สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

โปรแกรมจำลองอินเวอร์ทเพนดูลัม 3มิติ

Inverted Pendulum 3D Simulation

T045874

นายสมิทธิ์ เมฆอโนทัย

เลขหมู่.....
เลขทะเบียน 45874
วัน, เดือน, ปี 19 ก.พ. 2546

.b.....
.i.....

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมการวัดคุม

ภาควิชาวิศวกรรมการวัดคุม คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

INVERTED PENDULUM 3D SIMULATION

SMITH MEKANOTAI



A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

BACHELOR OF ENGINEERING IN INSTRUMENTATION ENGINEERING

DEPARTMENT OF INSTRUMENTATION ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTTE OF TECHNOLOGY LADKRABANG

2001

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาควิชาวิศวกรรมการวัดคุม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองปริญญานิพนธ์

หัวข้อปริญญานิพนธ์ โปรแกรมจำลองอินเวอร์ทเพนดูลัม 3มิติ

INVERTED PENDULUM 3D SIMULATION

นักศึกษาผู้จัดทำ นายสมิทธิ์ เมฆอโนทัย **รหัสประจำตัว** 41014444

ปริญญา วิศวกรรมศาสตรบัณฑิต

สาขาวิชา วิศวกรรมการวัดคุม

ปีการศึกษา 2544

อาจารย์ผู้ควบคุมปริญญานิพนธ์	**ลายมือชื่อ**
ผศ.วิริยะ กองรัตน์ อ.อาจินต์ น่วมสำราญ	

วัน/เดือน/ปี ที่สอบ วันพุธที่ 27 มีนาคม พ.ศ.2545

สถานที่สอบ ณ. ห้องสอบปริญญานิพนธ์ ภาควิชาวิศวกรรมการวัดคุม

ภาควิชารับรองแล้ว

(ผศ.ประสิทธ์ จุลเสรีวงศ์)

หัวหน้าภาควิชาฯ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หัวข้อปริญญานิพนธ์	ระบบจำลองอินเวอร์ทเพนดูลัม 3 มิติ
	INVERTED PENDULUM 3D SIMULATION
นักศึกษาผู้จัดทำ	นายสมิทธิ์ เมฆอโนทัย
อาจารย์ที่ปรึกษา	ผศ. วิริยะ กองรัตน์
	อ.อาจินต์ น่วมสำราญ
ปีการศึกษา	2544

บทคัดย่อ

การใช้งาน DIRECT3D ในการสร้างภาพ 3 มิติมีมานานแล้วจุดประสงค์หลัก ในการสร้างงานด้านคอมพิวเตอร์กราฟฟิก การ์ตูน หรือ เกม โครงการนี้จึงได้นำ DIRECT3D มาประยุกต์ เพื่อนำมาใช้งานในการสร้างภาพ 3 มิติ ให้กับระบบจำลองอินเวอร์ทเพนดูลัม

การทำระบบจำลองอินเวอร์ทเพนดูลัม 3 มิติทำได้ โดยการสร้างสมการทางคณิตศาสตร์ที่สามารถอธิบายระบบอินเวอร์ทเพนดูลัมแล้ว ใช้สมการนี้ เป็นตัวส่งข้อมูลให้กับภาพ 3 มิติที่สร้างขึ้นจาก DIRECT 3D ข้อมูลที่ส่งให้ภาพ 3 มิติ เป็นข้อมูลของตำแหน่งในการเคลื่อนที่ของ ระบบอินเวอร์ทเพนดูลัม หลังจากส่งข้อมูลดังกล่าวแล้ว จะได้ระบบจำลองอินเวอร์ทเพนดูลัม 3 มิติ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Thesis Title	Inverted Pendulum 3 D Simulation
Authers	Mr Smith Mekanotai
Thesis Advisor	Asst.Prof.Viriya Kongratana
	Mr.Arjin Numsomran
Year	2001

