาจากพื้นผิวของวัตถุนั้นจะขึ้นอยู่กับลักษณะพื้นผิวและสีของวัตถุ รวมถึงลักษณะของแสงที่ตกกระทบกับพื้นผิวนั้นด้วย การสะท้อนแสงแบบกระจาย (diffuse reflection) เกิดขึ้นเมื่อมีแสงตกกระทบกับพื้นผิวแล้วพื้นผิวนั้นจะดูดกลืน (absorb) พลังงานของแสงบางส่วนไว้ และปลดปล่อย (emit) แสงออกมาในทุกทิศทาง ซึ่งลักษณะของพื้นผิว เช่น สี และ ลวดลายต่างๆสามารถมองเห็นได้โดยการสะท้อนแสงแบบกระจาย ส่วนการสะท้อนแสงแบบกรวยหรือการสะท้อนแสงแบบกระจก (specular reflection) เป็นการสะท้อนของแสงโดยตรงจากพื้นผิวของวัตถุ ดังรูปที่ 6.4 พื้นผิวที่มีลักษณะมันวาว (shiny surface) จะสะท้อนแสงที่มากตกกระทบเกือบทั้งหมด ซึ่งจะทำให้เกิดจุดสว่าง (hot spot ,highlight) ขึ้นสำหรับพื้นผิวของวัตถุที่มีลักษณะหยาบ (rough) การสะท้อนของแสงแบบกรวยจะเกิดขึ้นน้อยมากหรือไม่เกิดขึ้นเลย

การสะท้อนแสงจากพื้นผิวของวัตถุนั้นจะเป็นผลรวมของการสะท้อนหลายชนิดด้วยกันซึ่งเป็นผลมาจากแสงที่อยู่โดยรอบและแสงที่มาจากต้นกำเนิดแสงโดยตรง

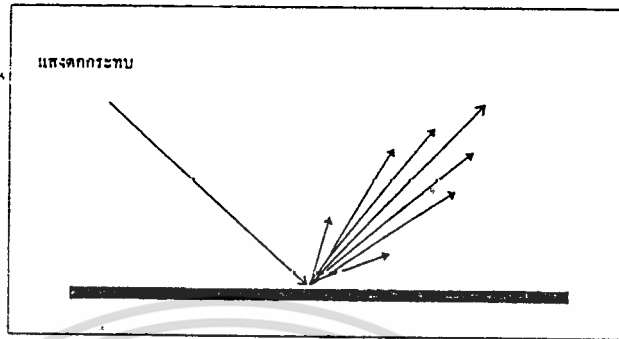
$$I = R_s + R_d + R_r$$

I = ผลรวมของแสงสะท้อนจากผิวของวัตถุ

R_s = การสะท้อนของแสงโดยรอบ

R_d = การสะท้อนแสงแบบกระจาย

R_r = การสะท้อนแสงแบบกรวยหรือการสะท้อนแสงแบบกระจก



รูปที่ 6.4 การสะท้อนแสงแบบกรวย

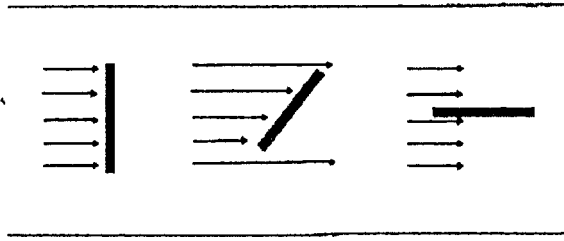
แสงที่กระจายอยู่โดยรอบจะตกกระทบบกับวัตถุในเวลาเดียวกัน ด้วยค่าความเข้มที่เท่ากัน ในทิศทางต่างๆกัน การกระจายแสงจะเป็นลักษณะที่เหมือนกัน (uniformly) ในทิศทางต่างๆ ถ้าความเข้มของแสงที่กระจายอยู่โดยรอบคือ I_0 ค่าความเข้มของแสงที่สะท้อนจากแสงสีที่กระจายอยู่โดยรอบ คือ

$$R_s = K_d I_0$$

เมื่อ K_d คือค่าสัมประสิทธิ์การสะท้อนแสงของพื้นผิวของวัตถุ (coefficient of reflectivity) หรืออัตราส่วนของแสงที่สะท้อนจากพื้นผิวของวัตถุและจำนวนของแสงที่ตกกระทบบทั้งหมด

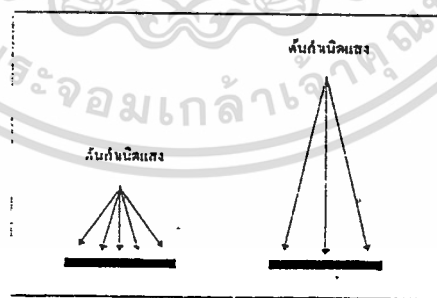
วัตถุที่มีพื้นผิวสีขาวจะสามารถสะท้อนแสงได้เกือบทั้งหมดจากจำนวนของแสงที่มาตกกระทบบ ดังนั้นค่าสัมประสิทธิ์การสะท้อนของแสงจะมีค่าใกล้เคียง 1 ส่วนวัตถุที่มีพื้นผิวสีดำจะดูดกลืนแสงที่มาตกกระทบบเกือบทั้งหมด ดังนั้นค่าสัมประสิทธิ์การสะท้อนจะมีค่าใกล้เคียง 0 วัตถุทุกชนิดจะสะท้อนบางสีของแสงที่มาตกกระทบบได้มากกว่าสีอื่นๆ เช่น วัตถุที่มีค่าสัมประสิทธิ์การสะท้อนของแสงที่มีปริมาณความเข้มของแสงทุกสีเท่ากัน มาตกกระทบบจะทำให้ส่วนของแสงสีแดงสะท้อนกลับได้มากกว่าสีอื่นๆจะทำให้เห็นวัตถุนั้นมีสีแดง

เมื่อแสงที่มาจากด้านกำเนิดแสงตกกระทบบกับวัตถุโดยตรงในทิศทางที่แน่นอนจะสะท้อนในทุกทิศทาง ดังนั้น เมื่อมุมของแสงที่ตกกระทบบมีค่ามากขึ้น พื้นผิวของวัตถุจะรับแสงได้น้อยลง ดังนั้นแสงที่สะท้อนจากพื้นผิวของวัตถุมีค่าน้อยลงด้วย ดังแสดงในรูปที่ 6.5



รูปที่ 6.5 แสงที่ตกกระทบกับพื้นผิวที่มุมต่างๆ

ความสัมพันธ์ดังกล่าวสามารถอธิบายโดย กฎของ Lambert ที่กล่าวว่าความเข้มของแสงที่สะท้อนแบบกระจายจะเป็นสัดส่วนโดยตรงกับค่า cosine ของมุมของแสงที่ตกกระทบ เมื่อต้นกำเนิดแสงอยู่บริเวณใกล้เคียงกับพื้นผิวของวัตถุ แสงจะตกกระทบที่ผิวของวัตถุด้วยมุมที่แตกต่างกันเนื่องจากแสงที่ตกกระทบไม่ได้มีทิศทางที่ขนานกัน ดังแสดงในรูปที่ 6.6 มุมของแสงที่ตกกระทบกับพื้นผิวจะมีหลายค่าด้วยกันซึ่งทำให้ยากต่อการคำนวณค่าความเข้มของแสงที่สะท้อนจากวัตถุ เพื่อความสะดวกในการคำนวณจึงกำหนดให้ต้นกำเนิดแสงอยู่ห่างจากวัตถุมากพอที่จะทำให้ต้นกำเนิดแสงนั้นเป็นเหมือนต้นกำเนิดแสงแบบจุด ซึ่งจะทำให้มุมของแสงที่ตกกระทบที่จุดทุกจุดบนพื้นผิวมีค่าเท่ากัน

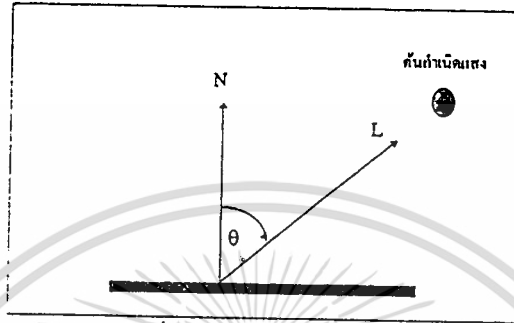


รูปที่ 6.6 ทิศทางของแสงที่ตกกระทบ เมื่อต้นกำเนิดแสงอยู่ใกล้เคียงกับวัตถุ

สำหรับต้นกำเนิดแสงแบบจุดใดๆนั้นค่า cosine ของมุมตกกระทบ สามารถคำนวณหาได้จาก dot product ของเวกเตอร์หนึ่งหน่วย (unit vector) ที่ตั้งฉากกับพื้นผิวของวัตถุ N และเวกเตอร์หนึ่งหน่วยที่ชี้ไปในทิศทางของต้นกำเนิดแสง L ดังรูปที่ 6.7

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

$$\cos \theta = N \cdot L$$



รูปที่ 6.7 มุมของแสงตกกระทบ

ถ้าคั่นกำเนิดแสงมีพลังงาน E_p พลังงานของแสงที่ไปถึงพื้นผิวของวัตถุจะขึ้นกับระยะทางระหว่างคั่นกำเนิดแสงและวัตถุด้วย โดยที่ความเข้มของแสงจากคั่นกำเนิดแสงที่กระทบวัตถุ (I_p) จะลดลงเป็นสองเท่าของระยะทางระหว่างคั่นกำเนิดแสงและวัตถุ

$$I = \frac{E_p}{D^2}$$

และค่าความเข้มของแสงที่เกิดจากการสะท้อนของการกระจาย คือ

$$R_d = \frac{K_d E_p}{D + D_0} (N \cdot L)$$

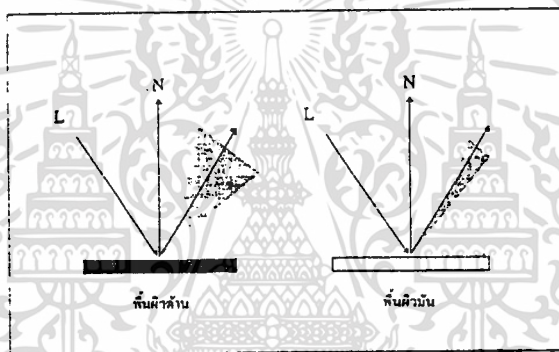
เมื่อ D_0 คือค่าคงที่และผลรวมของความเข้มของแสงที่เกิดจากการสะท้อนแบบกระจายโดยแสงที่อยู่โดยรอบและแสงจากคั่นกำเนิดแสงโดยตรง คือ

$$R_w = K_d I_a + \frac{K_d E_p}{D + D_0} (N \cdot L)$$

ขณะที่สีของพื้นผิวของวัตถุถูกกำหนดโดยการสะท้อนแสงแบบกระจาย แต่ความสว่างเอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น เมื่ออนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

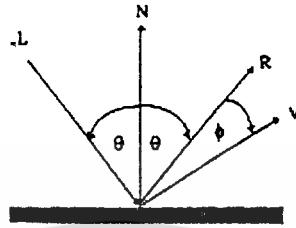
(shine) และรูปร่าง (shape) ของวัตถุจะถูกกำหนดโดย จุดสว่างที่เกิดจากการสะท้อนแสงแบบกรวย เนื่องจากมุมของการสะท้อนของแสง (angle of reflection) คือมุมระหว่างทิศทางของแสงที่สะท้อนจากผิวของวัตถุ และเวกเตอร์ที่ตั้งฉากกับผิวของวัตถุ จะเท่ากับมุมของแสงที่ตกกระทบกับพื้นผิวของวัตถุ ดังรูปที่ 6.8 ดังนั้นการสะท้อนแสงของพื้นผิววัตถุจะคล้ายกับการสะท้อนแสงของกระจก

ตัวแปรสำคัญที่เป็นตัวกำหนดลักษณะการสะท้อนแสงแบบกรวย คือ ลักษณะการกระจายตัวของพื้นผิวของวัตถุ ทิศทางของพื้นผิวของวัตถุ การดูดซับความเข้มของแสงของพื้นผิวของวัตถุและปริมาณของแสงที่ถูกบังโดยพื้นผิวของวัตถุ



รูปที่ 6.8 การสะท้อนของแสงที่พื้นผิวต่างๆ

ในการจำลองแสงแบบ phong ค่าความเข้มของแสงสะท้อนที่มุม ϕ จากมุมของแสงสะท้อน จะขึ้นกับค่าของ $\cos^n \phi$ ดังแสดงในรูปที่ 6.9 เมื่อ n คือคุณสมบัติของพื้นผิวของวัตถุที่เกี่ยวกับความมันวาว เมื่อ ϕ คือ 0 ค่าของ $\cos^n \phi$ จะมีค่าเท่ากับ 1 ดังนั้น ค่าของการสะท้อนแสงแบบกรวยจะมีค่ามากที่สุดเมื่อ ϕ มีค่าเท่ากับมุมของแสงสะท้อน และค่า n ของวัตถุที่มีพื้นผิวด้านจะมีค่าเท่ากับ 1 ส่วนวัตถุที่มีพื้นผิวมันวาวจะมีค่า n มากกว่า 1



รูปที่ 6.9 มุมตกกระทบของแสงและมุมมอง

และจากค่าสัมประสิทธิ์การสะท้อนของพื้นผิวของวัตถุที่จะมีค่าเปลี่ยนแปลงตามมุมของแสงตกกระทบ เช่น สำหรับพื้นผิวที่เป็นน้ำหรือแก้ว เมื่อมุมของแสงตกกระทบมีมาก จะทำให้สัมประสิทธิ์การสะท้อนมีค่ามากและเมื่อมุมตกกระทบของแสงมีค่าน้อย ค่าสัมประสิทธิ์การสะท้อนจะมีค่าน้อยตามไปด้วย การเปลี่ยนแปลงของค่านี้ขึ้นกับค่าความยาวคลื่นของแสง (wavelength, λ) ของแสงตกกระทบซึ่งเขียนได้ในรูป $w(\phi, \lambda)$ เรียกว่า ค่าสัมประสิทธิ์การสะท้อนแสงแบบทรงกรวย (specular reflection coefficient, specular reflectance) จากความสัมพันธ์ทั้งหมดสามารถคำนวณหาค่าของการสะท้อนแสงแบบกรวย R_s คือ

$$R_s = \frac{E_p}{D + D_0} W(\theta, \lambda) \cos^n \phi$$

สำหรับพื้นผิวทั่วไปๆ การเปลี่ยนแปลงของ $w(\phi, \lambda)$ มีค่าน้อยซึ่งสามารถแทนได้โดยค่าคงที่ k_s และ $\cos \phi = V \cdot R$

$$R_s = \frac{E_p}{D + D_0} K_s (V \cdot R)^n$$

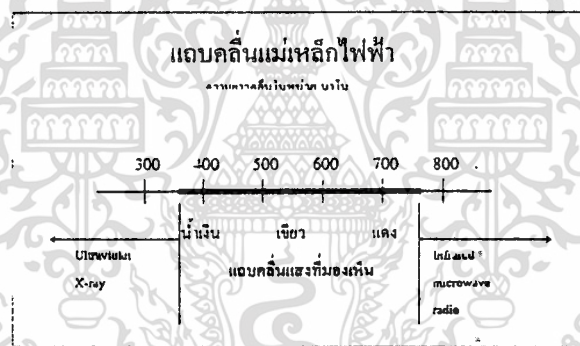
ดังนั้นสมการการสะท้อนแสงที่ผิวของวัตถุคือ

$$I = K_d I_a + \frac{E_p}{D + D_0} (K_d (N \cdot L) + K_s (V \cdot R)^n)$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

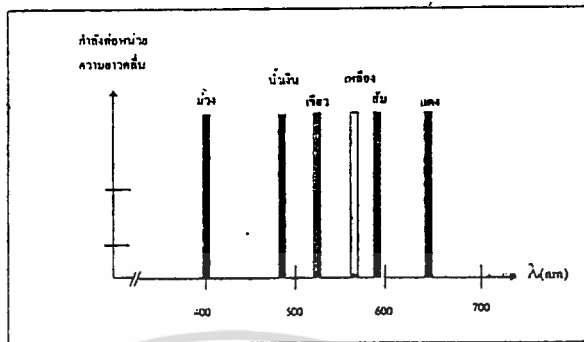
6.2 แสงและสี

สีพื้นฐานของจอภาพของคอมพิวเตอร์คือ สีแดง (Red) สีเขียว (Green) และสีน้ำเงิน (Blue) ดังนั้น จึงจำเป็นต้องใช้วิธีการบางอย่างมาช่วยในการแสดงผลของสีอื่น ๆ ที่สร้างขึ้นมาจากสีพื้นฐานทั้ง 3 สี แสงที่สามารถมองเห็นได้ (Visible light) นั้นเป็นส่วนประกอบส่วนหนึ่งของคลื่นแม่เหล็กไฟฟ้า (electromagnetic wave) รวมทั้ง คลื่นวิทยุ (Radio wave) หรือ รังสีเอ็กซ์ (X-ray) ความแตกต่างของคลื่นเหล่านี้คือค่าความยาวคลื่น ซึ่งคลื่นวิทยุจะมีช่วงความยาวคลื่นที่ยาวกว่า ความยาวคลื่นของรังสีเอ็กซ์ ส่วนแสงที่มองเห็นได้นั้นจะอยู่ในช่วงความยาวคลื่นช่วงสั้น ๆ ที่ต่อเนื่องกัน รูปที่ 6.10 แสดงช่วงความยาวคลื่นของคลื่นแม่เหล็กไฟฟ้าที่มองเห็นได้เรียกว่าแถบของคลื่นแสงที่มองเห็น (visible spectrum) ซึ่งมีความยาวคลื่นอยู่ระหว่าง 400 – 700 นาโนเมตร (nanometer ; nm ; 10^{-9} meter)



รูปที่ 6.10 ความยาวคลื่นของแม่เหล็กไฟฟ้าที่สามารถมองเห็นได้

ภายในดวงตาของมนุษย์นั้นจะมีเนื้อเยื่อที่บริเวณผนังรอบๆดวงตาที่มีความไวต่อแสงและประกอบด้วย เซลล์ตัวรับ (receptor cell) 2 ชนิดด้วยกันคือ เซลล์แบบกรวย (cone cell) และ เซลล์แบบแท่ง (rod cell) เซลล์แบบกรวยจะเป็นเซลล์ที่ตอบสนองต่อสี โดยที่มันจะตอบสนองต่อสีแดง สีเขียว สีน้ำเงิน สีต่างๆที่มองเห็นนั้นจะเป็นผลรวมของการตอบสนองต่อสีแดง สีเขียว สีน้ำเงิน เซลล์แบบแท่งจะตอบสนองต่อแสงที่มีความเข้มต่ำๆได้ดีกว่า



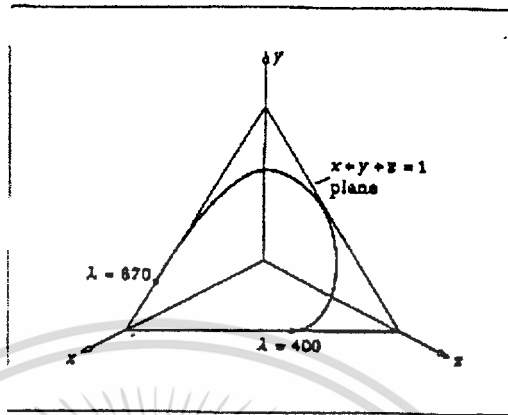
รูปที่ 6.11 สีต่างๆที่มองเห็นได้

6.2.1 ไดอะแกรม CIE chromaticity

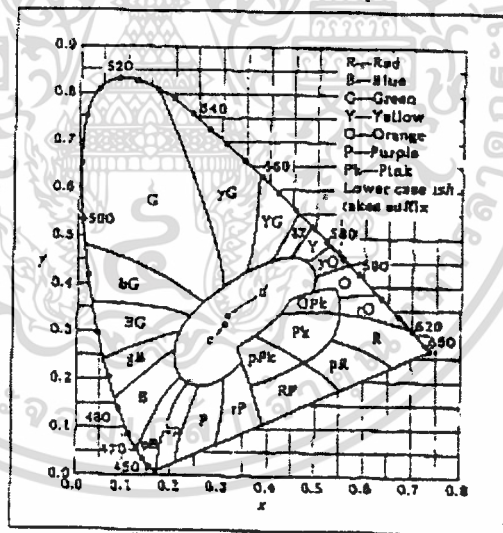
การกำหนดสีของแสงที่ต้องการโดยการผสมกันของค่าคงที่ 3 ค่าเป็นสิ่งที่มีความสำคัญมากในระบบคอมพิวเตอร์กราฟฟิก ในปี ค.ศ. 1931 CIE ได้กำหนดตัวแปรมาตรฐาน 3 ตัวด้วยกัน เรียกว่า x, y, z เพื่อแทนค่าของสีแดง สีเขียว สีนํ้าเงิน ในการผสมกันของสีเพื่อให้ได้สีของแสงที่ต้องการ ถ้า c คือสีที่ต้องการ

$$c = xX + yY + zZ$$

และการกำหนดค่าของ x, y และ z สามารถหาได้จากที่ระนาบ $x + y + z = 1$ ดังแสดงในรูปที่ 6.12 ดังนั้น เส้นโค้งของแถบสีบริสุทธิ์ (pure spectrum color curve) คือ $S(\lambda) = [X(\lambda), Y(\lambda), Z(\lambda)]$ จะอยู่ภายในบริเวณค่าที่เป็นบวกของรูปกราฟ เนื่องจากเส้นโค้งของแถบสีถูกกำหนดบนระบบ 3 มิติ บนระนาบที่ $x + y + z = 1$ ดังนั้นจึงสามารถที่จะแทนค่าไปสู่ระบบ 2 มิติได้ และ standard CIE chromaticity diagram คือเส้นโค้งของ $S'(\lambda) = [X(\lambda), Y(\lambda)]$ ดังแสดงในรูปที่ 6.13



รูปที่ 6.12 การสร้างมาตรฐาน CIE



รูปที่ 6.13 ไคอะแกรม CIE Chromaticity

6.2.2 แบบจำลองของสี

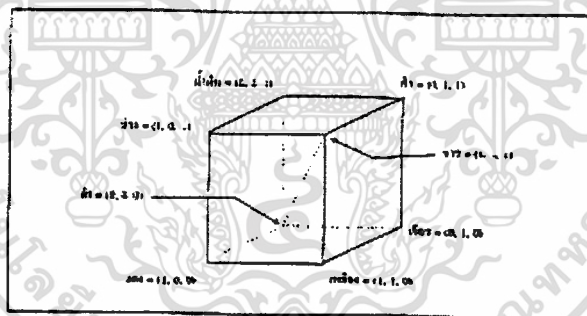
แบบจำลองของสี (color model) คือการกำหนดคุณลักษณะของสีที่ใช้ในระบบคอมพิวเตอร์กราฟิกเพื่อความสะดวกในการใช้งาน แบบจำลองของสีที่กำหนดคนนั้นสามารถแยกออกได้เป็นแบบจำลองของสีที่ใช้สำหรับอุปกรณ์ต่าง ๆ ในระบบคอมพิวเตอร์ และแบบจำลองของสีที่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ใช้ในระบบชีวิตประจำวันที่ประกอบด้วย ชนิดของสี (hue), ความอิ่มตัวของสี (Saturation) และความสว่างของสี (Brightness)

6.2.2.1 แบบจำลองของสีแบบ RGB

เป็นแบบจำลองของสีที่ใช้ในระบบจอภาพของเครื่องคอมพิวเตอร์ โดยประกอบจากสีแดง, สีเขียวและสีน้ำเงิน แบบจำลองของสีชนิดนี้ถูกกำหนดขึ้นในระบบพิกัดแบบ Cartesian ค่าของตัวแปรทั้ง 3 ตัว คือ R, G และ B นั้น สามารถที่จะนำมารวมกันได้โดยตรงเพื่อให้ได้สีที่ต้องการ โดยตัวแปร R, G และ B จะใช้แทนค่าของสีแดง, สีเขียว และสีน้ำเงินตามลำดับ แบบจำลองของสีแบบ RGB นี้ถูกกำหนดโดยรูปลูกบาศก์ขนาด 1 หน่วยดังรูปที่ 6.14 โดยแนวเส้นทะแยงมุมของรูปลูกบาศก์นั้น ค่าของตัวแปรทั้งสามจะมีค่าเท่ากันและสีที่เกิดจากค่าของตัวแปรทั้งสามตามเส้นทะแยงมุมของรูปลูกบาศก์คือ ระดับของสีเทา (gray level) และสีดำจะอยู่ที่บริเวณจุดกำเนิด ที่ตัวแปร (R,G,B) มีค่าเป็น (0,0,0) ส่วนสีขาวจะอยู่ที่ตำแหน่ง (1,1,1)



รูปที่ 6.14 แบบจำลองสีแบบ RGB

6.2.2.2 แบบจำลองของสีแบบ CMY

แบบจำลองของสีแบบ CMY เป็นแบบจำลองที่ใช้เป็นมาตรฐานในอุปกรณ์การพิมพ์ต่าง ๆ เช่น ink jet plotter ตัวแปรทั้ง 3 ของแบบจำลองของสีชนิดนี้ คือ C (Cyan, สีฟ้า), M (Magenta, สีม่วง) และ Y (Yellow, สีเหลือง) และตัวแปรทั้ง 3 ชนิดนี้ คือ C, M และ Y เป็นส่วนประกอบของตัวแปรของสีในแบบจำลองแบบ RGB ด้วย ตัวแปรทั้ง 3 นี้ถูกกำหนดขึ้นมาคล้ายกับตัวแปรในระบบ RGB แต่ต่างกันที่สีขาวจะอยู่ที่บริเวณจุดกำเนิด (0,0,0) แทนสีดำในแบบจำลองแบบของสีแบบ RGB การกำหนดสีของแบบจำลอง CMY นี้ กระทำได้โดยการแยกหรือลบสีที่ไม่ต้องการออกจากสีขาว แทนที่จะเป็นการเพิ่มสีที่ต้องการในสีดำตามแบบจำลองของสีแบบ RGB เช่น สีฟ้า ได้

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ไว้เพื่อใช้ในการศึกษาเท่านั้น ไม่สามารถนำเอกสารนี้ไปใช้
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จาก สีขาวลบด้วยสีแดงหรือ เกิดจากการรวมกันของสีน้ำเงินและเขียว ส่วนสีม่วงจะดูคลืนสีเขียว ดังนั้นจะเกิดจากสีแดงผสมกับสีน้ำเงิน สีเหลืองจะดูคลืนสีน้ำเงิน จึงเป็นผลของสีแดงผสมกับสีน้ำเงิน ดังนั้นพื้นผิวใด ๆ ที่มีสีที่เกิดจากการผสมกันของสีฟ้าและสีเหลืองจะดูคลืนสีแดงและน้ำเงิน ดังนั้น เมื่อมีแสงสีขาวมาตกกระทบ ส่วนของสีเขียวเท่านั้นจะสะท้อนออกมา ส่วนพื้นผิวที่มีสีที่เกิดจากการผสมกันของสีฟ้า สีเหลือง และสีม่วง จะดูคลืนสีแดง สีเขียวและสีน้ำเงิน เมื่อมีแสงสีขาวมาตกกระทบกับพื้นผิวจะทำให้เห็นเป็นสีดำ ดังนั้นสามารถกำหนดสมการของการเปลี่ยนแปลงจากแบบจำลองของสีแบบ RGB และแบบจำลองของสีแบบ CMY ได้คือ

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

6.2.2.3 แบบจำลองของสีแบบ YIQ

แบบจำลองของสีชนิดนี้ถูกใช้ในการแพร่ภาพทางโทรทัศน์ ในระบบของ National Television System Committee (NTSC) แบบจำลองของสีชนิดนี้อาศัยค่าพื้นฐานจากแบบจำลองของสีแบบ RGB โดยจะทำการเรียงข้อมูลของตัวแปร RGB เพื่อคุณภาพของการสื่อสาร ตัวแปร Y ในแบบจำลองสีแบบ YIQ คือ ความสว่าง (Luminance) ของสีแบบเดียวกับที่ใช้ในระบบ CIE และค่าของตัวแปร Y เท่านั้นที่ใช้ในการส่งสัญญาณโทรทัศน์จะถูกแสดงในระบบของโทรทัศน์ระบบขาวดำ ส่วนค่าของสีต่าง ๆ จะถูกกำหนดโดยตัวแปร I และ Q ซึ่งสามารถกำหนดสมการของการเปลี่ยนแปลงไปสู่แบบจำลองของสีแบบ RGB ได้ คือ

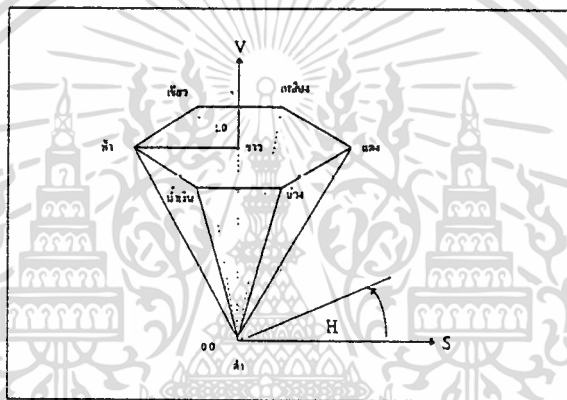
$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.144 \\ 0.596 & -0.275 & -0.321 \\ 0.212 & -0.528 & 0.311 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

6.2.2.4 แบบจำลองของสีแบบ HSV

แบบจำลองของสีที่กล่าวมาแล้วข้างต้นนั้นเป็นแบบจำลองของสีที่กำหนดขึ้นมาใช้เป็นมาตรฐานสำหรับอุปกรณ์ต่าง ๆ แต่โดยทั่วไปแล้วเป็นการยากที่จะใช้แบบจำลองของสีเหล่านี้ในการกำหนดสีที่ต้องการสำหรับผู้ใช้โดยตรง ดังนั้นแบบจำลองของสีแบบ HSV จึงถูกกำหนดขึ้นมาเพื่ออำนวยความสะดวกแก่ผู้ใช้งาน ซึ่งตัวแปรทั้ง 3 ตัวได้แก่ H(Hue) ใช้ในการกำหนดสีต่าง ๆ S

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

(Saturation) คือค่าความอิ่มตัวของสี และ V (Value) คือ ค่าความสว่างของสีที่ต้องการ ในบางครั้งเรียกแบบจำลองของสีแบบ HSV ว่า แบบจำลองของสีแบบ HSB (HSB Color model, B = Brightness) แบบจำลองของสีชนิดนี้ถูกกำหนดในพิกัดทรงกระบอก (Cylindrical coordinate) โดยรูปแบบที่ใช้มาจากกรวยหกเหลี่ยม (Hexagonal cone) หรือรูปพีระมิดที่มี 6 ด้าน (6 – sided pyramid) ดังแสดงในรูปที่ 6.15 ส่วนบนสุดของกรวยหกเหลี่ยมจะมีค่า $V = 1$ ที่บริเวณนี้ของภาพจะประกอบด้วยสีที่มีค่าความสว่างมากที่สุดค่า H หรือค่าของสีต่าง ๆ จะถูกกำหนดโดยมุมรอบ ๆ แกนตั้ง โดยที่สีแดงอยู่ที่มุม 0 องศา สีเขียวอยู่ที่มุม 120 องศา และสีน้ำเงินอยู่ที่มุม 240 องศา

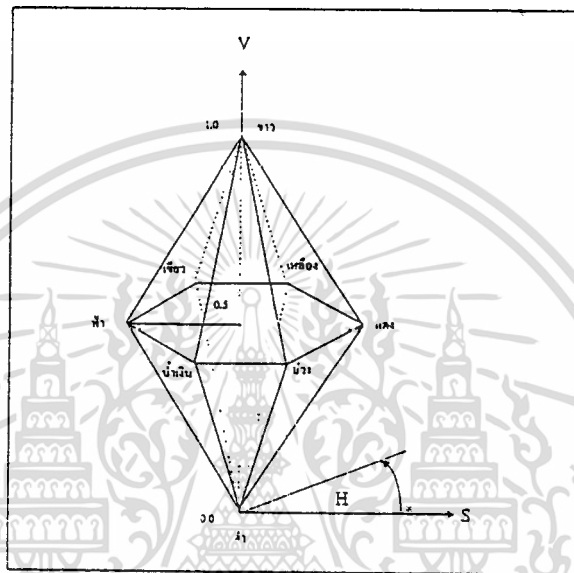


รูปที่ 6.15 แบบจำลองสีแบบ HSV

ค่าของความสว่าง หรือ V มีค่าระหว่าง 0 ที่ตำแหน่งกลางถึง 1 ที่บริเวณสามเหลี่ยมที่ประกอบเป็นรูปกรวยหกเหลี่ยม กรวยหกเหลี่ยมที่กำหนดขึ้นมีขนาดความสูง 1 หน่วย (V) โดยมีจุดยอดของรูปกรวยอยู่ที่บริเวณจุดกำเนิด ที่จุดยอดของรูปกรวยจะแทนที่โดยสีดำและมีค่า $V = 0$ ที่จุดนี้ค่าของ H และ S จะไม่มีผลต่อค่าของสีเลย และที่ $S = 0, V = 1$ จะแทนที่โดยสีขาว ที่ $S = 0.8$ คือค่าในแนวแกน V ที่มีค่าระหว่าง 0 และ 1 นั้น จะแทนที่โดยระดับของสีเทา โดยที่ค่าของ H จะไม่มีผลต่อสีเช่นกัน ดังนั้นเมื่อ S มีค่าไม่เท่ากับ 0 แล้วค่าของ H จะเป็นค่าที่กำหนดสีต่าง ๆ เช่น สีแดงบริสุทธิ์ (pure red) จะมีค่า $H = 0, S = 1$, และ $V = 1$ ดังนั้นที่ตำแหน่ง $S = 1, V = 1$ ค่าของสีที่ได้จะเป็นสีที่มีความบริสุทธิ์ (pure color) การเพิ่มสีขาวในสีต่าง ๆ ทำได้โดยการเปลี่ยนแปลงค่าของ S และระดับ (Shade) ของสีต่าง ๆ ทำได้โดยการเปลี่ยนแปลงค่าของ V

6.2.2.5 แบบจำลองของสีแบบ HLS

แบบจำลองของสีแบบ HLS (Hue, Lightness, Saturation) กำหนดขึ้นโดยอาศัยรูปกรวยหกเหลี่ยม 2 รูป ที่ซ้อนกันในระบบพิกัดทรงกระบอก ดังรูปที่ 6.16



รูปที่ 6.16 แบบจำลองสีแบบ HLS

H คือค่าของมุมรอบแกนตั้งของรูปหกเหลี่ยมที่ซ้อนกัน โดยมีสีแดงที่ค่ามุม 0 องศา เหมือนกับค่าของ H ในแบบจำลองของสีแบบ HSV และค่าของ S จะอยู่ที่บริเวณแกนนอน มีค่าอยู่ระหว่าง 0 ที่จุดกำเนิด และ 1 ที่ผิวของรูปหกเหลี่ยม L จะมีค่าเท่ากับ 0 ที่บริเวณจุดปลายของรูปหกเหลี่ยมรูปที่ต่ำกว่าซึ่งมีค่าเป็นสีดำ และจะมีค่าเท่ากับ 1 ที่บริเวณจุดปลายของรูปหกเหลี่ยมรูปบนซึ่งมีค่าเป็นสีขาว ระดับของสีเทาจะอยู่ที่บริเวณที่ S มีค่าเท่ากับ 0 และค่าสีที่มีความอิ่มตัวมากที่สุดจะอยู่ที่บริเวณ $S = 1, L = 0.5$

6.2.3 ทฤษฎีการกำหนดความเข้มของแสง

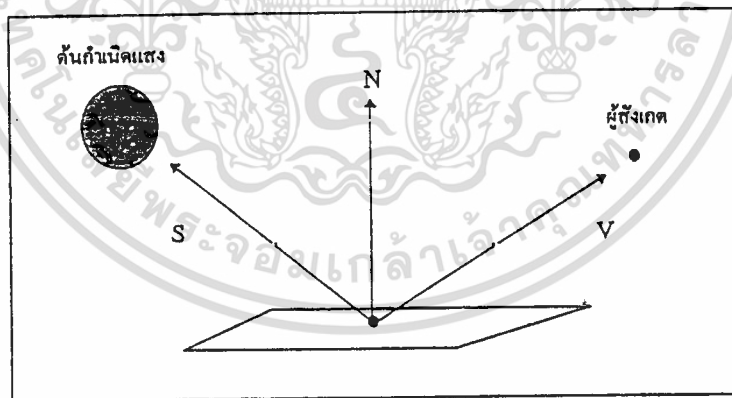
วัตถุใด ๆ ก็ตามที่มีแสงจากแหล่งกำเนิดใด ๆ มาตกกระทบ จะทำให้เกิดระดับความเข้มของแสงที่สะท้อนมีค่าต่าง ๆ กัน ความเข้มของแสงที่สะท้อนนั้นจะขึ้นอยู่กับลักษณะของแสงที่มา

จากต้นกำเนิดและคุณสมบัติบางประการของวัตถุเอง ทฤษฎีการกำหนดความเข้มของแสงที่พื้นผิวของวัตถุเมื่อมีแสงมาตกกระทบ สามารถแยกได้เป็น

6.2.3.1 การกำหนดความเข้มของแสงแบบคงที่ (Constant Shading)

การกำหนดความเข้มของแสงแบบคงที่เป็นวิธีการกำหนดความเข้มของแสงที่สะดวกและง่ายที่สุด เนื่องจากในวิธีนี้ จะคำนวณค่าความเข้มแสงของแสงที่สะท้อนจากพื้นผิวเพียงครั้งเดียว เพราะถือว่า ค่าความเข้มของแสงที่สะท้อนจากพื้นผิวของวัตถุนั้นมีค่าเท่ากันเสมอ วิธีการนี้จะต้องอยู่ภายใต้ข้อกำหนดที่ว่า

1. ต้นกำเนิดแสงอยู่ที่ infinity ดังนั้น เวกเตอร์ $N \cdot L$ จะมีค่าคงที่ทุกตำแหน่งบนพื้นผิวของวัตถุ เมื่อเวกเตอร์ N คือ เวกเตอร์ที่ตั้งฉากของพื้นผิว และ เวกเตอร์ L คือทิศทางของต้นกำเนิดแสง ดังแสดงในรูปที่ 6.17
2. ผู้สังเกตอยู่ที่อนันต์ (infinity) ดังนั้น เวกเตอร์ $N \cdot V$ จะมีค่าคงที่ทุกตำแหน่งบนพื้นผิวของวัตถุ เมื่อเวกเตอร์ V คือ เวกเตอร์ของผู้สังเกต
3. พื้นผิวของวัตถุจะต้องเป็นวัตถุผิวเรียบ



รูปที่ 6.17 ทิศทางการสะท้อนของแสง

จากข้อกำหนดทั้ง 3 ทำให้ค่าความเข้มของแสงจากสมการที่ 36 มีค่าคงที่เสมอทุกตำแหน่งของพื้นผิวของวัตถุ ในกรณีที่วัตถุประกอบมาจากรูปหลายเหลี่ยม (polygon) หลายภาพนั้น วิธีการจะทำให้เกิดความไม่ต่อเนื่องของความเข้มของแสงที่บริเวณรอยต่อของรูปหลายเหลี่ยมแต่ละรูป

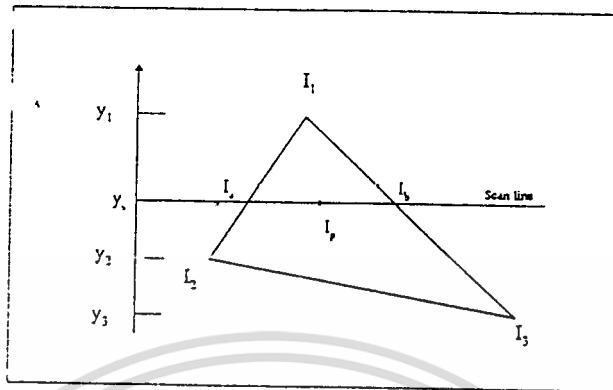
ดังแสดงในรูปที่ 6.18 บางครั้งเรียกวิธีการกำหนดความเข้มของแสงแบบนี้ว่า การกำหนดความเข้มของแสงแบบเรียบ (flat shading)



รูปที่ 6.18 วิธี Constant Shading

6.2.3.2 การกำหนดความเข้มของแสงแบบ Gouraud

วิธีการกำหนดความเข้มของแสงโดยวิธีของ Gouraud หรือ interpolation shading นี้จะทำให้ไม่มีผลของความไม่ต่อเนื่องกันของความเข้มของแสงที่บริเวณรอยต่อของรูปหลายเหลี่ยมที่ประกอบเป็นวัตถุ มีหลักการคือ การกำหนดความเข้มของแสงที่ตำแหน่งต่าง ๆ บนพื้นผิวของรูปหลายเหลี่ยมจะอาศัยค่าเฉลี่ยของค่าความเข้มของแสงจากค่าความเข้มของแสงที่จุดปลาย (vertex) ต่าง ๆ ของรูปหลายเหลี่ยม ดังนั้นสิ่งที่สำคัญที่สุดคือ จะต้องทราบเวกเตอร์ที่ตั้งฉาก (normal vector) ที่จุดปลายต่าง ๆ ของรูปหลายเหลี่ยม เพื่อใช้ในการคำนวณค่าความเข้มของแสง เมื่อทราบค่าความเข้มที่จุดปลายต่าง ๆ ของรูปหลายเหลี่ยมแล้วจึงสามารถหาค่าความเข้มของแสงที่ตำแหน่งใด ๆ บนรูปหลายเหลี่ยมได้โดยอาศัยวิธีการของ Scan line เข้ามาช่วยดังแสดงในรูปที่ 6.19



รูปที่ 6.19 การหาค่าความเข้มของแสงโดยวิธี Gouraud

จากรูปที่ 6.19 ค่าความเข้มที่จุด P สามารถหาได้โดย

I_1 = ความเข้มของแสงที่จุดปลายที่ 1

I_2 = ความเข้มของแสงที่จุดปลายที่ 2

I_3 = ความเข้มของแสงที่จุดปลายที่ 3

I_a = ความเข้มของแสงที่จุดปลายที่ a

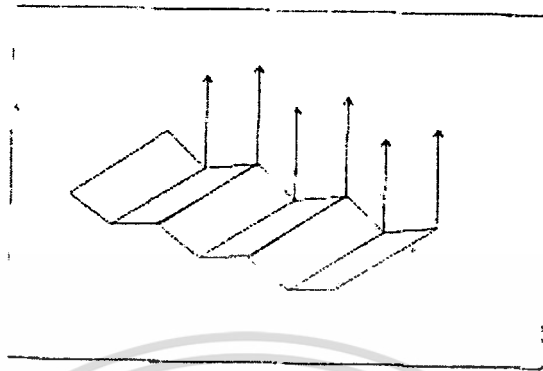
I_b = ความเข้มของแสงที่จุดปลายที่ b

I_p = ความเข้มของแสงที่จุดปลายที่ p

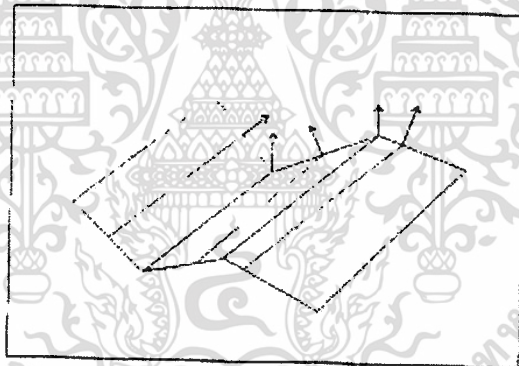
6.2.3.3 การกำหนดความเข้มของแสงแบบ Phong

การกำหนดความเข้มของแสงแบบ Phong หรือ normal-vector interpolation shading มีวิธีการคำนวณคล้ายกับวิธีการของ Gouraud Shading แต่แตกต่างกันที่ในวิธีของ Phong shading นั้น อาศัยเวกเตอร์ที่ตั้งฉากมาใช้ในการคำนวณค่าโดยประมาณของความเข้มของแสงบนวัตถุแทนการใช้ความเข้มของแสงตามแบบ Gouraud จากรูปที่ 47 การหาค่าความเข้มของแสงที่จุด P สามารถกระทำได้โดยการหาค่าเวกเตอร์ที่ตั้งฉาก ที่บริเวณจุดยอดของรูปหลายเหลี่ยมก่อน และค่าของเวกเตอร์ที่ตั้งฉากที่จุด A และ B นั้นสามารถประมาณได้จากค่าของเวกเตอร์ที่ตั้งฉากที่จุด 1, 2 และจุด 1, 3 โดยวิธีเดียวกันค่าเวกเตอร์ที่ตั้งฉากที่จุด P ก็จะสามารถประมาณได้จากเวกเตอร์ที่ตั้งฉากที่จุด a และ b ซึ่งเวกเตอร์ที่ตั้งฉากที่จุด p นี้จะทำให้สามารถนำไปคำนวณหาความเข้มของแสงที่จุด p ได้รูปที่ 6.20 และ 6.21 แสดงพื้นผิวที่ใช้ในการกำหนดความเข้มของแสงแบบ Gouraud และ