ABSTRACT

API OPENGL and DIRECT3D was used to create 3D picture for many years . The main Purpose of this application is to create advertisement animation and game . Therefore, This Project use the process of creating 3D animation apply to the Inverted Pendulum System, to do the Inverted Pendulum 3D simulation. The methodology to make the Inverted Pendulum 3D simulation is creating the mathematic equation by simulating from the Inverted Pendulum System, then passing the parameter of the pendulum position from this mathematic equation to the 3D picture , which is created by DIRECT3D, will perform animation of the Inverted Pendulum 3D simulation.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยดี เพราะได้รับคำปรึกษาและความอนุเคราะห์จาก ผู้ช่วยศาสตราจารย์วิริยะ กองรัตน์ และ อาจารย์อาจินต์ น่วมสำราญ ซึ่งข้าพเจ้ารู้สึกซาบซึ้งและขอกราบขอบพระคุณเป็นอย่างสูง ที่ได้ให้การสนับสนุนและคำปรึกษาเกี่ยวกับปริญญานิพนธ์ จนสำเร็จตามจุดประสงค์

ขอขอบคุณภาควิชาวิศวกรรมการวัดคุม และอาจารย์ภาควิชาวิศวกรรมการวัดคุมทุกท่าน ที่ให้คำปรึกษาในด้านต่างๆ ขอขอบคุณเพื่อนทุกคน ที่ได้ให้กำลังใจและคำปรึกษาทางด้านต่างๆ จนสำเร็จลุล่วงไปได้ด้วยดี ขอขอบคุณพี่สาวข้าพเจ้า ที่ได้ให้กำลังใจและคำปรึกษาเกี่ยวกับภาพ 3 มิติจนสำเร็จลุล่วงไปได้ด้วยดี

คุณค่าและประโยชน์อันพึงมีจากปริญญานิพนธ์ฉบับนี้ ขอมอบแด่บิดาและมารดา

ผู้จัดทำ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

หน้า

บทคัดย่อภาษาไทย..... I

บทคัดย่อภาษาอังกฤษ..... II

กิตติกรรมประกาศ..... III

สารบัญ..... IV

สารบัญรูป..... VII

บทที่ 1 บทนำ..... 1

บทที่ 2 ความรู้ด้านภาพ 3 มิติ..... 2

2.1 พื้นฐานความรู้ของภาพ 3 มิติ..... 2

2.2 Vertices ,Vector..... 2

2.3 Primitive Object..... 3

2.3.1 Triangle Strip..... 4

2.3.2 Triangle fan..... 5

2.3.3 Triangle List..... 6

2.4 Matrix..... 6

2.4.1 Translation..... 6

2.4.2 Rotation..... 6

2.4.3 Scaling..... 7

2.4.4 World ,View ,Projection Matrix..... 9

2.4.5 View Matrix..... 9

2.4.6 Projection Matrix..... 11

2.5 Material..... 12

2.5.1 Diffuse Reflect..... 12

2.5.2 Diffuse Transmission..... 12

2.5.3 Specular Reflect..... 12

2.5.4 Specular Transmission..... 12

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ(ต่อ)

หน้า

2.6 Lighting..... 12

2.6.1 Point light..... 12

2.6.2 Directional Light..... 13

2.6.3 Spot Light..... 13

2.7 Normal vector..... 14

2.8 Render Pipeline..... 14

2.9 Clipping And Viewport Scaling..... 15

2.10 Rasterize..... 15

2.11 Shading..... 17

2.12 Texture Mapping..... 17

บทที่ 3 DirectX..... 18

3.1 การทำงานของ Direct3D..... 19

3.2 การสร้างภาพใน Direct3D8..... 20

3.2.1 การสร้างอุปกรณ์หรือเครื่องมือที่ใช้ในการวาดภาพ..... 20

3.2.2 วิธีการสร้างภาพ มี 2 วิธี..... 22

1.แบบ manual..... 22

2.การใช้Mesh..... 24

3.2.3 การกำหนดตำแหน่งของวัตถุ..... 24

3.2.4 การกำหนดค่า Lighting และ Material..... 26

3.2.5 Texture Mapping..... 27

1.Wrap Texture Address Mode..... 27

2. Mirror Texture Address Mode..... 28

3 Clamp Texture Address mode..... 28

4. Border Texture Address Mode..... 28

3.2.6 Render..... 30

3.3 การสร้างภาพเคลื่อนไหว..... 30

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ(ต่อ)