Phong ตามลำดับ



รูปที่ 6.20 การกำหนดความเข้มของแสงแบบ Gouraud



รูปที่ 6.21 การกำหนดความเข้มของแสงแบบ Phong

จากรูปที่ 6.20 เมื่อทำการกำหนดค่าความเข้มของแสงโดยวิธี Gouraud ของพื้นผิวดังกล่าว ค่าความเข้มที่ได้จะมีค่าเท่ากันเนื่องจากค่าเฉลี่ยของค่าเวกเตอร์ที่ตั้งฉากจะมีค่าเท่ากัน แต่ในกรณีของการกำหนดค่าความเข้มของแสงโดยวิธี Phong ดังรูปที่ 6.21 นั้นพื้นที่ที่กำหนดจะถูกแบ่งออกเป็นพื้นที่ย่อย ๆ ทำให้ค่าของเวกเตอร์ตั้งฉากที่จุดยอดของพื้นที่มีค่าแตกต่างกันด้วย ดังนั้นค่าความเข้มของแสงของพื้นผิวที่ได้จะมีค่าแตกต่างกัน

สรุปแสง สี และ การจัดลำดับความเข้มของแสงเป็นวิธีหนึ่งที่ทำให้ภาพจำลองที่ได้มีลักษณะเหมือนจริงมากขึ้น เนื่องจากค่าความเข้มของแสงที่แตกต่างกันที่ปรากฏบนวัตถุจะทำให้สามารถที่จะมองเห็นลักษณะรูปร่างของวัตถุได้ดีขึ้น จากรูปที่ 6.20 เมื่อทำการกำหนดค่าความเข้มเอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ของแสงโดยวิธี Gouraud ของพื้นผิวดังกล่าว ค่าความเข้มที่ได้จะมีค่าเท่ากันเนื่องจากค่าเฉลี่ยของค่าเวกเตอร์ที่ตั้งฉากจะมีค่าเท่ากัน แต่ในกรณีของการกำหนดค่าความเข้มของแสงโดยวิธี Phong ดังรูปที่ 6.21 นั้นพื้นที่ที่กำหนดจะถูกแบ่งออกเป็นพื้นที่ย่อย ๆ ทำให้ค่าของเวกเตอร์ตั้งฉากที่จุดยอดของพื้นที่มีค่าแตกต่างกันด้วย ดังนั้นค่าความเข้มของแสงของพื้นผิวที่ได้จะมีค่าแตกต่างกัน

สรุป แสง สี และ การจัดลำดับความเข้มของแสงเป็นวิธีหนึ่งที่ทำให้ภาพจำลองที่ได้มีลักษณะเหมือนจริงมากขึ้น เนื่องจากค่าความเข้มของแสงที่แตกต่างกันที่ปรากฏบนวัตถุจะทำให้สามารถที่จะมองเห็นลักษณะรูปร่างของวัตถุได้ดีขึ้น



บทที่ 7

โปรแกรม OpenGL

7.1 เบื้องต้นเกี่ยวกับ OpenGL

OpenGL ถูกออกแบบมาเป็น software ที่ติดต่อกับ hardware ด้าน graphics โดยตัว OpenGL จะเป็นไลบรารีของแบบจำลองและกราฟิกวัตถุ 3 มิติที่นำไปใช้ได้อย่างรวดเร็ว ในการใช้ OpenGL ทำให้สามารถสร้างกราฟิกวัตถุ 3 มิติที่สวยงามใกล้เคียงกับการใช้ ray tracer แต่เร็วกว่ามาก

OpenGL นำไปใช้มากขึ้นทั้งเครื่องคอมพิวเตอร์ที่ออกแบบมาไว้แสดงและประมวลผลภาพ 3 มิติและทั้งเครื่องที่เพิ่ม 3D graphics card

OpenGL ถูกนำไปใช้หลายทาง เช่น CAD engineering การสร้างตัวโคโนเสาร์จำลองด้วยเครื่องคอมพิวเตอร์ ในอนาคตภาพ 3 มิติจะพบได้ทั่วไปในงานธุรกิจและงานทั่วไป ไม่เฉพาะงานวิทยาศาสตร์และเกมเหมือนเดิม

7.2 หลักการทำงานเบื้องต้น

OpenGL เป็นภาษากาแฟที่มีลักษณะแบบชุดคำสั่งมากกว่าการบรรยาย หมายถึง แทนที่จะกำหนดหน้าจอแล้วบรรยายสิ่งที่จะแสดง ผู้ใช้สามารถเพียงบรรยายขั้นตอนที่จะทำให้เกิดสิ่งนั้น ซึ่งขั้นตอนจะเป็นตัวเรียก API ประมาณ 120 คำสั่งและฟังก์ชันรวมกัน ขั้นตอนทำให้เกิดภาพหน่วยย่อย คือ จุด เส้น และรูปหลายเหลี่ยม อีกทั้งยังช่วยการให้แสงและเงา การให้ลายพื้นผิว การเคลื่อนไหวและเทคนิคอื่นๆ

OpenGL ไม่ได้รวมถึงฟังก์ชันการจัดการเกี่ยวกับวินโดว์ การตอบสนองกับผู้ใช้ การจัดการเกี่ยวกับไฟล์ เพราะระบบปฏิบัติการผู้ใช้จะมีฟังก์ชันเฉพาะไว้ทำหน้าที่เหล่านี้อยู่แล้ว

OpenGL เมื่อออกใช้ครั้งแรกทำงานบน windows NT 3.5 ดังนั้นการนำมาใช้กับ windows 95 ต้องมีไฟล์ DLL ช่วยสนับสนุน

7.3 สถาปัตยกรรมกราฟิก

การวาดรูปด้วย OpenGL จะไม่เหมือนกับการวาดด้วย GDI ทั้งหมด การเลือกปากกา แปรง ตัวอักษร และวัตถุ GDI อื่นๆ จะไม่มีผลกระทบกับ OpenGL เพราะในขณะที่ GDI ใช้ device context ควบคุมการวาดในหน้าต่าง OpenGL จะใช้ rendering context ซึ่งรวบรวมจาก device context ซึ่งก็คือรวบรวมโดยหน้าต่างอีกที แสดงว่า OpenGL วาดในหน้าต่างเลย

OpenGL มีข้อจำกัด คือ ไม่มีการสนับสนุนโดยตรงในการพิมพ์รูปวาดทางเครื่องพิมพ์ monochrome หรือเครื่องพิมพ์สีที่มีน้อยกว่า 16 สี การที่ windows มี hardware palette เพียงหนึ่ง

เอกสารนี้เป็นเอกสารลิขสิทธิ์สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้เผยแพร่โดยไม่ได้รับอนุญาต

ไม่ว่าการณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เดียวซึ่งไม่เพียงพต่อการวาดหน้าต่าหลายรูปแบบ และการที่ OpenGL เพิ่มความสามารถพิเศษบางประการ โดยที่ hardware บางอันไม่ได้เพิ่มความสามารถนั้นเข้ามาด้วย เช่น ภาพ stereoscopic หน่วยความจำช่วย



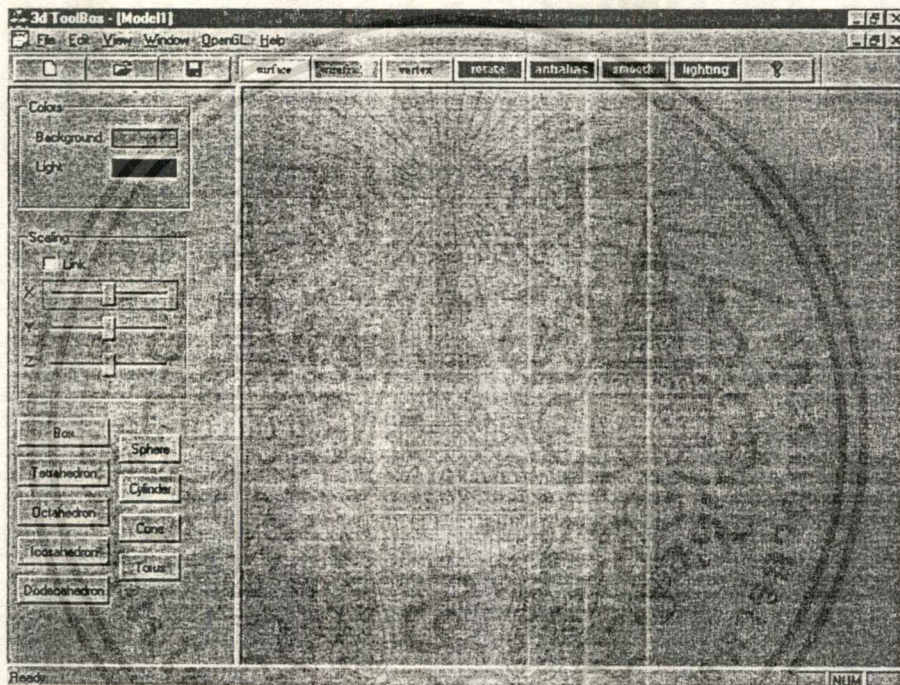
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 8

ผลการทดลอง

8.1 ผลการทำงานของโปรแกรมสร้างภาพ 3 มิติ

หลังจากสั่งให้โปรแกรมทำงานจะได้หน้าจอเป็นดังรูปที่ 8.1



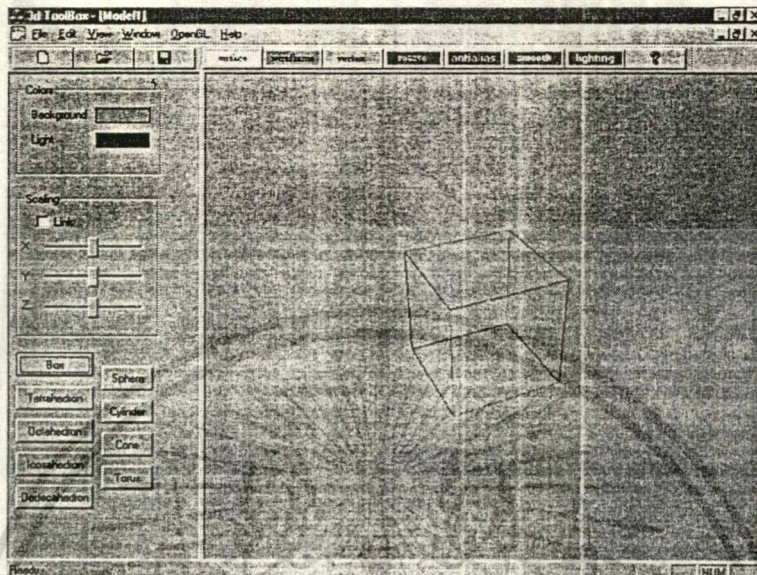
รูปที่ 8.1 หน้าจอของโปรแกรมสร้างภาพ 3 มิติ

ซึ่งสามารถเลือกชนิดวัตถุได้ทั้งหมด 9 ชนิด คือ

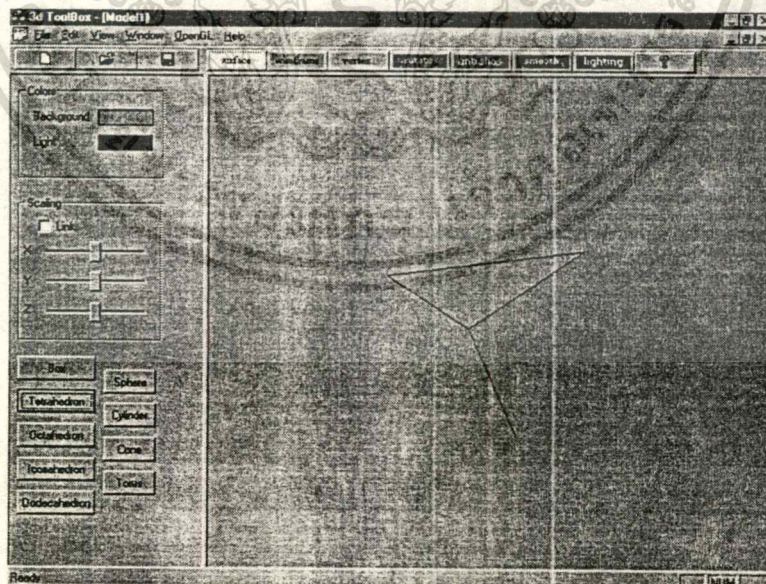
1. กล่อง (box)
2. รูปทรง 4 หน้า (tetrahedron)
3. รูปทรง 8 หน้า (octahedron)
4. รูปทรง 20 หน้า (icosahedron)
5. รูปทรง 12 หน้า (dodecahedron)
6. ทรงกลม (sphere)
7. ทรงกระบอก (cylinder)
8. ทรงกรวย (cone)

9. ทรงทอรัส (torus) ไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โดยผู้จะใช้จะเห็นในรูปแบบ wireframe ซึ่งเป็นค่า default ดังแสดงในรูปที่ 8.2 – รูปที่ 8.10

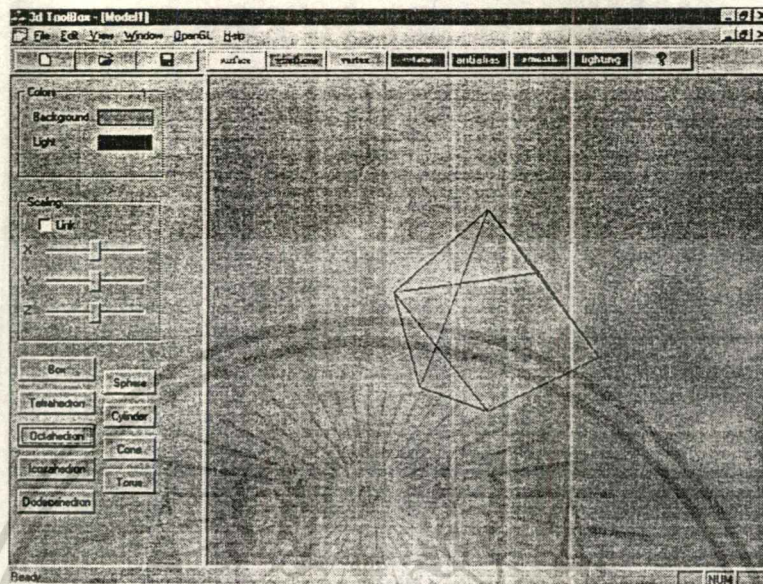


รูปที่ 8.2 wireframe ของ box

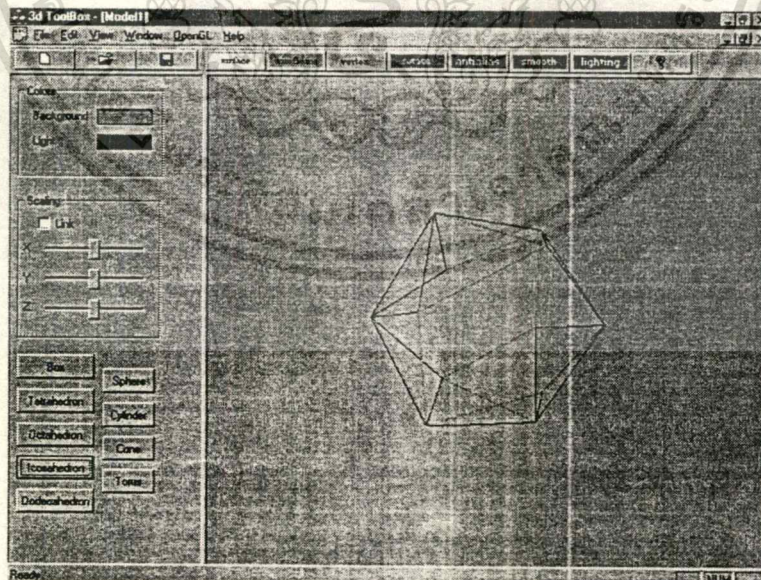


รูปที่ 8.3 wireframe ของ tetrahedron

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

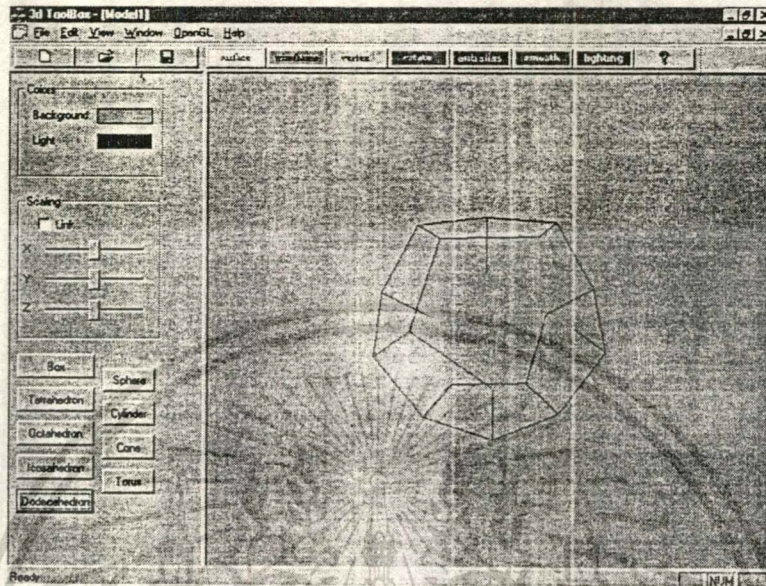


รูปที่ 8.4 wireframe ของ octahedron

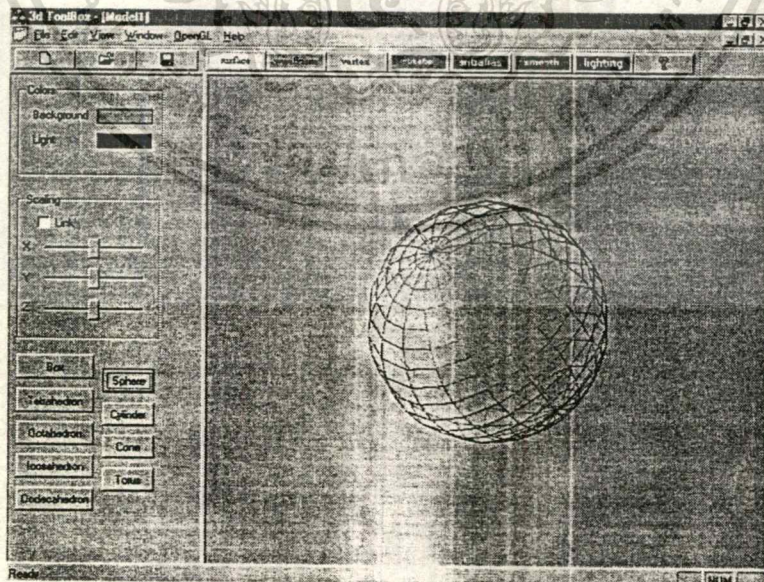


รูปที่ 8.5 wireframe ของ icosahedron

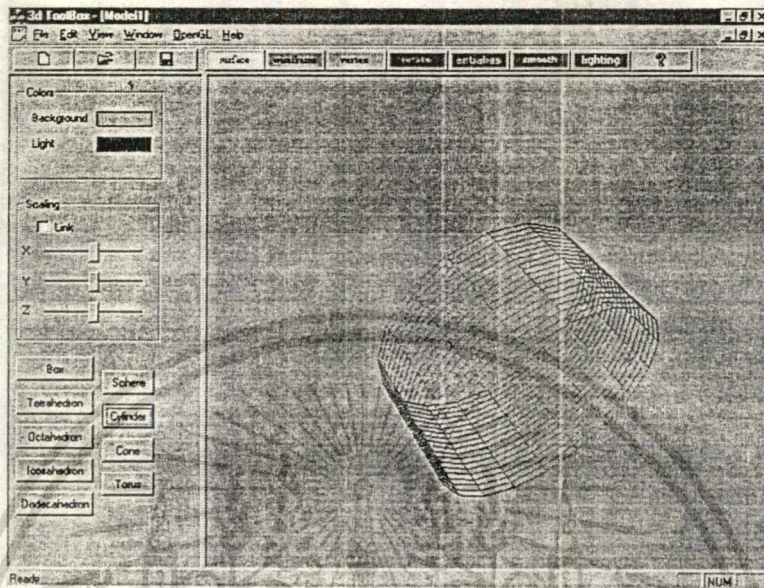
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



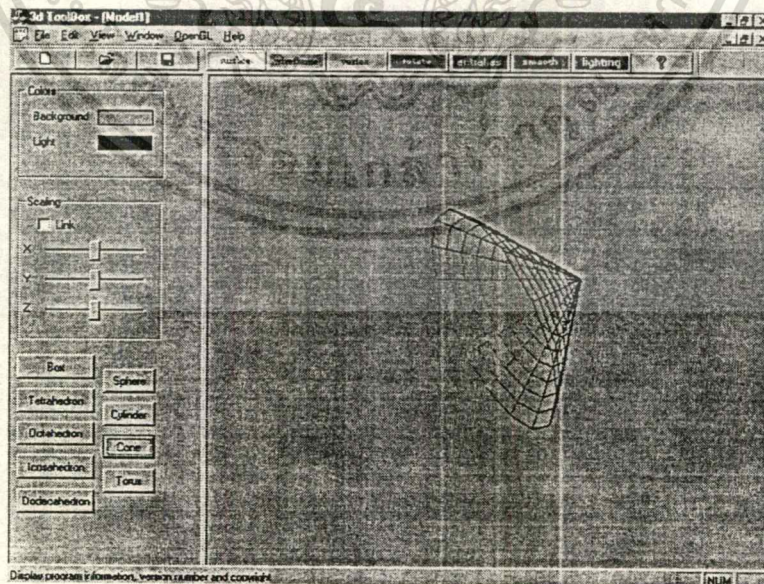
รูปที่ 8.6 wireframe ของ dodecahedron



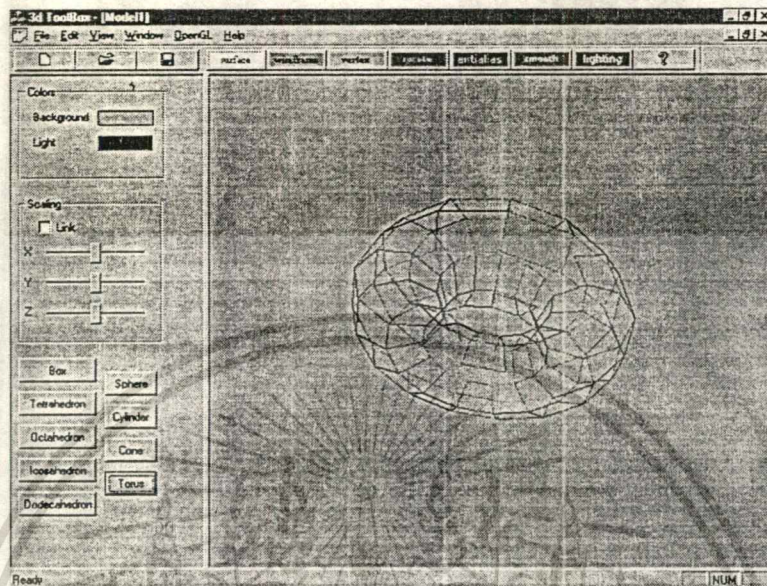
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับรูปที่ 8.7 wireframe ของ sphere เพื่อการศึกษาเท่านั้น ไม่ควรนำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 8.8 wireframe ของ cylinder



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการรูปที่ 8.9 wireframe ของ cone อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