หน้า

บทที่ 4 สมการคุณลักษณะของ Invert pendulum..... 32

4.1 โมเดลทางคณิตศาสตร์ของPendulum..... 32

4.2 โมเดลคณิตศาสตร์ของดีซีมอเตอร์..... 34

4.2.1 โมเดลอิเล็คโทรแมคคานิคอล..... 34

4.3 ทรานสเฟอร์ฟังก์ชันของดีซีมอเตอร์..... 38

4.4 โมเดลทางคณิตศาสตร์ของPID คอนโทรลเลอร์..... 41

4.5 Simulation Invert Pendulum..... 41

4.6 โมเดลภาพ 3 มิติ..... 43

บทที่ 5 ผลการทดลอง..... 46

บทที่ 6 บทสรุปและปัญหาที่เกิดขึ้น..... 48

ภาคผนวก

หนังสืออ้างอิง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป

รูปที่ หน้า

2.1 โครงสร้างข้อมูลของ vertice..... 3

2.2 Polygonชนิดต่างๆ..... 3

2.3 Triangle Strip 4

2.4 Triangle Strip 4

2.5 สี่เหลี่ยมลูกบาศก์ 4

2.6 หลอด 4

2.7 ทรงกระบอก..... 5

2.8 Triangle Fan 5

2.9 วงกลม..... 5

2.10 Triangle Strip 6

2.11 World Coordinate 9

2.12 การตั้งกล้องและตั้งทิศทางการมองที่ผิดพลาด 10

2.13 การตั้งทิศและที่ตั้งที่สามารถมองเห็นวัตถุได้ 10

2.14 ลักษณะของภาพเมื่อมองผ่านกล้อง..... 11

2.15 Point light..... 12

2.16 Directional Light..... 13

2.17 ลักษณะของแสงประเภท Spot lightใ 13

2.18 Spot light..... 13

2.19 เส้นสีเหลืองแสดงถึง Normal Vector..... 14

2.20 Render Pipeline..... 14

2.21 View port 15

2.22 แสดงวิธีการกำหนดการ Map ภาพไปยัง Polygon..... 16

2.23 Top-Left edge..... 16

2.24 Flat Shading..... 17

2.25 Gourade Shading..... 17

3.1 ขั้นตอนการ Render..... 19

3.2 ความสัมพันธ์ระหว่าง GDI กับ Hal..... 19

3.3 รูปทรงกระบอก..... 23

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป(ต่อ)

รูปที่	หน้า
3.4 รูปกล่อง	24
3.5 Mesh	24
3.6 แสดงถึง Coordinate ที่ใช้ใน Direct3D	25
3.7 กราฟแสดงความสัมพันธ์ของค่า Fall off กับค่า Intensity	27
3.8 Wrap mode	27
3.9 Mirror mode	28
3.10 Clamp Texture mode	28
3.11 Border Texture mode	29
3.12 ภาพก่อนนำไป Map	30
3.13 หลังจากการ Map ภาพ	30
3.14 Tearing	31
3.15 การ Flipping Page	31
4.1 การแตกแรงเพื่อหาสมการรูปแบบทางคณิตศาสตร์	32
4.2 โมเดลของดีซีมอเตอร์แบบฟิลด์แยกกระตุ้น	35
4.3 แสดงถึงแรงบิดต่างๆ ที่เกิดขึ้นต่อโหลดของมอเตอร	36
4.4 บล็อคไดอะแกรมของดีซีมอเตอร์โมเด	38
4.5 ตัวอย่างคอนโทรลมอเตอร์และเยนเนอร์เรเตอร์	39
4.6 ดีซีมอเตอร์ และทาโคมิเตอร์สำหรับใช้ในระบบบังคับมีการป้อนกลับ	40
4.7 Block Diagram	44
4.8 ก้านลูกตุ้ม	44
4.9 แขนขับ	44
4.10 ส่วนฐาน	45
4.11 ภาพสมบูรณ์	45
5.1 กราฟแสดงค่ามุมของ Moter	46
5.2 กราฟแสดงค่ามุมของ Moter	46
5.3 กราฟแสดงการแก่วงของลูกตุ้ม	46
5.4 กราฟแสดงการแก่วงของลูกตุ้ม	47