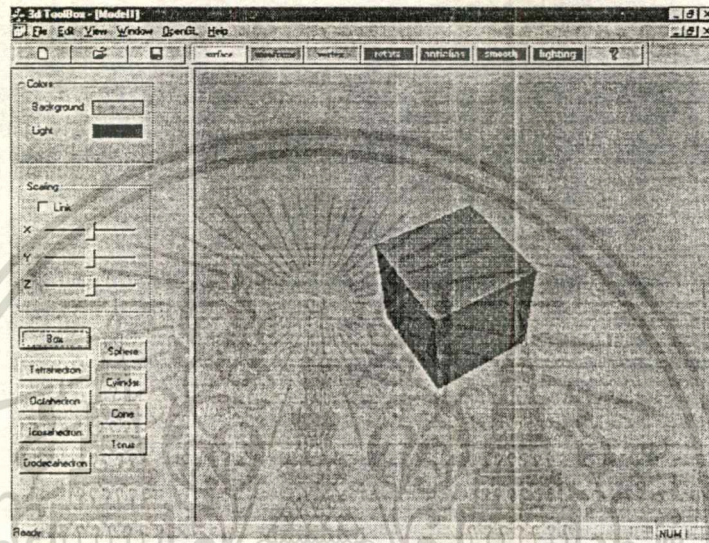


รูปที่ 8.10 wireframe ของ torus

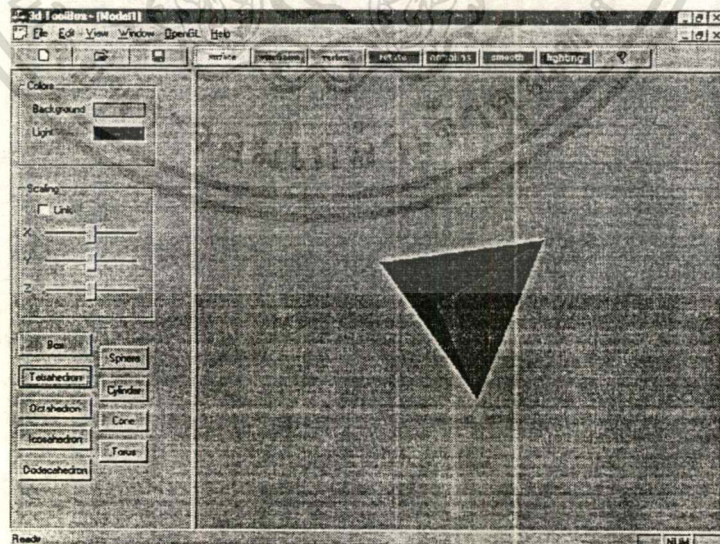
ผู้ใช้สามารถเปลี่ยนรูปแบบการแสดงผลวัตถุได้ 3 รูปแบบคือ

1. แบบพื้นผิว (surface)
2. แบบโครงตาข่าย (wireframe)
3. แบบจุดมุมของรูปหลายเหลี่ยม (vertex)

ในการแสดงวัตถุแบบมีจุดมุมของรูป polygon จะเห็นได้ไม่ชัดเจนนักจึงไม่ได้รวมไว้ในที่นี้ ส่วนการแสดงผลผิวของวัตถุ 3 มิติ สามารถแสดงได้ทั้งแบบปกติดังในรูปที่ 8.11-รูปที่ 8.19 และแบบละเอียดดังในรูปที่ 8.20 – รูปที่ 8.23 ซึ่งวัตถุทั้งหมดได้ให้แสงไว้ด้วย

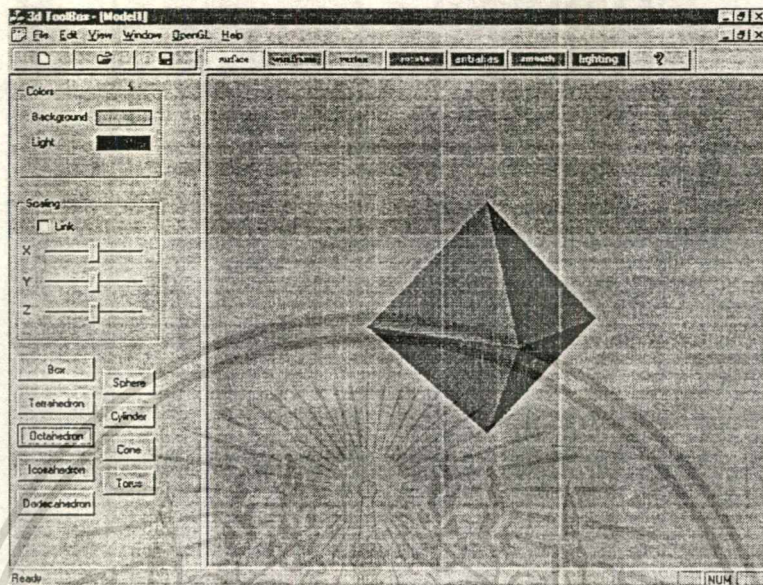


รูปที่ 8.11 surface ของ box

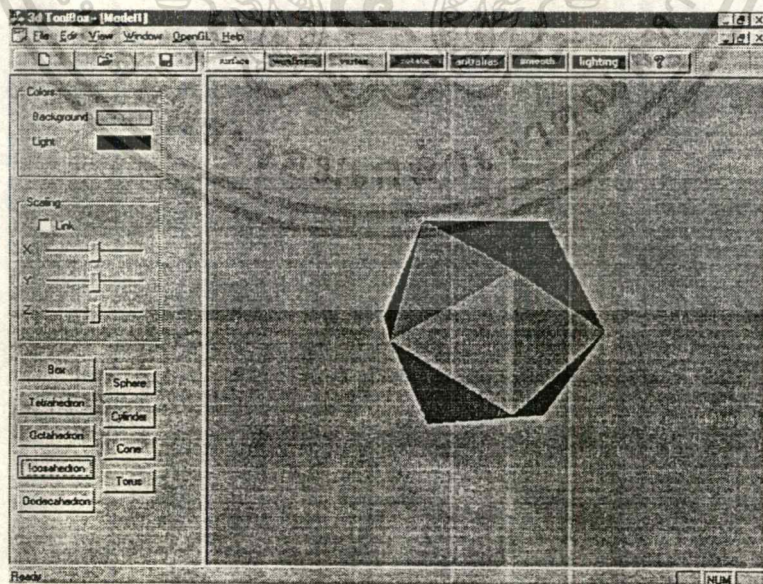


รูปที่ 8.12 surface ของ tetrahedron

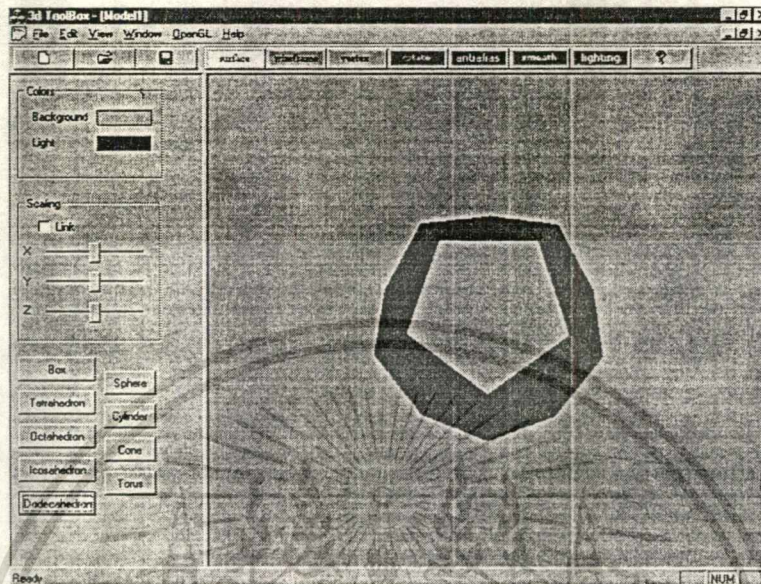
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



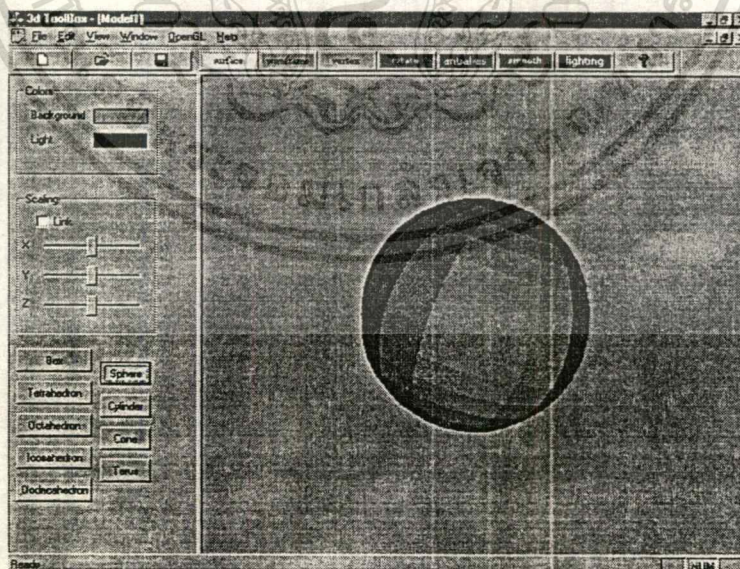
รูปที่ 8.13 surface ของ octahedron



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับรูปที่ 8.14 ที่ surface ของ icosahedron ชาติให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

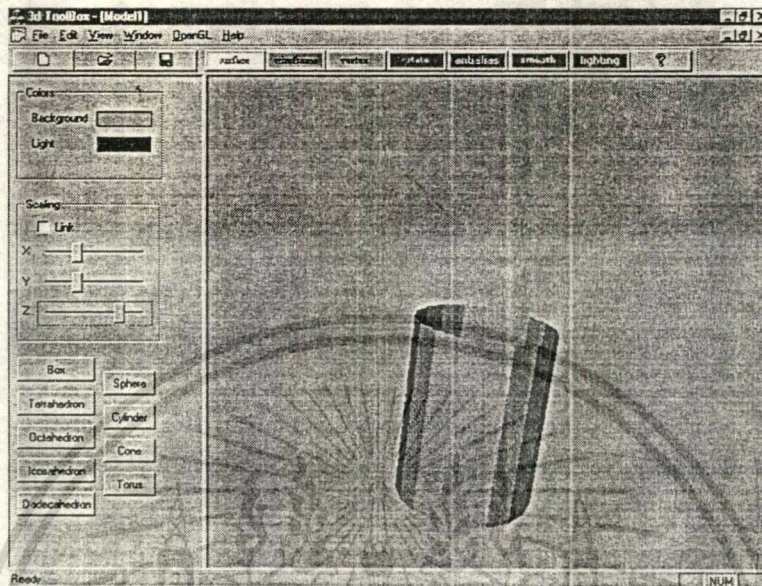


รูปที่ 8.15 surface ของ dodecahedron

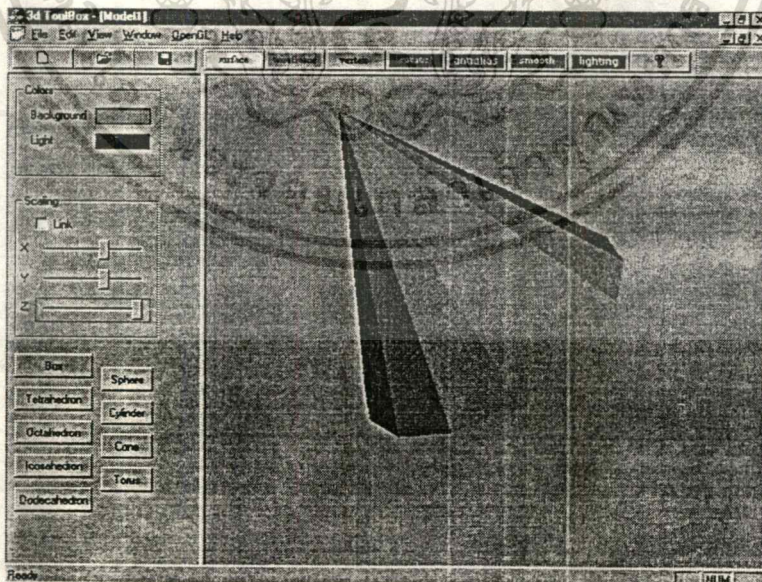


รูปที่ 8.16 surface ของ sphere

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

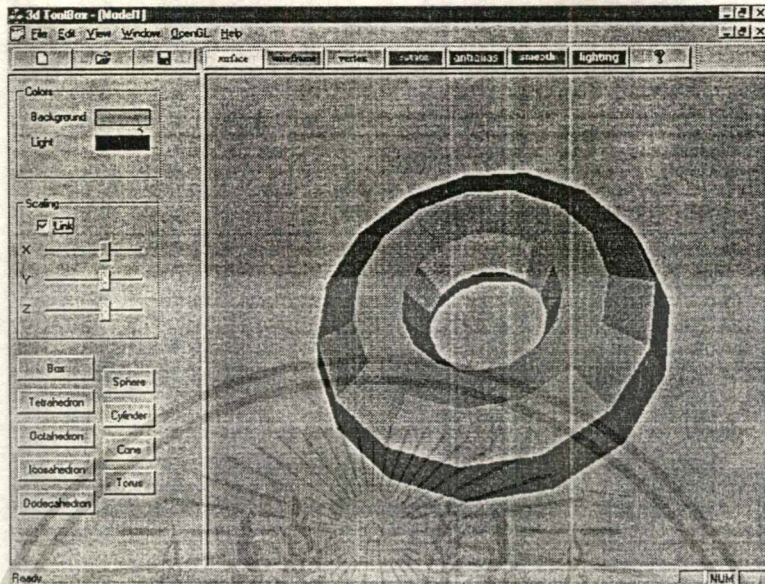


รูปที่ 8.17 surface ของ cylinder



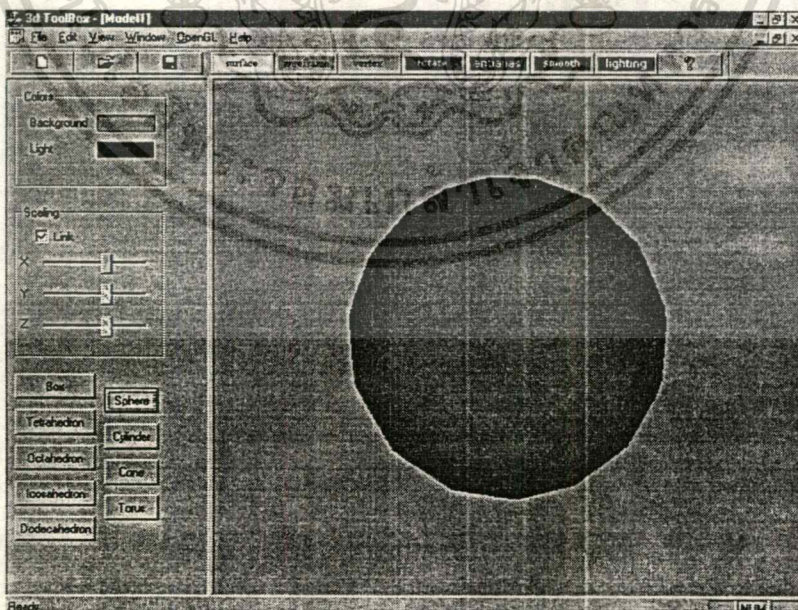
รูปที่ 8.18 surface ของ cone

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

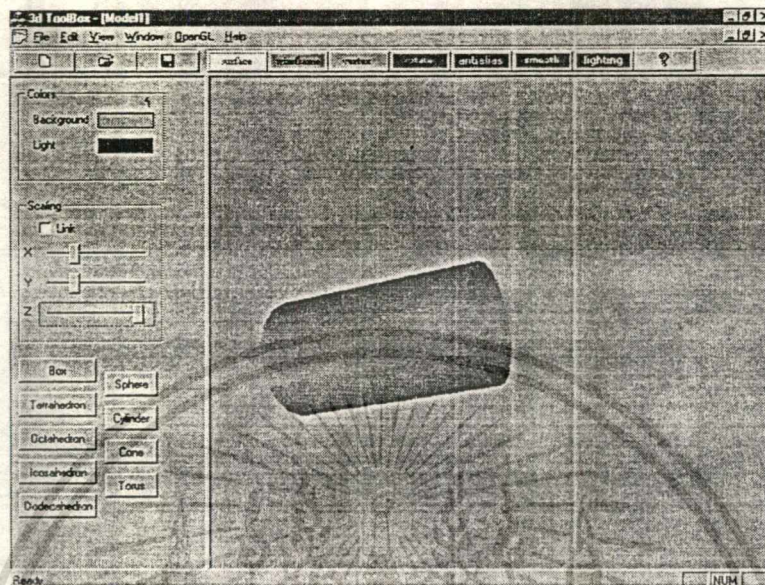


รูปที่ 8.19 surface ของ torus

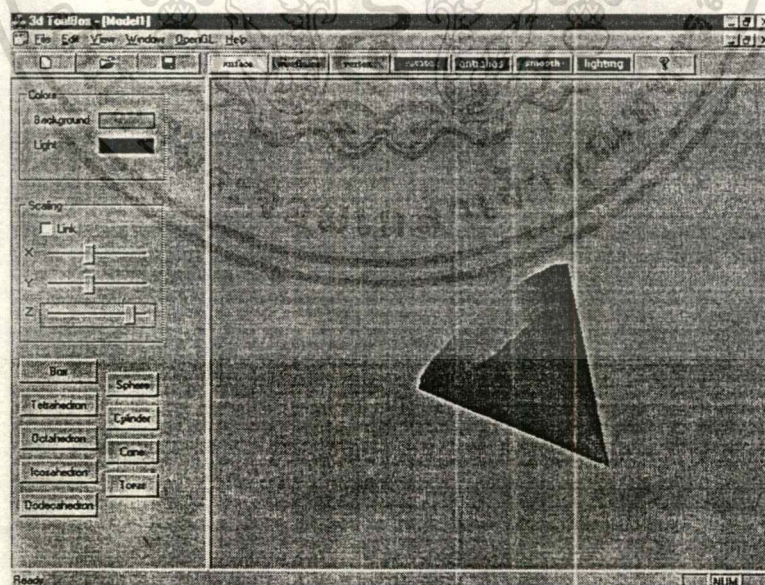
ในการแสดงวัตถุแบบพื้นผิวสามารถเพิ่มความเรียบได้ ดังแสดงในรูปต่อไป



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับรูปที่ 8.20 smooth surface ของ sphere ให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

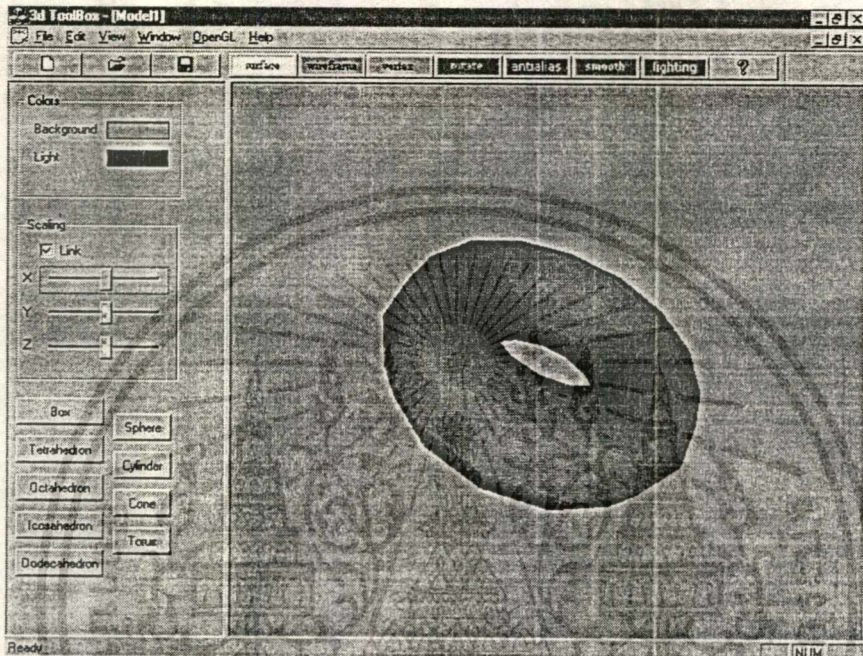


รูปที่ 8.21 smooth surface ของ cylinder



รูปที่ 8.22 smooth surface ของ cone

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 8.23 smooth surface ของ torus

ส่วนวัตถุนิคอื่นๆ เมื่อทำการ smooth surface แล้วจะไม่เปลี่ยนแปลงไปจากเดิม จึงไม่แสดงไว้ในที่นี้ วัตถุ 3มิติต่างๆสามารถแสดงได้หลายขนาดด้วยการปรับ scale ที่ scroll bar ทั้งในแนวแกน x แกน y แกน z (ทีละแกนและพร้อมกันด้วยการเลือก link) รวมทั้งการหมุนวัตถุเพื่อเปลี่ยนมุมมอง ดังสังเกตคุณสมบัติทั้งสองอย่างจากรูปของวัตถุเดียวกันที่ต่างผลที่ปรากฏ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

8.2 ผลการทำงานของโปรแกรมควบคุมแขนกล

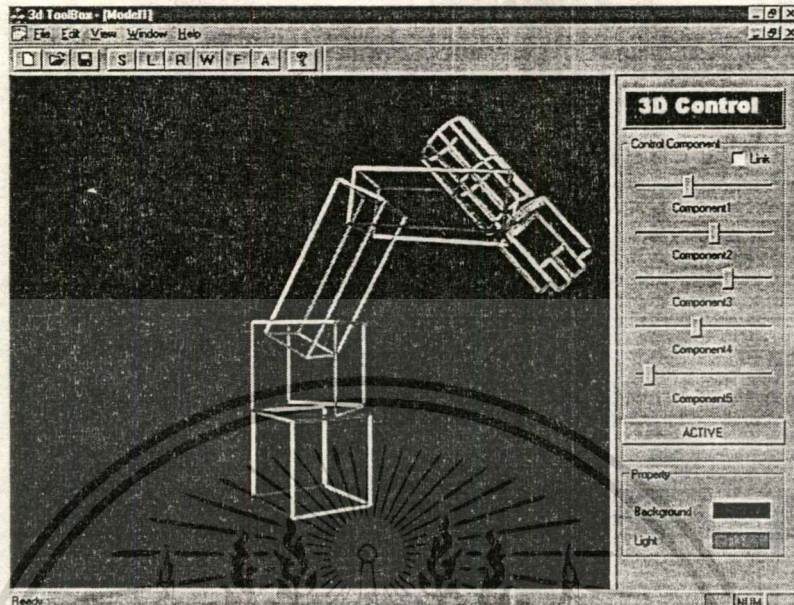
โปรแกรมควบคุมแขนกลนี้เขียนขึ้นมาเพื่อนำมาใช้ประโยชน์ในการมองเห็นภาพการทำงานของเครื่องมือที่มีการเคลื่อนที่ไปสู่ตำแหน่งต่าง ๆ ที่ต้องการซึ่งมีการควบคุมส่วนหนึ่งมาจากผู้ใช้ผ่านเครื่องคอมพิวเตอร์ โดยตัวแขนกลนี้ประกอบไปด้วยชิ้นส่วน 5 ชิ้น คือ

- Component 1 ทำหน้าที่ควบคุมการหมุนของแขนกลซึ่งเป็นส่วนที่ต่ออยู่บนส่วนฐาน
- Component 2 ทำหน้าที่ควบคุมการเคลื่อนที่ขึ้นลงของแขนกลตอนที่ 2
- Component 3 ทำหน้าที่ควบคุมการเคลื่อนที่ขึ้นลงของแขนกลตอนที่ 3
- Component 4 ทำหน้าที่ควบคุมการเคลื่อนที่ขึ้นลงของแขนกลตอนที่ 4
- Component 5 ทำหน้าที่ควบคุมการหมุนของแขนกลส่วนที่เป็นส่วนจับวัตถุของแขนกล

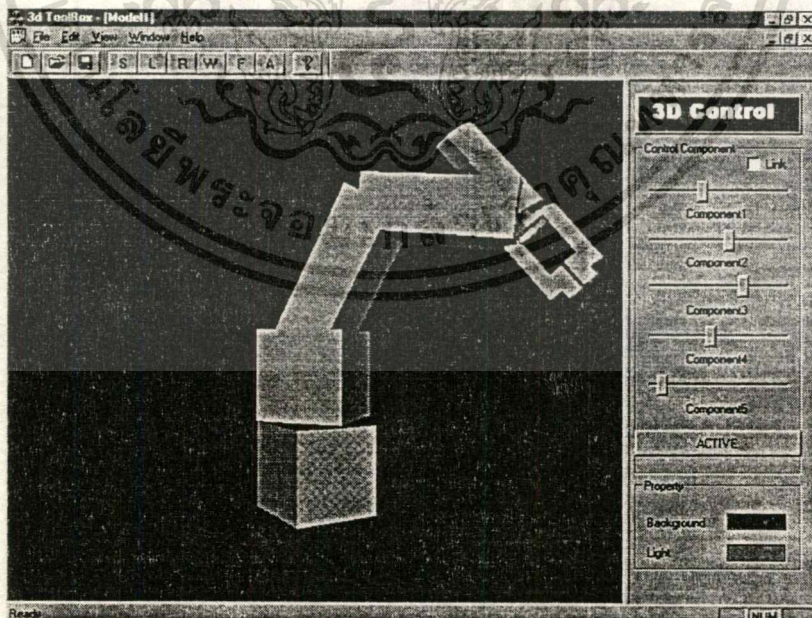
ปุ่มควบคุมการลักษณะแสดงรูปแขนกล

- Background ทำหน้าที่เปลี่ยนสีพื้นหลังของจอภาพ
- Light ทำหน้าที่เปลี่ยนสีของตัวแขนกล
- ปุ่ม S ทำหน้าที่เพิ่มหรือลดความเร็วของตัวแขนกล
- ปุ่ม L ทำหน้าที่ใส่หรือไม่ใส่แสงให้กับแขนกล (ทำงานแบบ Toggle)
- ปุ่ม R ทำหน้าที่หมุนหรือไม่หมุนตัวแขนกล
- ปุ่ม W ทำหน้าที่วาดแขนกลให้เป็นแบบ wireframe
- ปุ่ม F ทำหน้าที่ใส่ผิวให้กับวัตถุ
- ปุ่ม A ทำหน้าที่เพิ่มหรือไม่เพิ่มความหนาของเส้นที่เป็นแบบ wireframe ของแขนกล
- ปุ่ม Active ทำหน้าที่ใช้ส่งข้อมูลโปรแกรมนี้ผ่านไปตามสายอนุกรม

เมื่อสั่งให้โปรแกรมทำงานจะได้หน้าจอซึ่งแสดงภาพแขนกลและส่วนควบคุมแขนกล โดยสามารถเลือกที่จะแสดงเป็นรูปแบบ wireframe หรือ surface ได้ดังรูป



รูปที่ 8.24 หน้าจอแสดงแขนกลแบบ wireframe



รูปที่ 8.25 หน้าจอแสดงแขนกลแบบ surface

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรมที่ 2 นี้สามารถส่งข้อมูลออกจากพอร์ตอนุกรมได้ โดยใช้โปรแกรมใดๆที่สามารถรับข้อมูลแบบอนุกรมมาทดสอบการส่งข้อมูล ปรากฏว่า ส่งข้อมูลได้จริง แต่ยังไม่สามารถคำนวณค่าที่จะสั่งให้มอเตอร์หมุนไปตำแหน่งเดียวกับที่เห็นในหน้าจอ



บทที่ 9

สรุปและวิจารณ์

ปริญญานิพนธ์ฉบับนี้ ในบทแต่ละบทนั้นเราได้กล่าวถึงส่วนของทฤษฎีต่าง ๆ ที่เกี่ยวข้องกับรายละเอียดที่ใช้ในการประมวลผลภาพเสมือนจริงแบบ 3 มิติ เช่น การย้ายแกน การเปลี่ยนขนาด การหา normal vector การทำเทคนิค gouraud และ ray tracing โดยในบทหลัง ๆ จะเป็นส่วนของโปรแกรม ซึ่งเราใช้โปรแกรม visual c++ ของบริษัท Microsoft และ OpenGL ของบริษัท siligon graphics ในการเขียนโปรแกรม ซึ่งเราได้สร้างขึ้น 2 โปรแกรมดังนี้

1. โปรแกรมสร้างภาพ 3 มิติ เขียนขึ้นเพื่อแสดงให้เห็นถึงความสามารถในการสร้างภาพรูปทรงเรขาคณิตได้หลายชนิด โดยโปรแกรมสามารถแก้ไขลักษณะในการมองของภาพที่แสดงได้ เช่น การมองเห็นเป็นแบบ wireframe แบบมีพื้นผิว โดยคำนึงถึงแสงเงา การทำ smooth shading หรือ flat shading นอกจากนี้ยังสามารถบันทึกรูปที่ได้เป็นไฟล์ได้
2. โปรแกรมการประยุกต์ใช้งาน เป็นโปรแกรมซึ่งแสดงถึงแนวทางในการนำเอาประโยชน์ในการสร้างภาพ 3 มิติบนคอมพิวเตอร์มาใช้ในการจำลองภาพเคลื่อนไหวแบบ 3 มิติได้ โดยโปรแกรมนี้อาจจำลองภาพแขนกลซึ่งเราสามารถควบคุมการเคลื่อนไหวของภาพได้ และยังสามารถเขียนส่วนของการเชื่อมต่อกับอุปกรณ์ภายนอกผ่านพอร์ตอนุกรมของคอมพิวเตอร์ไว้ในโปรแกรมด้วยเพื่อเป็นแนวทางให้ผู้สนใจได้ศึกษาต่อไป

ปัญหาในการทำปริญญานิพนธ์

เนื่องจากเรามีความต้องการในการแสดงให้เห็นถึงประโยชน์ที่เกิดจากการทำปริญญานิพนธ์ทำให้เราสร้างส่วนของฮาร์ดแวร์ขึ้นมาด้วย แต่เนื่องจากข้อจำกัดทางด้านเทคนิค การเงิน ความสนับสนุน ความช่วยเหลือ และเวลา ทำให้ไม่สามารถสร้างฮาร์ดแวร์ขึ้นมาประยุกต์ใช้ได้ทันที เราจึงแก้ปัญหาโดยใช้ซอฟต์แวร์เป็นตัวตรวจสอบการทำงานแทน

วิจารณ์

เนื่องจากความก้าวหน้าทางเทคโนโลยีไมโครอิเล็กทรอนิกส์ทำให้คอมพิวเตอร์มีความสามารถสูง การประมวลผลที่รวดเร็ว อุปกรณ์เชื่อมต่อที่หลากหลาย ราคาที่ถูกลง ประกอบกับซอฟต์แวร์ที่สร้างมาสนับสนุนการทำงานอย่างมากมาย ซึ่งสิ่งต่าง ๆ เหล่านี้ทำให้เกิดความตื่นตัวในการติดตามเทคโนโลยีคอมพิวเตอร์ ภาพกราฟิก 3 มิติบนคอมพิวเตอร์ก็เป็นส่วนหนึ่งที่ได้รับ

ความนิยมเช่นกัน เนื่องมาจากความสามารถในการสื่อสารกับผู้ใช้ให้เกิดความเข้าใจได้ดี รวมทั้งความสวยงามของภาพด้วยจึงเป็นที่มาของการเขียน โปรแกรมที่มีภาพ 3 มิติเป็นองค์ประกอบ



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

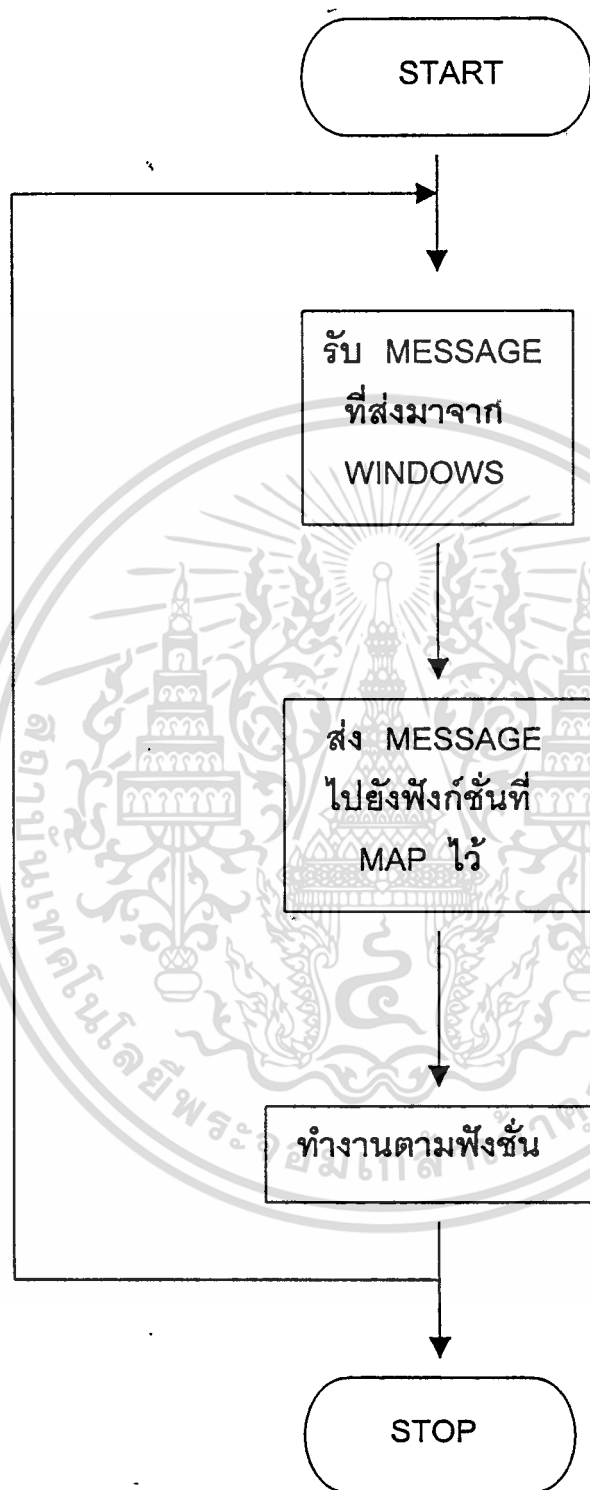
เอกสารอ้างอิง

1. Alan Watt (1994) “ 3D Computer Graphic “ Addison – Wesley.
2. Scott Anderson (1994) “ PC Graphics Unleashed ” Sams Publishing .
3. Kate Gregory (1997) “ Using Visual C++ 5 ” Que
4. Michael Sweet , Richard S. Wright JR (1996) “ OpenGL Superbible ” Waite Group press





เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



FLOW CHART แสดงขั้นตอนการทำงาน โปรแกรม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



โปรแกรมที่ 1 การสร้างวัตถุ 3 มิติ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
//*****
```

```
// ChildFrm.cpp
```

```
//*****
```

```
#include "stdafx.h"
```

```
#include "Tool.h"
```

```
#include "RenderView.h"
```

```
#include "FormCommandView.h"
```

```
#include "ChildFrm.h"
```

```
#ifdef _DEBUG
```

```
#define new DEBUG_NEW
```

```
#undef THIS_FILE
```

```
static char THIS_FILE[] = __FILE__;
```

```
#endif
```

```
////////////////////////////////////
```

```
// CChildFrame
```

```
IMPLEMENT_DYNCREATE(CChildFrame, CMDIChildWnd)
```

```
BEGIN_MESSAGE_MAP(CChildFrame, CMDIChildWnd)
```

```
//{{AFX_MSG_MAP(CChildFrame)
```

```
ON_WM_PAINT()
```

```
//}}AFX_MSG_MAP
```

```
END_MESSAGE_MAP()
```

```
////////////////////////////////////
```

```
// CChildFrame construction/destruction
```

```
CChildFrame::CChildFrame()
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

    }

    return TRUE;
}

BOOL CChildFrame::PreCreateWindow(CREATESTRUCT& cs)
{
    // TODO: Modify the Window class or styles here by modifying
    // the CREATESTRUCT cs
    cs.style = WS_CHILD | WS_VISIBLE | WS_OVERLAPPED | WS_CAPTION |
    WS_SYSMENU
        | FWS_ADDTOTITLE | WS_THICKFRAME | WS_MINIMIZEBOX |
    WS_MAXIMIZEBOX | WS_MAXIMIZE;

    return CMDIChildWnd::PreCreateWindow(cs);
}

////////////////////////////////////
// CChildFrame diagnostics
#ifdef _DEBUG
void CChildFrame::AssertValid() const
{
    CMDIChildWnd::AssertValid();
}
void CChildFrame::Dump(CDumpContext& dc) const
{
    CMDIChildWnd::Dump(dc);
}
#endif // _DEBUG
////////////////////////////////////
// CChildFrame message handlers
void CChildFrame::OnPaint()
{
    CPaintDC dc(this);
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

{
    // TODO: add member initialization code here

}

CChildFrame::~CChildFrame()
{
}

//*****
// OnCreateClient
//*****
BOOL CChildFrame::OnCreateClient( LPCREATESTRUCT /*lpcs*/,
    CCreateContext* pContext)
{
    if (!m_wndSplitter.CreateStatic(this, 1, 2, WS_CHILD | WS_VISIBLE))
    {
        TRACE("Failed to CreateStaticSplitter\n");
        return FALSE;
    }

    // First splitter pane
    if (!m_wndSplitter.CreateView(0, 0,
        RUNTIME_CLASS(CFormCommandView), CSize(200,200), pContext))
    {
        TRACE("Failed to create command view pane\n");
        return FALSE;
    }

    // Second splitter pane
    if (!m_wndSplitter.CreateView(0, 1,
        RUNTIME_CLASS(CRenderView), CSize(200,200), pContext))
    {
        TRACE("Failed to create preview pane\n");
        return FALSE;
    }
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
//*****
```

```
// DlgAbout.cpp
```

```
//*****
```

```
#include "StdAfx.h"
```

```
#include "DlgAbout.h"
```

```
CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
```

```
{
```

```
   //{{AFX_DATA_INIT(CAboutDlg)
```

```
   //}}AFX_DATA_INIT
```

```
}
```

```
void CAboutDlg::DoDataExchange(CDataExchange* pDX)
```

```
{
```

```
    CDialog::DoDataExchange(pDX);
```

```
   //{{AFX_DATA_MAP(CAboutDlg)
```

```
   //}}AFX_DATA_MAP
```

```
}
```

```
BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
```

```
   //{{AFX_MSG_MAP(CAboutDlg)
```

```
        // No message handlers
```

```
   //}}AFX_MSG_MAP
```

```
END_MESSAGE_MAP()
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
CFormCommandView::~CFormCommandView()
```

```
{  
}
```

```
void CFormCommandView::DoDataExchange(CDataExchange* pDX)
```

```
{  
    CFormView::DoDataExchange(pDX);  
   //{{AFX_DATA_MAP(CFormCommandView)  
    DDX_Control(pDX, IDC_SLIDER_Z, m_SliderScaleZ);  
    DDX_Control(pDX, IDC_SLIDER_Y, m_SliderScaleY);  
    DDX_Control(pDX, IDC_SLIDER_X, m_SliderScaleX);  
    DDX_Control(pDX, IDC_FRAME_COLOR_LIGHT_AMBIENT,  
m_ControlColorLightAmbient);  
    DDX_Control(pDX, IDC_FRAME_COLOR_BACK, m_ControlBackColor);  
    DDX_Check(pDX, IDC_CHECK_LINK_SCALE, m_LinkScale);  
  
   //}}AFX_DATA_MAP  
}
```

```
BEGIN_MESSAGE_MAP(CFormCommandView, CFormView)
```

```
    {{{AFX_MSG_MAP(CFormCommandView)  
    ON_WM_PAINT()  
    ON_WM_LBUTTONDOWN()  
    ON_WM_HSCROLL()  
    ON_BN_CLICKED(IDC_CHECK_LINK_SCALE, OnCheckLinkScale)  
    ON_COMMAND(ID_OPENGL_TYPE_SURFACE, OnOpenglTypeSurface)  
    ON_COMMAND(ID_OPENGL_TYPE_VERTEX, OnOpenglTypeVertex)  
    ON_COMMAND(ID_OPENGL_TYPE_WIREFRAME, OnOpenglTypeWireframe)  
    ON_COMMAND(ID_OPENGL_ROTATE, OnOpenglRotate)  
    ON_COMMAND(ID_OPENGL_EFFECT_ANTIALIASING, OnOpenglEffectAntialiasing)  
    ON_COMMAND(ID_OPENGL_EFFECT_SMOOTH, OnOpenglEffectSmooth)  
    ON_COMMAND(ID_OPENGL_EFFECT_LIGHT, OnOpenglEffectLight)  
    ON_BN_CLICKED(IDC_SPHERE, OnSphere)  
    ON_BN_CLICKED(IDC_CYLINDER, OnCylinder)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
// FormCommandView.cpp : implementation file
```

```
//
```

```
#include "stdafx.h"
```

```
#include "Tool.h"
```

```
#include "ChildFrm.h"
```

```
#include "MainFrm.h"
```

```
#include "FormCommandView.h"
```

```
#include "ToolDoc.h"
```

```
#include "RenderView.h"
```

```
#ifdef _DEBUG
```

```
#define new DEBUG_NEW
```

```
#undef THIS_FILE
```

```
static char THIS_FILE[] = __FILE__;
```

```
#endif
```

```
////////////////////////////////////
```

```
// CFormCommandView
```

```
IMPLEMENT_DYNCREATE(CFormCommandView, CFormView)
```

```
CFormCommandView::CFormCommandView()
```

```
: CFormView(CFormCommandView::IDD)
```

```
{
```

```
    {{{AFX_DATA_INIT(CFormCommandView)
```

```
        m_Lighting = TRUE;
```

```
        m_Smooth = FALSE;
```

```
        m_VRotate = FALSE;
```

```
        m_LinkScale = FALSE;
```

```
        m_Antialias = FALSE;
```

```
    }}}AFX_DATA_INIT
```

```
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

ON_BN_CLICKED(IDC_BOX, OnBox)
ON_BN_CLICKED(IDC_CONE, OnGone)
ON_BN_CLICKED(IDC_DODECAHEDRON, OnDodecahedron)
ON_BN_CLICKED(IDC_ICOSAHEDRON, OnIcosahedron)
ON_BN_CLICKED(IDC_OCTAHEDRON, OnOctahedron)
ON_BN_CLICKED(IDC_TETRAHEDRON, OnTetrahedron)
ON_BN_CLICKED(IDC_TORUS, OnTorus)

//}}AFX_MSG_MAP
,END_MESSAGE_MAP()

////////////////////////////////////
// CFormCommandView diagnostics

#ifdef _DEBUG
void CFormCommandView::AssertValid() const
{
    CFormView::AssertValid();
}

void CFormCommandView::Dump(CDumpContext& dc) const
{
    CFormView::Dump(dc);
}

CToolDoc* CFormCommandView::GetDocument() // non-debug version is inline
{
    ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CToolDoc)));
    return (CToolDoc*)m_pDocument;
}

#endif // _DEBUG

//*****
// OnPaint
//*****

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

void CFormCommandView::OnPaint()
{
    // Device context for painting
    CPaintDC dc(this);

    // Options are stored in Application
    CToolApp *pApp = (CToolApp *)AfxGetApp();
    CRect rect;

    // Color back
    m_ControlBackColor.GetWindowRect(&rect);
    ScreenToClient(&rect);
    CBrush BrushBack(pApp->m_OptionColorGiBack);
    dc.FillRect(&rect,&BrushBack);

    // Color light ambient
    m_ControlColorLightAmbient.GetWindowRect(&rect);
    ScreenToClient(&rect);
    CBrush BrushLightAmbient(pApp->m_OptionColorGLightAmbient);
    dc.FillRect(&rect,&BrushLightAmbient);
}

//*****
// OnLButtonUp
//*****

void CFormCommandView::OnLButtonUp(UINT nFlags, CPoint point)
{
    CRect rect;
    CToolApp *pApp = (CToolApp *)AfxGetApp();

    // Option back color
    m_ControlBackColor.GetWindowRect(&rect);
    ScreenToClient(&rect);
    if(rect.PtInRect(point))
    {

```

```

CColorDialog dlg(pApp->m_OptionColorGIBack);
if(dlg.DoModal()==IDOK)
{
    pApp->m_OptionColorGIBack = dlg.GetColor();
    CRenderView *pView = (CRenderView *)GetRenderView();
    pView->m_ClearColorRed = (float)GetRValue(pApp->
m_OptionColorGIBack) / 255.0f;
    pView->m_ClearColorGreen = (float)GetGValue(pApp->
m_OptionColorGIBack) / 255.0f;
    pView->m_ClearColorBlue = (float)GetBValue(pApp->
m_OptionColorGIBack) / 255.0f;
    this->InvalidateRect(&rect,FALSE);
    pView->InvalidateRect(NULL,FALSE);
}
}

// Option ambient light color
m_ControlColorLightAmbient.GetWindowRect(&rect);
ScreenToClient(&rect);
if(rect.PtInRect(point))
{
    CColorDialog dlg(pApp->m_OptionColorGILightAmbient);
    if(dlg.DoModal()==IDOK)
    {
        pApp->m_OptionColorGILightAmbient = dlg.GetColor();
        CRenderView *pView = (CRenderView *)GetRenderView();

        // Refresh Light0
        float r = (float)GetRValue(pApp->m_OptionColorGILightAmbient) /
255.0f;
        float g = (float)GetGValue(pApp->m_OptionColorGILightAmbient) /
255.0f;
        float b = (float)GetBValue(pApp->m_OptionColorGILightAmbient) /
255.0f;
        float ambientProperties[] = {r,g,b,1.0f};