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 1
บทนำ

ปริญญานิพนธ์ฉบับนี้ มีชื่อว่า ระบบจำลองการควบคุมอินเวอร์ทเพนดูลัมคอนโทรล 3 มิติ มีจุดมุ่งหมายเพื่อแสดงภาพการเคลื่อนไหว ของอินเวอร์ทเพนดูลัม ออกมาในรูปแบบ 3 มิติเพื่อใช้ใน การทดลองการควบคุมที่สมมุติขึ้นมา โดยไม่มีตัวอุปกรณ์ ที่ใช้จริง โดยผู้ใช้สามารถเห็นการเคลื่อนที่ของ อินเวอร์ทเพนดูลัม ผ่านภาพ 3 มิติ และสามารถทำการควบคุมได้ จากตัวควบคุม ทั้ง อินเวอร์ทเพนดูลัม และตัวควบคุมถูกเขียนให้อยู่ในรูปแบบของแบบจำลองทางคณิตศาสตร์ ซึ่งสามารถพิสูจน์ได้และเพื่อเป็นตัวอย่างสำหรับการเขียนโปรแกรมเพื่อสร้างภาพการเคลื่อนไหวแบบ 3 มิติในการจำลองระบบต่างๆ โดยใช้ Direct3D ซึ่งแต่เดิมนั้นจะใช้การจำลองระบบ ซึ่งจะดูผลจากกราฟเป็นหลัก หรือแสดงค่าออกมาเป็นกราฟเท่านั้น

ปริญญานิพนธ์ได้จัดทำขึ้นโดยใช้ข้อมูลจากปริญญานิพนธ์เก่าที่ได้ทำเรื่องนี้ไว้ก่อนแล้ว

ขอบเขตของปริญญานิพนธ์

1.สามารถสร้างภาพ 3 มิติเพื่อจำลองระบบอินเวอร์ทเพนดูลัม

2.ภาพที่สร้างสามารถเคลื่อนไหวและตอบสนองต่อเวลาแบบ Real Time ได้

เนื้อหาในปริญญานิพนธ์นี้

บทที่ 1 เป็นบทนำกล่าวถึงขอบเขตของปริญญานิพนธ์ฉบับนี้

บทที่ 2 เป็นความรู้ทางด้านภาพ 3 มิติที่ได้ทำการศึกษา

บทที่ 3 เป็นวิธีการใช้งาน Direct3D ในการสร้างภาพ 3 มิติ

บทที่ 4 เป็นการอธิบายถึงสมการคุณลักษณะของอินเวอร์เพนดูลัม

บทที่ 5 เป็นผลการทดลอง

บทที่ 6 เป็นบทสรุปและปัญหาทั้งหมดที่เกิดขึ้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 2
ทฤษฎีและเนื้อหาที่เกี่ยวข้อง

กล่าวนำ

ปริญญานิพนธ์เล่มนี้ได้นำเสนอ การสร้างรูปแบบจำลองของระบบอินเวอร์ทเพนดูลัม และการสร้าง แบบจำลองในลักษณะของรูป 3 มิติ ซึ่งประกอบไปด้วยส่วนๆ เนื้อหาดั่งต่อไปนี้