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        UpdateScale();
        break;
    case TB_ENDTRACK:
        UpdateScale();
        break;
    case TB_LINEDOWN:
        UpdateScale();
        break;
    case TB_LINEUP:
        UpdateScale();
        break;
    case TB_PAGEDOWN:
        UpdateScale();
        break;
    case TB_PAGEUP:
        UpdateScale();
        break;
    case TB_THUMBPOSITION:
        UpdateScale();
        break;
    case TB_THUMBTRACK:
        UpdateScale();
        break;
    case TB_TOP:
        UpdateScale();
        break;
    default:
        {}
    }

    // Update scaling
    GetRenderView()->InvalidateRect(NULL,FALSE);

    CFormView::OnHScroll(nSBCode, nPos, pScrollBar);
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
glLightfv( GL_LIGHT0, GL_AMBIENT, ambientProperties);
```

```
// Refresh views
```

```
this->InvalidateRect(&rect,FALSE);
```

```
pView->InvalidateRect(NULL,FALSE);
```

```
}
```

```
}
```

```
CFormView::OnLButtonUp(nFlags, point);
```

```
}
```

```
//*****
```

```
// GetRenderView
```

```
//*****
```

```
CView *CFormCommandView::GetRenderView()
```

```
{
```

```
CToolApp *pApp = (CToolApp *)AfxGetApp();
```

```
CMainFrame *pMainFrame = (CMainFrame *)pApp->m_pMainWnd;
```

```
CChildFrame *pFrame = (CChildFrame *)pMainFrame->GetActiveFrame();
```

```
CView *pView = (CView *)pFrame->m_wndSplitter.GetPane(0,1);
```

```
return pView;
```

```
}
```

```
//*****
```

```
// Model
```

```
//*****
```

```
//*****
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
// OnCheckLighting
```

```
//*****
```

```
//*****
```

```
// OnCheckSmooth
```

```
//*****
```

```
//*****
```

```
// OnInitialUpdate
```

```
//*****
```

```
void CFormCommandView::OnInitialUpdate()
```

```
{
```

```
    CFormView::OnInitialUpdate();
```

```
    // Slider
```

```
    TRACE("Sliders : updating...\n");
```

```
    m_SliderScaleX.SetRange(1,100,TRUE);
```

```
    m_SliderScaleY.SetRange(1,100,TRUE);
```

```
    m_SliderScaleZ.SetRange(1,100,TRUE);
```

```
    m_SliderScaleX.SetPos(50);
```

```
    m_SliderScaleY.SetPos(50);
```

```
    m_SliderScaleZ.SetPos(50);
```

```
}
```

```
//*****
```

```
// OnHScroll
```

```
//*****
```

```
void CFormCommandView::OnHScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar)
```

```
{
```

```
    switch( nSBCode )
```

```
    {
```

```
        case TB_BOTTOM:
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

BOOL CFormCommandView::UpdateScale()
{
    CRenderView *pView = (CRenderView *)GetRenderView();
    pView->m_xScaling = (float)m_SliderScaleX.GetPos()/50.0f;
    pView->m_yScaling = (float)m_SliderScaleY.GetPos()/50.0f; // TODO : link
    pView->m_zScaling = (float)m_SliderScaleZ.GetPos()/50.0f;

    if(m_LinkScale)
    {
        m_SliderScaleY.SetPos(m_SliderScaleX.GetPos());
        m_SliderScaleZ.SetPos(m_SliderScaleX.GetPos());
        pView->m_yScaling = pView->m_zScaling = pView->m_xScaling;
    }

    return TRUE;
}

void CFormCommandView::OnCheckLinkScale()
{
    m_LinkScale = !m_LinkScale;

    if(m_LinkScale)
    {
        CRenderView *pView = (CRenderView *)GetRenderView();
        m_SliderScaleY.SetPos(m_SliderScaleX.GetPos());
        m_SliderScaleZ.SetPos(m_SliderScaleX.GetPos());
        pView->m_yScaling = pView->m_zScaling = pView->m_xScaling;
    }

    m_SliderScaleY.EnableWindow(!m_LinkScale);
    m_SliderScaleZ.EnableWindow(!m_LinkScale);
    GetRenderView()->InvalidateRect(NULL, FALSE);
}

```

```
}
```

```
//*****
```

```
// OnCheckAntialias
```

```
// Toggle antialiased lines
```

```
//*****
```

```
void CFormCommandView::OnOpenGLTypeSurface()
{
    // TODO: Add your command handler code here
    CRenderView *pView = (CRenderView *)GetRenderView();
    glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
    this->GetRenderView()->InvalidateRect(NULL, FALSE);
}
```

```
void CFormCommandView::OnOpenGLTypeVertex()
{
    // TODO: Add your command handler code here
    CRenderView *pView = (CRenderView *)GetRenderView();
    glPolygonMode(GL_FRONT_AND_BACK, GL_POINT);
    GetRenderView()->InvalidateRect(NULL, FALSE);
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

void CFormCommandView::OnOpenglTypeWireframe()
{
    // TODO: Add your command handler code here
    glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
    this->GetRenderView()->InvalidateRect(NULL, FALSE);
}

```

```

void CFormCommandView::OnOpenglRotate()
{
    // TODO: Add your command handler code here
    m_VRotate = !m_VRotate;
    CRenderView *pView = (CRenderView *)GetRenderView();
    if(m_VRotate)
        pView->SetTimer(1, 10, NULL);
    else
        pView->KillTimer(1);
}

```

```

void CFormCommandView::OnOpenglEffectAntialiasing()
{
    m_Antialias = !m_Antialias;

    if(m_Antialias)
    {
        glEnable(GL_LINE_SMOOTH);
        glEnable(GL_BLEND);
        glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
        glHint(GL_LINE_SMOOTH_HINT, GL_NICEST);
    }
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        glLineWidth(1.5);
    }
    else
    {
        glDisable(GL_LINE_SMOOTH);
        glDisable(GL_BLEND);
        glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
        glHint(GL_LINE_SMOOTH_HINT, GL_NICEST);
        glLineWidth(1.0);
    }
    GetRenderView()->InvalidateRect(NULL, FALSE);
}

void CFormCommandView::OnOpenGLEffectSmooth()
{
    // TODO: Add your command handler code here
    m_Smooth = !m_Smooth;
    if(m_Smooth)
        glShadeModel(GL_SMOOTH);
    else
        glShadeModel(GL_FLAT);
    this->GetRenderView()->InvalidateRect(NULL, FALSE);
}

void CFormCommandView::OnOpenGLEffectLight()
{
    // TODO: Add your command handler code here
    m_Lighting = !m_Lighting;
    if(m_Lighting)
        glEnable(GL_LIGHTING);
    else
        glDisable(GL_LIGHTING);
    this->GetRenderView()->InvalidateRect(NULL, FALSE);
}

```

```

void CFormCommandView::OnSphere()
{
    // TODO: Add your control notification handler code here
    CToolDoc *pDoc = (CToolDoc*)GetDocument();
    pDoc->m_antDoo = 1;
    GetRenderView()->InvalidateRect(NULL,FALSE);
}

```

```

void CFormCommandView::OnCylinder()
{
    // TODO: Add your control notification handler code here
    CToolDoc *pDoc = (CToolDoc*)GetDocument();
    pDoc->m_antDoc = 2;
    GetRenderView()->InvalidateRect(NULL,FALSE);
}

```

```

void CFormCommandView::OnBox()
{
    // TODO: Add your control notification handler code here
    CToolDoc *pDoc = (CToolDoc*)GetDocument();
    pDoc->m_antDoc = 4;
    GetRenderView()->InvalidateRect(NULL,FALSE);
}

```

```

void CFormCommandView::OnCone()
{
    // TODO: Add your control notification handler code here
    CToolDoc *pDoc = (CToolDoc*)GetDocument();
    pDoc->m_antDoc = 5;
    GetRenderView()->InvalidateRect(NULL,FALSE);
}

```

```

void CFormCommandView::OnDodecahedron()
{
    // TODO: Add your control notification handler code here
    CToolDoc *pDoc = (CToolDoc*)GetDocument();

```

```

        pDoc->m_antDoc = 6;
        GetRenderView()->InvalidateRect(NULL,FALSE);
    }

void CFormCommandView::OnIcosahedron()
{
    // TODO: Add your control notification handler code here
    CToolDoc *pDoc = (CToolDoc*)GetDocument();
    pDoc->m_antDoc = 7;
    GetRenderView()->InvalidateRect(NULL,FALSE);
}

void CFormCommandView::OnOctahedron()
{
    // TODO: Add your control notification handler code here
    CToolDoc *pDoc = (CToolDoc*)GetDocument();
    pDoc->m_antDoc = 8;
    GetRenderView()->InvalidateRect(NULL,FALSE);
}

void CFormCommandView::OnTetrahedron()
{
    // TODO: Add your control notification handler code here
    CToolDoc *pDoc = (CToolDoc*)GetDocument();
    pDoc->m_antDoc = 9;
    GetRenderView()->InvalidateRect(NULL,FALSE);
}

void CFormCommandView::OnTorus()
{
    // TODO: Add your control notification handler code here
    CToolDoc *pDoc = (CToolDoc*)GetDocument();
    pDoc->m_antDoc = 10;
    GetRenderView()->InvalidateRect(NULL,FALSE);
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

//*****
// FormCommandView.h
// The command panel view
//*****

#if
!defined(AFX_FORMCOMMANDVIEW_H_A4283CC1_72E6_11D1_A6C2_00A0242C0A32__INC
LUDED_)
#define
AFX_FORMCOMMANDVIEW_H_A4283CC1_72E6_11D1_A6C2_00A0242C0A32__INCLUDED_
#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
////////////////////
// CFormCommandView form view
#ifndef __AFXEXT_H__
#include <afxext.h>
#endif
class CToolDoc:
class CFormCommandView : public CFormView
{
protected:
    CFormCommandView(); // protected constructor used by dynamic creation
    DECLARE_DYNCREATE(CFormCommandView)

// Form Data
public:
   //{{AFX_DATA(CFormCommandView)
    enum { IDD = IDD_FORM_COMMAND };
    CSliderCtrl      m_SliderScaleZ;
    CSliderCtrl      m_SliderScaleY;
    CSliderCtrl      m_SliderScaleX;
    CStatic m_ControlColorLightAmbient;
    CStatic m_ControlBackColor;
    BOOL m_Lighting;
    BOOL m_Smooth;
    BOOL m_VRotate;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        BOOL m_LinkScale;
    //}}AFX_DATA

// Attributes
public:
    CToolDoc* GetDocument();
    CView *GetRenderView();
    BOOL UpdateScale();

// Operations
public:
    BOOL m_Antialias;

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CFormCommandView)
    public:
        virtual void OnInitialUpdate();
        protected:
        virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}}AFX_VIRTUAL

// Implementation
protected:
    virtual ~CFormCommandView();

#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

// Generated message map functions
//{{AFX_MSG(CFormCommandView)
afx_msg void OnPaint();
afx_msg void OnLButtonUp(UINT nFlags, CPoint point);
afx_msg void OnHScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar);
afx_msg void OnCheckLinkScale();
afx_msg void OnCylindercheck();
afx_msg void OnSpherecheck();
afx_msg void OnOpenGLTypeSurface();

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

afx_msg void OnOpenglTypeVertex();
afx_msg void OnOpenglTypeWireframe();
afx_msg void OnOpenglRotate();
afx_msg void OnOpenglEffectAntialiasing();
afx_msg void OnOpenglEffectSmooth();
afx_msg void OnOpenglEffectLight();
afx_msg void OnSphere();
afx_msg void OnCylinder();
afx_msg void OnBox();
afx_msg void OnCone();
afx_msg void OnDodecahedron();
afx_msg void OnIcosahedron();
afx_msg void OnOctahedron();
afx_msg void OnTetrahedron();
afx_msg void OnTorus();
//}}AFX_MSG
DECLARE_MESSAGE_MAP()
};

#ifdef _DEBUG // debug version in RenderView.cpp
inline CToolDoc* CFormCommandView::GetDocument()
{ return (CToolDoc*)m_pDocument; }
#endif

////////////////////////////////////
//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately before the previous
line.

#endif //
!defined(AFX_FORMCOMMANDVIEW_H_A4283CC1_72E6_11D1_A6C2_00A0242C0A32__INC
LUDED_)

```

```

//*****
// CMainFram.h
// Main window
//*****

#ifndef AFX_MAINFRM_H__06A3562D_72E5_11D1_A6C2_00A0242C0A32__INCLUDED_
#define AFX_MAINFRM_H__06A3562D_72E5_11D1_A6C2_00A0242C0A32__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

class CMainFrame : public CMDIFrameWnd
{
public:
    DECLARE_DYNAMIC(CMainFrame)

    CMainFrame();

// Attributes
public:

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
   //{{AFX_VIRTUAL(CMainFrame)
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
   //}}AFX_VIRTUAL

// Implementation
public:
    virtual ~CMainFrame();

#ifdef _DEBUG

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

virtual void AssertValid() const;

virtual void Dump(CDumpContext& dc) const;

#endif

protected: // control bar embedded members
    CStatusBar m_wndStatusBar;
    CToolBar m_wndToolBar;

// Generated message map functions
protected:
   //{{AFX_MSG(CMainFrame)
    afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
    afx_msg void OnPaint();
   //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately before the previous
// line.

#endif //
!defined(AFX_MAINFRM_H__06A3562D_72E5_11D1_A6C2_00A0242C0A32__INCLUDED_)

```

```

//*****

// RenderView.h : interface of the CRenderView class

//*****

#ifdef __cplusplus
extern "C" {
#endif

#ifndef AFX_RENDERVIEW_H__06A35633_72E5_11D1_A6C2_00A0242C0A32__INCLUDED_
#define AFX_RENDERVIEW_H__06A35633_72E5_11D1_A6C2_00A0242C0A32__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

class CToolDoc;

class CRenderView : public CView
{
protected: // create from serialization only
    CRenderView();
    DECLARE_DYNCREATE(CRenderView)

// Attributes
public:
    CToolDoc* GetDocument();

// Operations
public:

    // OpenGL specific
    BOOL SetWindowPixelFormat(HDC hDC);
    BOOL CreateViewGLContext(HDC hDC);
    HGLRC m_hGLContext;
    int m_GLPixelFormat;

    // Mouse
    BOOL m_LeftButtonDown;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
 ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
BOOL m_RightButtonDown;  
CPoint m_LeftDownPos;  
CPoint m_RightDownPos;  
HCURSOR m_CursorRotation;
```

```
// Position, rotation ,scaling
```

```
void InitGeometry(void);
```

```
float m_xRotation;
```

```
float m_yRotation;
```

```
float m_xTranslation;
```

```
float m_yTranslation;
```

```
float m_zTranslation;
```

```
float m_xScaling;
```

```
float m_yScaling;
```

```
float m_zScaling;
```

```
// Colors
```

```
float m_ClearColorRed;
```

```
float m_ClearColorGreen;
```

```
float m_ClearColorBlue;
```

```
// Overrides
```

```
// ClassWizard generated virtual function overrides
```

```
//{{AFX_VIRTUAL(CRenderView)
```

```
public:
```

```
virtual void OnDraw(CDC* pDC); // overridden to draw this view
```

```
virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
```

```
protected:
```

```
virtual BOOL OnPreparePrinting(CPrintInfo* pInfo);
```

```
virtual void OnBeginPrinting(CDC* pDC, CPrintInfo* pInfo);
```

```
virtual void OnEndPrinting(CDC* pDC, CPrintInfo* pInfo);
```

```
//}}AFX_VIRTUAL
```

```
// Implementation
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

public:
    virtual ~CRenderView();

#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

protected:
    // Generated message map functions
protected:
   //{{AFX_MSG(CRenderView)
    afx_msg void OnDestroy();
    afx_msg void OnSize(UINT nType, int cx, int cy);
    afx_msg void OnLButtonDown(UINT nFlags, CPoint point);
    afx_msg void OnLButtonUp(UINT nFlags, CPoint point);
    afx_msg void OnMouseMove(UINT nFlags, CPoint point);
    afx_msg void OnPaint();
    afx_msg void OnTimer(UINT nIDEvent);
    afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
   //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

#ifndef _DEBUG // debug version in RenderView.cpp
inline CToolDoc* CRenderView::GetDocument()
{ return (CToolDoc*)m_pDocument; }
#endif

////////////////////
//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately before the previous
// line.

#endif //
!defined(AFX_RENDERVIEW_H__06A35633_72E5_11D1_A6C2_00A0242C0A32__INCLUDED_)

```

```

//{{NO_DEPENDENCIES}}
//Microsoft Developer Studio generated include file.
// Used by Tool.rc
//

#define IDD_ABOUTBOX 100
#define IDR_MAINFRAME 128
#define IDR_MODELTYPE 129
#define IDC_CURSOR_ROTATION 131
#define IDD_FORM_COMMAND 150
#define IDC_FRAME_COLOR_BACK 1000
#define IDC_CHECK_ANTIALIAS 1001
#define IDC_CHECK_SMOOTH 1002
#define IDC_RADIO_MODEL_1 1003
#define IDC_CHECK_LIGHTING 1004
#define IDC_RADIO_MODEL_2 1005
#define IDC_RADIO_MODEL_0 1006
#define IDC_CHECK_VROTATION 1007
#define IDC_SLIDER_X 1008
#define IDC_CHECK_LINK_SCALE 1009
#define IDC_FRAME_COLOR_LIGHT_AMBIENT 1010
#define IDC_SLIDER_Y 1011
#define IDC_SLIDER_Z 1012
#define IDC_FRAME_COLOR_LIGHT_AMBIENT2 1013
#define IDC_FRAME_COLOR_LIGHT_AMBIENT3 1014
#define IDC_SPHERECHECK 1015
#define IDC_CYLINDERCHECK 1016
#define IDC_SPHERE 1017
#define IDC_CYLINDER 1018
#define IDC_TEAPOT 1019
#define IDC_BOX 1020
#define IDC_CONE 1021
#define IDC_DODECAHEDRON 1022
#define IDC_ICOSAHEDRON 1023
#define IDC_OCTAHEDRON 1024
#define IDC_TETRAHEDRON 1025

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

#define IDC_TORUS          1026
#define ID_VIEW_TSPHERE    32771
#define ID_VIEW_TCYLINDER  32772
#define ID_OPENGL_TYPE_VERTEX 32775
#define ID_OPENGL_TYPE_WIREFRAME 32776
#define ID_OPENGL_TYPE_SURFACE 32777
#define ID_OPENGL_ROTATE    32785
#define ID_OPENGL_EFFECT_ANTIALIASING 32787
#define ID_OPENGL_EFFECT_SMOOTH 32788
#define ID_OPENGL_EFFECT_LIGHT 32789

```

```

// Next default values for new objects

```

```

//

```

```

#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_3D_CONTROLS 1
#define _APS_NEXT_RESOURCE_VALUE 132
#define _APS_NEXT_COMMAND_VALUE 32794
#define _APS_NEXT_CONTROL_VALUE 1027
#define _APS_NEXT_SYMED_VALUE 101
#endif
#endif

```

```

//*****

// Stdafx.h

//*****

// stdafx.h : include file for standard system include files,
// or project specific include files that are used frequently, but
// are changed infrequently
//

#ifndef AFX_STDAFX_H__06A3562B_72E5_11D1_A6C2_00A0242C0A32__INCLUDED_
#define AFX_STDAFX_H__06A3562B_72E5_11D1_A6C2_00A0242C0A32__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

#define VC_EXTRALEAN // Exclude rarely-used stuff from Windows headers

#include <afxwin.h> // MFC core and standard components
#include <afxext.h> // MFC extensions
#include <afxdisp.h> // MFC OLE automation classes
#ifndef _AFX_NO_AFXCMN_SUPPORT
#include <afxcmn.h> // MFC support for Windows Common Controls
#endif // _AFX_NO_AFXCMN_SUPPORT

////////////////////
////////////////////

// Do not care OpenGL Headers...
////////////////////
////////////////////

#include <gl\gl.h>
#include <gl\glu.h>
#include <GL\glaux.h>

//{{AFX_INSERT_LOCATION}}

// Microsoft Developer Studio will insert additional declarations immediately before the previous
// line.

#endif //

#ifndef AFX_STDAFX_H__06A3562B_72E5_11D1_A6C2_00A0242C0A32__INCLUDED_

```

```
*****  
// Tool.h  
// Application  
*****
```

```
#if !defined(AFX_TOOL_H__06A35629_72E5_11D1_A6C2_00A0242C0A32__INCLUDED_)  
#define AFX_TOOL_H__06A35629_72E5_11D1_A6C2_00A0242C0A32__INCLUDED_
```

```
#if _MSC_VER >= 1000  
#pragma once  
#endif // _MSC_VER >= 1000
```

```
#ifndef __AFXWIN_H__  
#error include 'stdafx.h' before including this file for PCH  
#endif
```

```
#include "resource.h" // main symbols
```

```
////////////////////////////////////  
// CToolApp:  
// See Tool.cpp for the implementation of this class  
//
```

```
class CToolApp : public CWinApp  
{
```

```
// Options  
public:
```

```
    // Registry keys  
    BOOL LoadOptions();  
    BOOL SaveOptions();
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        COLORREF m_OptionColorGIBack;
        COLORREF m_OptionColorGILightAmbient;

public:
        CToolApp();

```

// Overrides

```

        // ClassWizard generated virtual function overrides
        //{AFX_VIRTUAL(CToolApp)
public:
        virtual BOOL InitInstance();
        virtual int ExitInstance();
        //}AFX_VIRTUAL

```

// Implementation

```

        //{AFX_MSG(CToolApp)
afx_msg void OnAppAbout();

        // NOTE - the ClassWizard will add and remove member functions here.
        // DO NOT EDIT what you see in these blocks of generated code !

        //}AFX_MSG
        DECLARE_MESSAGE_MAP()

};

```

```

////////////////////////////////////

```

```

        //{AFX_INSERT_LOCATION}

```

// Microsoft Developer Studio will insert additional declarations immediately before the previous line.

```

#endif // !defined(AFX_TOOL_H__06A35629_72E5_11D1_A6C2_00A0242C0A32__INCLUDED_)

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

//*****
// ToolDoc.h
// The document, contain a SceneGraph
//*****

#if !defined(AFX_TOOLDOC_H_06A35631_72E5_11D1_A6C2_00A0242C0A32__INCLUDED_)
#define AFX_TOOLDOC_H_06A35631_72E5_11D1_A6C2_00A0242C0A32__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

class CSeneGraph3d;
class CToolDoc : public CDocument
{
protected: // create from serialization only
    CToolDoc();
    DECLARE_DYNCREATE(CToolDoc)

// Attributes
private:
    //CSeneGraph3d m_SceneGraph;

// Functions
public:
    // OpenGL
    void RenderScene();

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
   //{{AFX_VIRTUAL(CToolDoc)
public:
    virtual BOOL OnNewDocument();
    virtual void Serialize(CArchive& ar);
   //}}AFX_VIRTUAL

// Implementation
public:
    BOOL m_faceDoc;

    void Torus();

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

void Tetrahedron();

void Octahedron();

void Icosahedron();

void Dodecahedron();

void Cone();

void Box();

int m_antDoc;

virtual ~CToolDoc();

#ifdef _DEBUG

    virtual void AssertValid() const;

    virtual void Dump(CDumpContext& dc) const;

#endif

protected:

// Generated message map functions
protected:

//{{AFX_MSG(CToolDoc)
    // NOTE - the ClassWizard will add and remove member functions here.
    // DO NOT EDIT what you see in these blocks of generated code !
//}}AFX_MSG
    DECLARE_MESSAGE_MAP()

};

////////////////////////////////////

//{{AFX_INSERT_LOCATION}}

// Microsoft Developer Studio will insert additional declarations immediately before the previous
// line.