- พื้นฐานความรู้ของรูปในลักษณะเป็น 3 มิติ
- Direct3D Immediate Mode
- แบบจำลองทางคณิตศาสตร์ที่ใช้เป็นตัวแทนระบบ

2.1 พื้นฐานความรู้ของรูปในลักษณะ 3 มิติ

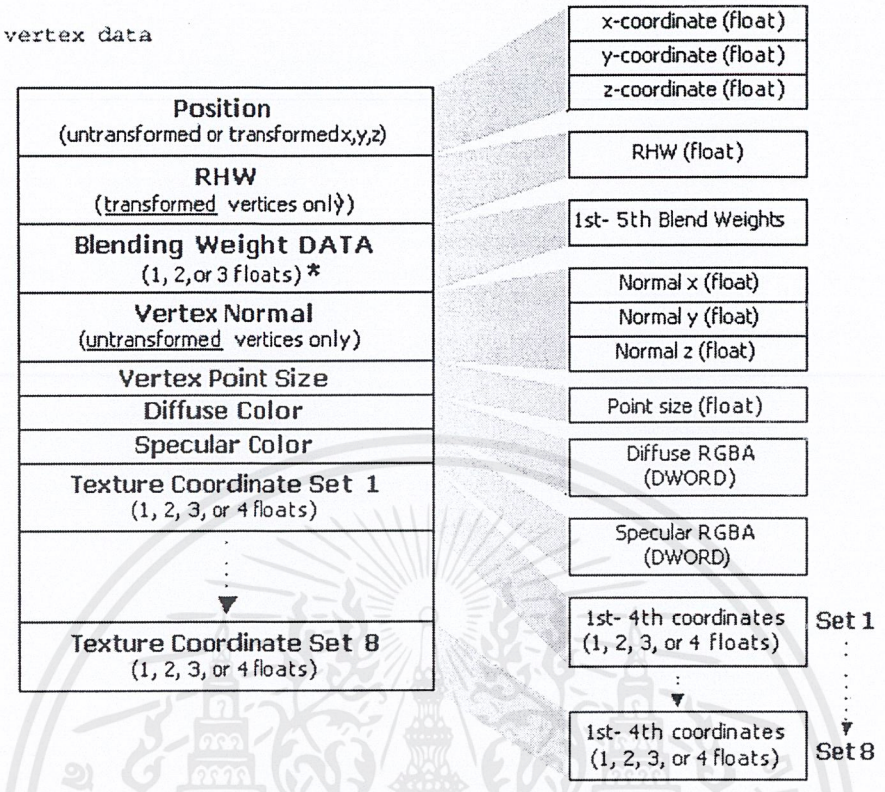
เนื้อหาในส่วนนี้จะประกอบไปด้วย

- Vertices ,Vector และQuaternion
- 3D Coordinate
- การสร้าง 3D Primitive Object
- การ Translation Object ,Rotation Object ,Scaling Object
- Matrix ชนิดต่างๆที่ใช้ในการอ้างอิงวัตถุ
- Lighting และ Material
- Texture Mapping
- Mesh

2.2 Vertices, Vector

Vertices คือ จุดในพิกัด X, Y, Z โน direct3d นั้นเราสามารถแบ่งจุดออกเป็นหลายชนิดโดยขึ้นอยู่กับข้อมูลที่เก็บอยู่ในจุดๆนั้นเช่น ใน 1Vertex อาจประกอบไปด้วย ข้อมูลของ พิกัด (X, Y, Z) มุมที่จะทำการหมุน(RHW) Normal Vector (เวกเตอร์ ที่ตั้งฉากกับพื้นผิวเพื่อช่วยในการคำนวณค่าแสงที่ตกกระทบ) Diffuse Color (ค่าสีของ Vertex ใดๆ มีค่าเป็น ค่า RBGA) หรืออาจมีค่าของ Speccular color (ค่าของแสงสีที่วัตถุสะท้อนออกมา) Blend Weight Data , Vertex point size ,Texture Coordinate set ดังรูป