#endif //

#ifndef AFX_TOOLDOC_H__06A35631_72E5_11D1_A6C2_00A0242C0A32__INCLUDED_

```

โปรแกรมที่ 2 การประยุกต์ใช้งาน



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

//*****
// ToolDoc.h
// The document, contain a SceneGraph

#ifndef AFX_TOOLDOC_H__06A35631_72E5_11D1_A6C2_00A0242C0A32__INCLUDED_
#define AFX_TOOLDOC_H__06A35631_72E5_11D1_A6C2_00A0242C0A32__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
#ifndef __CTOOLDOC__
#define __CTOOLDOC__

class CSeneGraph3d;

class CToolDoc : public CDocument
{
protected: // create from serialization only
    CToolDoc();
    DECLARE_DYNCREATE(CToolDoc)

// Attributes
private:
    //CSeneGraph3d m_SceneGraph;

// Functions
public:

    // OpenGL
    void RenderScene(float compo1,float compo2,float compo3,float compo4,float
    compo5);

// Operations
public:
    /*
    float m_fComponent1;
    float m_fComponent2;
    float m_fComponent3;
    float m_fComponent4;
    float m_fComponent5;
    */
    float a,b,c,d,e;

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CToolDoc)
    public:
        virtual BOOL OnNewDocument();
        virtual void Serialize(CArchive& ar);
    //}}AFX_VIRTUAL

// Implementation
public:
    virtual ~CToolDoc();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

protected:

```

เอกสารนี้ Generated message map functions ศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
 ไม่ว่ากรณีใดๆ //{{AFX_MSG(CToolDoc) ปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

// NOTE - the ClassWizard will add and remove member functions here.
//      DO NOT EDIT what you see in these blocks of generated code !
//}}AFX_MSG
DECLARE_MESSAGE_MAP()
};

////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately
before the previous line.

#endif // !defined(AFX_TOOLDOC_H__06A35631_72E5_11D1_A6C2_00A0242C0A32__INCLUD
ED_)
#endif

```



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

//*****
// ToolDoc.cpp
//*****

#include "stdafx.h"
#include "math.h"
#include "Tool.h"

#include "ToolDoc.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

////////////////////////////////////
// CToolDoc

IMPLEMENT_DYNCREATE(CToolDoc, CDocument)

BEGIN_MESSAGE_MAP(CToolDoc, CDocument)
    //{AFX_MSG_MAP(CToolDoc)
        // NOTE - the ClassWizard will add and remove mapping macros here.
        // DO NOT EDIT what you see in these blocks of generated code!
    //}AFX_MSG_MAP
END_MESSAGE_MAP()

////////////////////////////////////
// CToolDoc construction/destruction

//*****
// Constructor
//*****
CToolDoc::CToolDoc()
{
}

//*****
// Destructor
//*****
CToolDoc::~CToolDoc()
{
}

BOOL CToolDoc::OnNewDocument()
{
    if (!CDocument::OnNewDocument())
        return FALSE;

    return TRUE;
}

////////////////////////////////////
// CToolDoc serialization

void CToolDoc::Serialize(CArchive& ar)
{
    if (ar.IsStoring())
    {
        // TODO: add storing code here
    }
    else
    {
        // TODO: add loading code here
    }
}

```

เอกสารนี้ถ้าหาก (ar.IsStoring()) การใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
 ไม่ว่ากรณีใดๆ // TODO: add storing code here หรืออ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

else
{
    // TODO: add loading code here
}
}

////////////////////////////////////
// CToolDoc diagnostics

#ifdef _DEBUG
void CToolDoc::AssertValid() const
{
    CDocument::AssertValid();
}

void CToolDoc::Dump(CDumpContext& dc) const
{
    CDocument::Dump(dc);
}
#endif // _DEBUG

////////////////////////////////////
// CToolDoc commands

////////////////////////////////////
// RENDERING
////////////////////////////////////

//*****
// RenderScene
//*****
void CToolDoc::RenderScene(float compo1,float compo2,float compo3
                           ,float compo4,float compo5)
{
    // Default rendering

    //a = compo1 - 50;
    b = compo2 - 30;
    c = compo3;
    d = compo4 - 90;
    e = compo5;

    glColor3ub(255,255,255);
    GLUQuadricObj* q = gluNewQuadric();
    gluQuadricDrawStyle(q, GLU_FILL);
    gluQuadricNormals(q, GLU_SMOOTH);

    //
    glTranslatef (0.0, -1.0, 0.0);
    glRotatef (90, 0.0, 0.0, 1.0);

    //auxWireBox(1.0, 5.0, 5.0);
    auxSolidBox(0.7, 0.7, 0.7);
    glTranslatef ( 0.35f, 0.0, 0.0);    //1

    glRotatef (compo1, 1.0, 0.0, 0.0);
    glTranslatef (0.39f, 0.0, 0.0);    //2
    auxSolidBox(0.7, 0.7, 0.7);
    glTranslatef (0.3f, 0.0, 0.0);    //3
    glRotatef (b, 0.0, 0.0, 1.0);

```

```
glTranslatef (0.5f, 0.0, 0.0); //4
//glRotatef (-30, 0.0, 0.0, 1.0);
auxSolidBox(1.3f, 0.4, 0.65f);
```

```
glTranslatef ( 0.6f, 0.0, 0.0); //5
glRotatef (c, 0.0, 0.0, 1.0);
glTranslatef (0.5f, 0.0, 0.0); //6
auxSolidBox(1.1f, 0.4, 0.65f);
```

```
glTranslatef (0.5f, 0.05f, 0.0); //7
glRotatef (d, 0.0, 0.0, 1.0);
glTranslatef (0.05f, -0.98f, 0.0f); //8
auxSolidCylinder ( 0.2f, 0.75);
```

```
////////////////////////////////////
//////// Hand component
```

```
glTranslatef (0.018f, 0.1f, 0.0);
glRotatef ( e, 0.0, 1.0, 0.0);
auxSolidCylinder ( 0.2f, 0.04f);
```

```
glTranslatef (0.17f, 1.21f, 0.0f);
//glRoTatef( 0, 0.0, 0.0, 1.0);
auxSolidBox(0.1f, 0.48f, 0.1f);
glTranslatef (-0.1f, 0.21f, 0.0f);
auxSolidBox(0.1f, 0.2f, 0.1f);
```

```
glTranslatef (-0.23f, -0.20f, 0.0f);
//glRoTatef( 0, 0.0, 0.0, 1.0);
auxSolidBox(0.1f, 0.48f, 0.1f);
glTranslatef (0.1f, 0.21f, 0.0f);
auxSolidBox(0.1f, 0.2f, 0.1f);
```

```
gluDeleteQuadric(q);
```

```
}
```

```

//*****
// RenderView.h : interface of the CRenderView class
//*****
#ifdef AFX_RENDERVIEW_H__06A35633_72E5_11D1_A6C2_00A0242C0A32__INCLUDED_
)
#define AFX_RENDERVIEW_H__06A35633_72E5_11D1_A6C2_00A0242C0A32__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

class CToolDoc;

class CRenderView : public CView
{
protected: // create from serialization only
    CRenderView();
    DECLARE_DYNCREATE(CRenderView)

// Attributes
public:
    CToolDoc* GetDocument();

// Operations
public:

    // OpenGL specific
    BOOL SetWindowPixelFormat(HDC hDC);
    BOOL CreateViewGLContext(HDC hDC);
    HGLRC m_hGLContext;
    int m_GLPixelFormatIndex;

    // Mouse
    BOOL m_LeftButtonDown;
    BOOL m_RightButtonDown;
    CPoint m_LeftDownPos;
    CPoint m_RightDownPos;
    HCURSOR m_CursorRotation;

    // Position, rotation ,scaling
    void InitGeometry(void);
    float m_xRotation;
    float m_yRotation;
    float m_xTranslation;
    float m_yTranslation;
    float m_zTranslation;
    float m_xScaling;
    float m_yScaling;
    float m_zScaling;
    float m_4Scaling;
    float m_5Scaling;
    float temp1,temp2,temp3,temp4,temp5;
    // Colors
    float m_ClearColorRed;
    float m_ClearColorGreen;
    float m_ClearColorBlue;

    // Active
    BOOL m_bActive;
    BOOL a,b,c,d,e,f,serial;
    BYTE data1,data2,data3,data4,data5;

// Overrides
// ClassWizard generated virtual function overrides

```

เอกสารนี้เป็นเอกสารของบริษัทฯ ห้ามเผยแพร่โดยไม่ได้รับอนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ทำซ้ำหรือดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

//{{AFX_VIRTUAL(CRenderView)
public:
virtual void OnDraw(CDC* pDC); // overridden to draw this view
virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
protected:
virtual BOOL OnPreparePrinting(CPrintInfo* pInfo);
virtual void OnBeginPrinting(CDC* pDC, CPrintInfo* pInfo);
virtual void OnEndPrinting(CDC* pDC, CPrintInfo* pInfo);
//}}AFX_VIRTUAL

// Implementation
public:
void Compare(void);
virtual ~CRenderView();
#ifdef _DEBUG
virtual void AssertValid() const;
virtual void Dump(CDumpContext& dc) const;
#endif

protected:

// Generated message map functions
protected:
//{{AFX_MSG(CRenderView)
afx_msg void OnDestroy();
afx_msg void OnSize(UINT nType, int cx, int cy);
afx_msg void OnLButtonDown(UINT nFlags, CPoint point);
afx_msg void OnLButtonUp(UINT nFlags, CPoint point);
afx_msg void OnMouseMove(UINT nFlags, CPoint point);
afx_msg void OnPaint();
afx_msg void OnTimer(UINT nIDEvent);
afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
//}}AFX_MSG
DECLARE_MESSAGE_MAP()
};

#ifndef _DEBUG // debug version in RenderView.cpp
inline CToolDoc* CRenderView::GetDocument()
{ return (CToolDoc*)m_pDocument; }
#endif

/////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately
before the previous line.

#endif // !defined(AFX_RENDERVIEW_H__06A35633_72E5_11D1_A6C2_00A0242C0A32__INC
LUDED_)

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

//*****
// RenderView.cpp
//*****

#include "stdafx.h"
#include "Tool.h"

#include "ToolDoc.h"
#include "RenderView.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

////////////////////////////////////
// CRenderView

IMPLEMENT_DYNCREATE(CRenderView, CView)

BEGIN_MESSAGE_MAP(CRenderView, CView)
   //{{AFX_MSG_MAP(CRenderView)
    ON_WM_DESTROY()
    ON_WM_SIZE()
    ON_WM_LBUTTONDOWN()
    ON_WM_LBUTTONUP()
    ON_WM_MOUSEMOVE()
    ON_WM_PAINT()
    ON_WM_TIMER()
    ON_WM_CREATE()
   //}}AFX_MSG_MAP
    // Standard printing commands
    ON_COMMAND(ID_FILE_PRINT, CView::OnFilePrint)
    ON_COMMAND(ID_FILE_PRINT_DIRECT, CView::OnFilePrint)
    ON_COMMAND(ID_FILE_PRINT_PREVIEW, CView::OnFilePrintPreview)
END_MESSAGE_MAP()

////////////////////////////////////
// CRenderView construction/destruction

CRenderView::CRenderView()
{
    // OpenGL
    m_hGLContext = NULL;
    m_GLPixelFormat = 0;

    // Mouse
    m_LeftButtonDown = FALSE;
    m_RightButtonDown = FALSE;
    m_CursorRotation = AfxGetApp()->LoadCursor(IDC_CURSOR_ROTATION);

    // Colors
    CToolApp *pApp = (CToolApp *)AfxGetApp();
    m_ClearColorRed = GetRValue(pApp->m_OptionColorGLBack);
    m_ClearColorGreen = GetGValue(pApp->m_OptionColorGLBack);
    m_ClearColorBlue = GetBValue(pApp->m_OptionColorGLBack);

    InitGeometry();
}

//*****
// InitGeometry
//*****
void CRenderView::InitGeometry(void)

```

```

}

void CRenderView::Dump(CDumpContext& dc) const
{
    CView::Dump(dc);
}

CToolDoc* CRenderView::GetDocument() // non-debug version is inline
{
    ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CToolDoc)));
    return (CToolDoc*)m_pDocument;
}
#endif // _DEBUG

```

```

////////////////////////////////////
////////////////////////////////////
// OPENGL
////////////////////////////////////
////////////////////////////////////

```

```

//*****
// OnCreate
// Create OpenGL rendering context
//*****
int CRenderView::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
    if (CView::OnCreate(lpCreateStruct) == -1)
        return -1;

    HWND hWnd = GetSafeHwnd();
    HDC hDC = ::GetDC(hWnd);

    if(SetWindowPixelFormat(hDC)==FALSE)
        return 0;

    if(CreateViewGLContext(hDC)==FALSE)
        return 0;

    // Default mode
    glPolygonMode(GL_FRONT, GL_LINE);
    glPolygonMode(GL_BACK, GL_LINE);
    glShadeModel(GL_FLAT);
    glEnable(GL_NORMALIZE);
    glLineWidth(3.5);

    // Lights, material properties
    GLfloat ambientProperties[] = {0.7f, 0.7f, 0.7f, 1.0f};
    GLfloat diffuseProperties[] = {0.8f, 0.8f, 0.8f, 1.0f};
    GLfloat specularProperties[] = {1.0f, 1.0f, 1.0f, 1.0f};

    glClearDepth( 1.0 );

    glLightfv( GL_LIGHT0, GL_AMBIENT, ambientProperties);
    glLightfv( GL_LIGHT0, GL_DIFFUSE, diffuseProperties);
    glLightfv( GL_LIGHT0, GL_SPECULAR, specularProperties);
    glLightModel(GL_LIGHT_MODEL_TWO_SIDE, 1.0);

```

เอกสารนี้เป็นเอกสารที่จัดทำขึ้นโดยระบบอัตโนมัติของเอกสารทุกครั้งที่มีการนำไปใช้
 ไม่ควรแก้ไข ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

{
    m_xRotation = 0.0f;
    m_yRotation = 0.0f;
    m_xTranslation = 0.0f;
    m_yTranslation = 0.0f;
    m_zTranslation = -5.0f;
    m_xScaling = 1.0f;
    m_yScaling = 1.0f;
    m_zScaling = 90.0f;
    m_4Scaling = 1.0f;
    m_5Scaling = 1.0f;
    m_bActive = TRUE;
    a = b = c = d = e = TRUE;
}

```

```

CRenderView::~CRenderView()
{
}

```

```

BOOL CRenderView::PreCreateWindow(CREATESTRUCT& cs)
{
    // TODO: Modify the Window class or styles here by modifying
    // the CREATESTRUCT cs

    return CView::PreCreateWindow(cs);
}

```

```

////////////////////////////////////
// CRenderView drawing

```

```

void CRenderView::OnDraw(CDC* pDC)
{
    CToolDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    // TODO: add draw code for native data here
}

```

```

////////////////////////////////////
// CRenderView printing

```

```

BOOL CRenderView::OnPreparePrinting(CPrintInfo* pInfo)
{
    // default preparation
    return DoPreparePrinting(pInfo);
}

```

```

void CRenderView::OnBeginPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
    // TODO: add extra initialization before printing
}

```

```

void CRenderView::OnEndPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
    // TODO: add cleanup after printing
}

```

```

////////////////////////////////////
// CRenderView diagnostics

```

```

#ifdef _DEBUG

```

void CRenderView::AssertValid() const { CView::AssertValid(); }
 เอกสารที่แนบมานี้ ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
 ไม่ว่ากรณีใดๆ และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

// Default : lighting
glEnable(GL_LIGHT0);
glEnable(GL_LIGHTING);

return 0;
}

//*****
// SetWindowPixelFormat
//*****
BOOL CRenderView::SetWindowPixelFormat(HDC hDC)
{
    PIXELFORMATDESCRIPTOR pixelDesc;

    pixelDesc.nSize = sizeof(PIXELFORMATDESCRIPTOR);
    pixelDesc.nVersion = 1;

    pixelDesc.dwFlags = PFD_DRAW_TO_WINDOW | PFD_SUPPORT_OPENGL |
        PFD_DOUBLEBUFFER | PFD_STEREO_DONTCARE;

    pixelDesc.iPixelFormat = PFD_TYPE_RGBA;
    pixelDesc.cColorBits = 32;
    pixelDesc.cRedBits = 8;
    pixelDesc.cRedShift = 16;
    pixelDesc.cGreenBits = 8;
    pixelDesc.cGreenShift = 8;
    pixelDesc.cBlueBits = 8;
    pixelDesc.cBlueShift = 0;
    pixelDesc.cAlphaBits = 0;
    pixelDesc.cAlphaShift = 0;
    pixelDesc.cAccumBits = 64;
    pixelDesc.cAccumRedBits = 16;
    pixelDesc.cAccumGreenBits = 16;
    pixelDesc.cAccumBlueBits = 16;
    pixelDesc.cAccumAlphaBits = 0;
    pixelDesc.cDepthBits = 32;
    pixelDesc.cStencilBits = 8;
    pixelDesc.cAuxBuffers = 0;
    pixelDesc.iLayerType = PFD_MAIN_PLANE;
    pixelDesc.bReserved = 0;
    pixelDesc.dwLayerMask = 0;
    pixelDesc.dwVisibleMask = 0;
    pixelDesc.dwDamageMask = 0;

    m_GLPixelFormat = ChoosePixelFormat(hDC, &pixelDesc);
    if(m_GLPixelFormat == 0) // Choose default
    {
        m_GLPixelFormat = 1;
        if(DescribePixelFormat(hDC, m_GLPixelFormat,
            sizeof(PIXELFORMATDESCRIPTOR), &pixelDesc) == 0)
            return FALSE;
    }

    if(!SetPixelFormat(hDC, m_GLPixelFormat, &pixelDesc))
        return FALSE;

    return TRUE;
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
 ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

//*****

```

```

// CreateViewGLContext
// Create an OpenGL rendering context
//*****
BOOL CRenderView::CreateViewGLContext(HDC hDC)
{
    m_hGLContext = wglCreateContext(hDC);

    if(m_hGLContext==NULL)
        return FALSE;

    if(wglMakeCurrent(hDC,m_hGLContext)==FALSE)
        return FALSE;

    return TRUE;
}

//*****
// OnDestroy
// Cleanup every OpenGL rendering context
//*****
void CRenderView::OnDestroy()
{
    if(wglGetCurrentContext() != NULL)
        wglMakeCurrent(NULL,NULL);

    if(m_hGLContext != NULL)
    {
        wglDeleteContext(m_hGLContext);
        m_hGLContext = NULL;
    }

    CView::OnDestroy();
}

//*****
// OnSize
//*****
void CRenderView::OnSize(UINT nType, int cx, int cy)
{
    CView::OnSize(nType, cx, cy);

    // Set OpenGL perspective, viewport and mode
    CSize size(cx,cy);
    double aspect;
    aspect = (cy == 0) ? (double)size.cx : (double)size.cx/(double)size.cy;

    glViewport(0,0,size.cx,size.cy);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45,aspect,1,15.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glDrawBuffer(GL_BACK);
    glEnable(GL_DEPTH_TEST);

}

```

```

////////////////////
// MOUSE

```

เอกสารนี้เป็นทรัพย์สินทางปัญญาของมหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
 ไม่ว่าในรูปแบบใด ๆ ของเอกสารทุกครั้งที่มีการนำไปใช้

```

//*****
// Left button Mouse
//*****
void CRenderView::OnLButtonDown(UINT nFlags,
                                CPoint point)
{
    m_LeftButtonDown = TRUE;
    m_LeftDownPos = point;
    CView::OnLButtonDown(nFlags, point);
}

void CRenderView::OnLButtonUp(UINT nFlags, CPoint point)
{
    m_LeftButtonDown = FALSE;
    CView::OnLButtonUp(nFlags, point);
}

void CRenderView::OnMouseMove(UINT nFlags, CPoint point)
{
    if(m_LeftButtonDown)
    {
        m_yRotation -= (float)(m_LeftDownPos.x - point.x)/3.0f;
        m_LeftDownPos = point;
        InvalidateRect(NULL, FALSE);
    }
    CView::OnMouseMove(nFlags, point);
}

////////////////////
////////////////////
// PAINTING
////////////////////
////////////////////

//*****
// OnPaint
//*****
void CRenderView::OnPaint()
{
    // Device context for painting
    CPaintDC dc(this);

    // Model is stored in Document
    CToolDoc *pDoc = (CToolDoc *)GetDocument();
    ASSERT_VALID(pDoc);

    // Useful in multidoc templates
    HWND hWnd = GetSafeHwnd();
    HDC hDC = ::GetDC(hWnd);
    wglMakeCurrent(hDC, m_hGLContext);

    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    glClearColor(m_ClearColorRed, m_ClearColorGreen, m_ClearColorBlue, 1.0f);

    glPushMatrix();

    // Position / translation / scale
    glTranslated(m_xTranslation, m_yTranslation, m_zTranslation);
    glRotatef(m_xRotation, 1.0, 0.0, 0.0);

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
 ไม่ว่าการตีพิมพ์เอกสารนี้เป็นการละเมิดลิขสิทธิ์ของเอกสารที่มีการนำไปใช้

```

glRotatef(m_yRotation, 0.0, 1.0, 0.0);

// Start rendering...
pDoc->RenderScene(m_xScaling,m_yScaling,m_zScaling,m_4Scaling,m_5Scaling);

glPopMatrix();

// Double buffer
SwapBuffers(dc.m_ps.hdc);
}

//*****
// OnTimer
//*****
// Currently animation timer
//*****
void CRenderView::OnTimer(UINT nIDEvent)
{
    switch(nIDEvent)
    {
        case 0:
            break;
        // Rotation
        case 1:
            m_yRotation += 5.0f;
            InvalidateRect(NULL,FALSE);
            break;
        default:
            {}
    }
}

void CRenderView::Compare()
{
    if( m_xScaling < temp1)
    {
        data1 = BYTE(temp1 - m_xScaling);
        a = FALSE;
    }
    else
        data1 = BYTE(m_xScaling - temp1);

    if( m_yScaling < temp2)
    {
        data2 = BYTE(temp2 - m_yScaling);
        b = FALSE;
    }
    else
        data2 = BYTE(m_yScaling - temp2);

    if( m_zScaling < temp3)
    {
        data3 = BYTE(m_zScaling - temp3);
        c = FALSE;
    }
    else
        data3 = BYTE(m_zScaling - temp3);

    if( m_4Scaling < temp4)
    {
        data4 = BYTE(m_4Scaling - temp4);
        d = FALSE;
    }
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ผู้สนใจสามารถติดต่อขอข้อมูลเพิ่มเติมได้ที่ฝ่ายงานเอกสารและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

else
    data4 = BYTE(m_4Scaling - temp4);

if( m_5Scaling < temp5)
{
    data5 = BYTE(m_5Scaling - temp5);
    e = FALSE;
}
else
    data5 = BYTE(m_5Scaling - temp5);

```



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

//*****
// ChildFrm.h
// Child frame window, do not forget this
// is a multidoc template
//*****

#ifdef AFX_CHILDFRM_H__06A3562F_72E5_11D1_A6C2_00A0242C0A32__INCLUDED_
#define AFX_CHILDFRM_H__06A3562F_72E5_11D1_A6C2_00A0242C0A32__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

class CChildFrame : public CMDIChildWnd
{
    DECLARE_DYNCREATE(CChildFrame)
public:
    CChildFrame();

// Attributes
public:
    CSplitterWnd m_wndSplitter;
public:

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CChildFrame)
public:
    virtual BOOL OnCreateClient(LPCREATESTRUCT lpcs, CCreateContext* pContext)
;
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
    //}}AFX_VIRTUAL

// Implementation
public:
    virtual ~CChildFrame();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

// Generated message map functions
protected:
    //{{AFX_MSG(CChildFrame)
    afx_msg void OnPaint();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately
// before the previous line.

#endif // !defined(AFX_CHILDFRM_H__06A3562F_72E5_11D1_A6C2_00A0242C0A32__INCLU
DED_)

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

//*****
// ChildFrm.cpp
//*****

#include "stdafx.h"
#include "Tool.h"

#include "RenderView.h"
#include "FormCommandView.h"

#include "ChildFrm.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

////////////////////////////////////
// CChildFrame

IMPLEMENT_DYNCREATE(CChildFrame, CMDIChildWnd)

BEGIN_MESSAGE_MAP(CChildFrame, CMDIChildWnd)
    //{{AFX_MSG_MAP(CChildFrame)
    ON_WM_PAINT()
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

////////////////////////////////////
// CChildFrame construction/destruction

CChildFrame::CChildFrame()
{
    // TODO: add member initialization code here
}

CChildFrame::~CChildFrame()
{
}

//*****
// OnCreateClient
//*****
BOOL CChildFrame::OnCreateClient( LPCTSTR lpszClassName,
    CCreateContext* pContext)
{
    if (!m_wndSplitter.CreateStatic(this, 1, 2, WS_CHILD | WS_VISIBLE))
    {
        TRACE("Failed to CreateStaticSplitter\n");
        return FALSE;
    }

    // First splitter pane
    if (!m_wndSplitter.CreateView(0, 0,
        RUNTIME_CLASS(CRenderView), CSize(600,200), pContext))
    {
        TRACE("Failed to create command view pane\n");
        return FALSE;
    }

    // Second splitter pane
    if (!m_wndSplitter.CreateView(0, 1,

```

```

        RUNTIME_CLASS(CFormCommandView), CSize(100,200), pContext))
    {
        TRACE("Failed to create preview pane\n");
        return FALSE;
    }

    return TRUE;
}

BOOL CChildFrame::PreCreateWindow(CREATESTRUCT& cs)
{
    // TODO: Modify the Window class or styles here by modifying
    // the CREATESTRUCT cs
    cs.style = WS_CHILD | WS_VISIBLE | WS_OVERLAPPED | WS_CAPTION | WS_SYSMEN
U   | FWS_ADDTOTITLE | WS_THICKFRAME | WS_MINIMIZEBOX | WS_MAXIMIZEBOX | W
S_MAXIMIZE;
    return CMDIChildWnd::PreCreateWindow(cs);
}

//////////////////////////////////////
// CChildFrame diagnostics

#ifdef _DEBUG
void CChildFrame::AssertValid() const
{
    CMDIChildWnd::AssertValid();
}

void CChildFrame::Dump(CDumpContext& dc) const
{
    CMDIChildWnd::Dump(dc);
}

#endif //_DEBUG

//////////////////////////////////////
// CChildFrame message handlers

void CChildFrame::OnPaint()
{
    CPaintDC dc(this);
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

//*****
// FormCommandView.h
// The command panel view
//*****

```

```

#if !defined(AFX_FORMCOMMANDVIEW_H_A4283CC1_72E6_11D1_A6C2_00A0242C0A32__INCL
UDED_)
#define AFX_FORMCOMMANDVIEW_H_A4283CC1_72E6_11D1_A6C2_00A0242C0A32__INCLUDED_

```

```

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

```

```

////////////////////////////////////
// CFormCommandView form view

```

```

#ifdef __AFXEXT_H__
#include <afxext.h>
#endif

```

```

class CToolDoc;

```

```

class CFormCommandView : public CFormView
{
protected:
    CFormCommandView(); // protected constructor used by dynamic cre
ation
    DECLARE_DYNCREATE(CFormCommandView)

```

```

// Form Data
public:

```

```

//{{AFX_DATA(CFormCommandView)
enum { IDD = IDD_FORM_COMMAND };
CSliderCtrl m_Slider1;
CSliderCtrl m_Slider2;
CSliderCtrl m_Slider3;
CSliderCtrl m_Slider4;
CSliderCtrl m_Slider5;
CStatic m_ControlColorLightAmbient;
CStatic m_ControlBackColor;
BOOL m_Lighting;
BOOL m_Smooth;
BOOL m_Antialias;
int m_Model;
BOOL m_VRotate;
BOOL m_LinkScale;
//BYTE data1,data2,data3,data4,data5;
//BOOL a,b,c,d,e;
//int temp1,temp2,temp3,temp4,temp5;
//int rentemp1,rentemp2,rentemp3,rentemp4,rentemp5;
BYTE nData;
//}}AFX_DATA

```

```

// Attributes
public:

```

```

CToolDoc* GetDocument();
CView *GetRenderView();
BOOL UpdateScale();
HANDLE hCom;
LPVOID lpBufW;

```

```

// Operations
public:

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
 ไม่อนุญาตให้นำไปใช้ซ้ำโดยไม่ได้รับอนุญาตจากเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

void OutComport(BYTE nData);

// Overrides
// ClassWizard generated virtual function overrides
//{{AFX_VIRTUAL(CFormCommandView)
public:
virtual void OnInitialUpdate();
protected:
virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
//}}AFX_VIRTUAL

// Implementation
protected:
virtual ~CFormCommandView();
#ifdef _DEBUG
virtual void AssertValid() const;
virtual void Dump(CDumpContext& dc) const;
#endif

// Generated message map functions
//{{AFX_MSG(CFormCommandView)
afx_msg void OnPaint();
afx_msg void OnLButtonUp(UINT nFlags, CPoint point);
afx_msg void OnCheckLighting();
afx_msg void OnCheckSmooth();
afx_msg void OnHScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar);
afx_msg void OnCheckLinkScale();
afx_msg void OnCheckAntialias();
afx_msg void OnRadio1();
afx_msg void OnRadio2();
afx_msg void OnActive();
//}}AFX_MSG
DECLARE_MESSAGE_MAP()
};

#ifndef _DEBUG // debug version in RenderView.cpp
inline CToolDoc* CFormCommandView::GetDocument()
{ return (CToolDoc*)m_pDocument; }
#endif

//////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately
before the previous line.

#endif // !defined(AFX_FORMCOMMANDVIEW_H__A4283CC1_72E6_11D1_A6C2_00A0242C0A32__INCLUDED_)

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

// FormCommandView.cpp : implementation file
//

#include "stdafx.h"
#include "Tool.h"
#include "ChildFrm.h"
#include "MainFrm.h"
#include "FormCommandView.h"
#include "ToolDoc.h"
#include "RenderView.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

//////////////////////////////////////
// CFormCommandView

IMPLEMENT_DYNCREATE(CFormCommandView, CFormView)

CFormCommandView::CFormCommandView()
: CFormView(CFormCommandView::IDD)
{
    //{{AFX_DATA_INIT(CFormCommandView)
    m_Lighting = TRUE;
    m_Smooth = FALSE;
    m_Antialias = TRUE;
    m_Model = 1;
    m_VRotate = FALSE;
    m_LinkScale = FALSE;
    //}}AFX_DATA_INIT
}

CFormCommandView::~CFormCommandView()
{
}

void CFormCommandView::DoDataExchange(CDataExchange* pDX)
{
    CFormView::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CFormCommandView)
    DDX_Control(pDX, IDC_SLIDER1, m_Slider1);
    DDX_Control(pDX, IDC_SLIDER2, m_Slider2);
    DDX_Control(pDX, IDC_SLIDER3, m_Slider3);
    DDX_Control(pDX, IDC_SLIDER4, m_Slider4);
    DDX_Control(pDX, IDC_SLIDER5, m_Slider5);
    DDX_Control(pDX, IDC_FRAME_COLOR_LIGHT_AMBIENT, m_ControlColorLightAmbient);
    DDX_Control(pDX, IDC_FRAME_COLOR_BACK, m_ControlBackColor);
    DDX_Check(pDX, IDC_CHECK_LIGHTING, m_Lighting);
    DDX_Check(pDX, IDC_CHECK_SMOOTH, m_Smooth);
    DDX_Check(pDX, IDC_CHECK_ANTIALIAS, m_Antialias);
    DDX_Radio(pDX, IDC_RADIO_MODEL_0, m_Model);
    DDX_Check(pDX, IDC_CHECK_VROTATION, m_VRotate);
    DDX_Check(pDX, IDC_CHECK_LINK_SCALE, m_LinkScale);
    //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CFormCommandView, CFormView)
    //{{AFX_MSG_MAP(CFormCommandView)
    ON_WM_PAINT()
    ON_WM_LBUTTONDOWN()

```

```

ON_BN_CLICKED(IDC_CHECK_LIGHTING, OnCheckLighting)
ON_BN_CLICKED(IDC_CHECK_SMOOTH, OnCheckSmooth)
ON_WM_HSCROLL()
ON_BN_CLICKED(IDC_CHECK_LINK_SCALE, OnCheckLinkScale)
ON_BN_CLICKED(IDC_CHECK_ANTIALIAS, OnCheckAntialias)
ON_BN_CLICKED(IDC_RADIO1, OnRadio1)
ON_BN_CLICKED(IDC_RADIO2, OnRadio2)
ON_BN_CLICKED(IDC_BUTTON2, OnActive)
//}}AFX_MSG_MAP
END_MESSAGE_MAP()

```

```

////////////////////////////////////
// CFormCommandView diagnostics

```

```

#ifdef _DEBUG
void CFormCommandView::AssertValid() const
{
    CFormView::AssertValid();
}

```

```

void CFormCommandView::Dump(CDumpContext& dc) const
{
    CFormView::Dump(dc);
}

```

```

CToolDoc* CFormCommandView::GetDocument() // non-debug version is inline
{
    ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CToolDoc)));
    return (CToolDoc*)m_pDocument;
}

```

```

#endif // _DEBUG

```

```

//*****
// OnPaint
//*****

```

```

void CFormCommandView::OnPaint()
{
    // Device context for painting
    CPaintDC dc(this);

    // Options are stored in Application
    CToolApp *pApp = (CToolApp *)AfxGetApp();
    CRect rect;

    // Color back
    m_ControlBackColor.GetWindowRect(&rect);
    ScreenToClient(&rect);
    CBrush BrushBack(pApp->m_OptionColorGlBack);
    dc.FillRect(&rect, &BrushBack);

    // Color light ambient
    m_ControlColorLightAmbient.GetWindowRect(&rect);
    ScreenToClient(&rect);
    CBrush BrushLightAmbient(pApp->m_OptionColorGlLightAmbient);
    dc.FillRect(&rect, &BrushLightAmbient);
}

```

```

//*****
// OnLButtonUp
//*****

```

```

void CFormCommandView::OnLButtonUp(UINT nFlags, CPoint po
int)
{
    //

```

```

CRect rect;
CToolApp *pApp = (CToolApp *)AfxGetApp();

// Option back color
m_ControlBackColor.GetWindowRect(&rect);
ScreenToClient(&rect);
if(rect.PtInRect(point))
{
    CColorDialog dlg(pApp->m_OptionColorGlbBack);
    if(dlg.DoModal() == IDOK)
    {
        pApp->m_OptionColorGlbBack = dlg.GetColor();
        CRenderView *pView = (CRenderView *)GetRenderView();
        pView->m_ClearColorRed = (float)GetRValue(pApp->m_OptionColorGlb
ack) / 255.0f;
        pView->m_ClearColorGreen = (float)GetGValue(pApp->m_OptionColorGlb
ack) / 255.0f;
        pView->m_ClearColorBlue = (float)GetBValue(pApp->m_OptionColorGlb
ack) / 255.0f;
        this->InvalidateRect(&rect, FALSE);
        pView->InvalidateRect(NULL, FALSE);
    }
}

// Option ambient light color
m_ControlColorLightAmbient.GetWindowRect(&rect);
ScreenToClient(&rect);
if(rect.PtInRect(point))
{
    CColorDialog dlg(pApp->m_OptionColorGllLightAmbient);
    if(dlg.DoModal() == IDOK)
    {
        pApp->m_OptionColorGllLightAmbient = dlg.GetColor();
        CRenderView *pView = (CRenderView *)GetRenderView();

        // Refresh Light0
        float r = (float)GetRValue(pApp->m_OptionColorGllLightAmbient) / 25
5.0f;
        float g = (float)GetGValue(pApp->m_OptionColorGllLightAmbient) / 25
5.0f;
        float b = (float)GetBValue(pApp->m_OptionColorGllLightAmbient) / 25
5.0f;

        float ambientProperties[] = {r,g,b,1.0f};
        glLightfv( GL_LIGHT0, GL_AMBIENT, ambientProperties);

        // Refresh views
        this->InvalidateRect(&rect, FALSE);
        pView->InvalidateRect(NULL, FALSE);
    }
}

CFormView::OnLButtonUp(nFlags, point);
}

//*****
// GetRenderView
//*****
CView *CFormCommandView::GetRenderView()
{
    CToolApp *pApp = (CToolApp *)AfxGetApp();
    CMainFrame *pMainFrame = (CMainFrame *)pApp->m_pMainWnd;

```

```

CChildFrame *pFrame = (CChildFrame *)pMainFrame->GetActiveFrame();
CView *pView = (CView *)pFrame->m_wndSplitter.GetPane(0,0);
return pView;
}

```

```

//*****

```

```

// Model

```

```

//*****

```

```

void CFormCommandView::OnRadio1()

```

```

{
    // TODO: Add your control notification handler code here
    glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
    this->GetRenderView()->InvalidateRect(NULL, FALSE);
}

```

```

void CFormCommandView::OnRadio2()

```

```

{
    // TODO: Add your control notification handler code here
    glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
    this->GetRenderView()->InvalidateRect(NULL, FALSE);
}

```

```

//*****

```

```

// OnCheckLighting

```

```

//*****

```

```

void CFormCommandView::OnCheckLighting()

```

```

{
    m_Lighting = !m_Lighting;
    if(m_Lighting)
        glEnable(GL_LIGHTING);
    else
        glDisable(GL_LIGHTING);
    this->GetRenderView()->InvalidateRect(NULL, FALSE);
}

```

```

//*****

```

```

// OnCheckSmooth

```

```

//*****

```

```

void CFormCommandView::OnCheckSmooth()

```

```

{
    m_Smooth = !m_Smooth;
    if(m_Smooth)
        glShadeModel(GL_SMOOTH);
    else
        glShadeModel(GL_FLAT);
    this->GetRenderView()->InvalidateRect(NULL, FALSE);
}

```

```

//*****

```

```

// OnInitialUpdate

```

```

//*****

```

```

void CFormCommandView::OnInitialUpdate()

```

```

{
    CFormView::OnInitialUpdate();

```

```

    DCB dcb;
    DWORD dwError;
    BOOL fSuccess;
    BOOL fWaitingOnStat = FALSE;

```

```

    // Slider

```

```

    TRACE("Sliders : updating...\n");

```

เอกสารนี้เป็นเอกสารสงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่าการคัดลอกหรือการเผยแพร่ข้อมูลทั้งหมดนี้ให้ผู้อื่นโดยไม่ได้รับอนุญาตจากเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

m_Slider1.SetRange(1,100,TRUE);
m_Slider2.SetRange(1,100,TRUE);
m_Slider3.SetRange(1,100,TRUE);
m_Slider4.SetRange(1,100,TRUE);
m_Slider5.SetRange(1,100,TRUE);
m_Slider1.SetPos(1);
m_Slider2.SetPos(1);
m_Slider3.SetPos(90);
m_Slider4.SetPos(1);
m_Slider5.SetPos(1);

```

```

// Set for Serial Port
////////////////////

```

```

hCom = CreateFile( "COM2",

```

```

    GENERIC_READ | GENERIC_WRITE,
    0,          // comm devices must be opened w/exclusive-access
    NULL,       // no security attributes
    OPEN_EXISTING, // comm devices must use OPEN_EXISTING
    0,          // not overlapped I/O
    NULL        // hTemplate must be NULL for comm devices
);

```

```

if (hCom == INVALID_HANDLE_VALUE)
{
    dwError = GetLastError();
    // handle error
    MessageBox("Error Code NUM ", " hCom == INVALID_HANDLE_VALUE", MB_OK);
}

```

```

// Omit the call to SetupComm to use the default queue sizes.
// Get the current configuration.

```

```

fSuccess = GetCommState(hCom, &dcb);
if (!fSuccess)
{

```

```

    // Handle the error.
    MessageBox("Can not check COM Port", "Error At GetCommState", MB_OK);
}

```

```

// Fill in the DCB: baud=9600, 8 data bits, no parity, 1 stop bit.

```

```

dcb.BaudRate = 9600;

```

```

dcb.ByteSize = 8;

```

```

dcb.Parity = NOPARITY;

```

```

dcb.StopBits = ONESTOPBIT;

```

```

fSuccess = SetCommState(hCom, &dcb);

```

```

if (!fSuccess)

```

```

{

```

```

    // Handle the error.

```

```

    MessageBox("Can not set COM Port", "Error At SetCommState", MB_OK);
}

```

```

}

//*****
// OnHScroll
//*****
void CFormCommandView::OnHScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar)
{
    switch( nSBCode )
    {
        case TB_BOTTOM:
            UpdateScale();
            break;
        case TB_ENDTRACK:
            UpdateScale();
            break;
        case TB_LINEDOWN:
            UpdateScale();
            break;
        case TB_LINEUP:
            UpdateScale();
            break;
        case TB_PAGEDOWN:
            UpdateScale();
            break;
        case TB_PAGEUP:
            UpdateScale();
            break;
        case TB_THUMBPOSITION:
            UpdateScale();
            break;
        case TB_THUMBTRACK:
            UpdateScale();
            break;
        case TB_TOP:
            UpdateScale();
            break;
        default:
            {}
    }

    // Update scaling
    GetRenderView()->InvalidateRect(NULL, FALSE);

    CFormView::OnHScroll(nSBCode, nPos, pScrollBar);
}

```

```

BOOL CFormCommandView::UpdateScale()
{
    CRenderView *pView = (CRenderView *)GetRenderView();
    pView->m_xScaling = (float)m_Slider1.GetPos();
    pView->m_yScaling = (float)m_Slider2.GetPos(); // TODO : link
    pView->m_zScaling = (float)m_Slider3.GetPos();
    pView->m_4Scaling = (float)m_Slider4.GetPos();
    pView->m_5Scaling = (float)m_Slider5.GetPos();

    if(m_LinkScale)
    {
        m_Slider2.SetPos(m_Slider1.GetPos());
        m_Slider3.SetPos(m_Slider1.GetPos());
        m_Slider4.SetPos(m_Slider1.GetPos());
        m_Slider5.SetPos(m_Slider1.GetPos());
    }
}

```

เอกสารนี้เป็นเอกสารของสำนักงานคณะกรรมการ
 ไม่ว่าการใด ๆ ภายใต้อาณัติของสำนักงานคณะกรรมการ
 ไม่ว่าการใด ๆ ภายใต้อาณัติของสำนักงานคณะกรรมการ

```

        pView->m_yScaling = pView->m_zScaling = pView->m_xScaling = pView->m_4
Scaling
        =pView->m_5Scaling;
    }
    return TRUE;
}

void CFormCommandView::OnCheckLinkScale()
{
    m_LinkScale = !m_LinkScale;

    if(m_LinkScale)
    {
        CRenderView *pView = (CRenderView *)GetRenderView();
        m_Slider2.SetPos(m_Slider1.GetPos());
        m_Slider3.SetPos(m_Slider1.GetPos());
        m_Slider4.SetPos(m_Slider1.GetPos());
        m_Slider5.SetPos(m_Slider1.GetPos());
        pView->m_yScaling = pView->m_zScaling = pView->m_xScaling = pView->m_4
Scaling
        =pView->m_5Scaling;
    }

    m_Slider2.EnableWindow(!m_LinkScale);
    m_Slider3.EnableWindow(!m_LinkScale);
    m_Slider4.EnableWindow(!m_LinkScale);
    m_Slider5.EnableWindow(!m_LinkScale);
    GetRenderView()->InvalidateRect(NULL, FALSE);
}

//*****
// OnCheckAntialias
// Toggle antialiased lines
//*****
void CFormCommandView::OnCheckAntialias()
{
    m_Antialias = !m_Antialias;

    if(m_Antialias)
    {
        glEnable(GL_LINE_SMOOTH);
        glEnable(GL_BLEND);
        glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
        glHint(GL_LINE_SMOOTH_HINT, GL_NICEST);
        glLineWidth(3.5);
    }
    else
    {
        glDisable(GL_LINE_SMOOTH);
        glDisable(GL_BLEND);
        glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
        glHint(GL_LINE_SMOOTH_HINT, GL_NICEST);
        glLineWidth(1.0);
    }
    GetRenderView()->InvalidateRect(NULL, FALSE);
}

void CFormCommandView::OnActive()
{
    // TODO: Add your control notification handler code here
    OutComport(1);
}

```

```

Sleep(2500);
OutComport(3);
Sleep(2500);
OutComport(108);
Sleep(2500);
OutComport(111);
Sleep(2500);
OutComport(118);
Sleep(2500);
OutComport(101);
AfxMessageBox(" Finish to Control Robot ", MB_OK );

```



```

void CFormCommandView::OutComport(BYTE nData)
{
    DWORD dwToWrite;
    DWORD dwWritten;
    BOOL bCheck;

    lpBufW = (LPVOID)&nData;
    dwToWrite = 1;
    bCheck=WriteFile(hCom, &nData, 1, &dwWritten,NULL);
    if(!bCheck)
    {
        AfxMessageBox("Can not Write to COM Port", MB_OK);
    }
}

```



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่ใช้