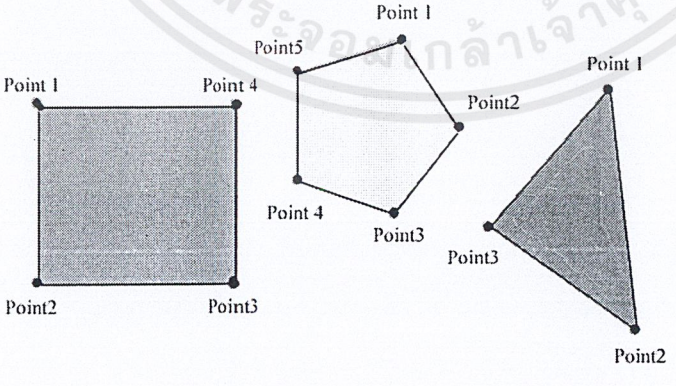
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.1 โครงสร้างข้อมูลของ vertice

2.3 Primitive Object

Polygon คือพื้นผิวเล็กๆที่เกิดขึ้นจากการสร้างพื้นผิวระหว่างจุดหลายๆจุดที่อยู่บนระนาบ X ,Y, Z มีหลายแบบเช่น รูปสี่เหลี่ยม รูปสามเหลี่ยม รูปห้าเหลี่ยม ดังรูป

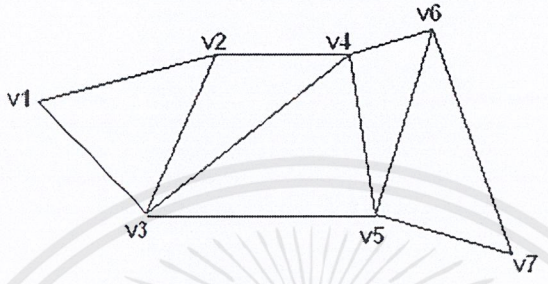


รูปที่ 2.2 Polygonชนิดต่างๆ

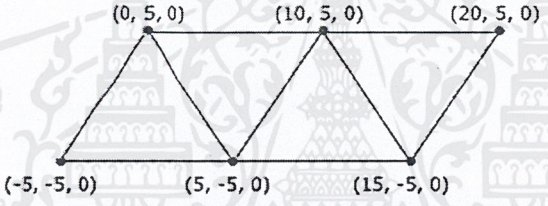
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

แต่รูปแบบที่นิยมใช้ในการสร้างรูปต่างๆ คือรูปสามเหลี่ยม ซึ่งสามารถเปลี่ยนแปลงรูปร่างได้อย่าง หลากหลาย และมีความซับซ้อนน้อยที่สุด ในการสร้างรูปทรงต่างๆ โดยเราสามารถแบ่งสามเหลี่ยม นี้ออกได้ 3 ชนิด คือ

2.3.1 Triangle Strip คือรูปสามเหลี่ยมต่อๆกันโดยสามเหลี่ยมที่สร้างขึ้นมาใหม่นั้นเกิดจากการลากเส้น ต่อระหว่างจุดที่เพิ่มเข้าไปใหม่กับ สองจุดหลังสุดที่ถูกจุดลงไปดังรูป

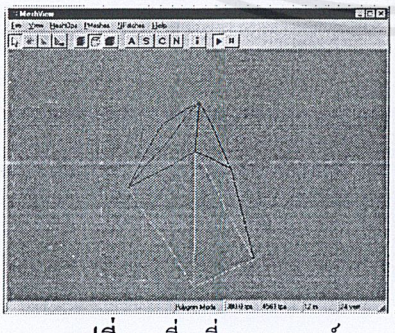


รูปที่ 2.3 Triangle Strip

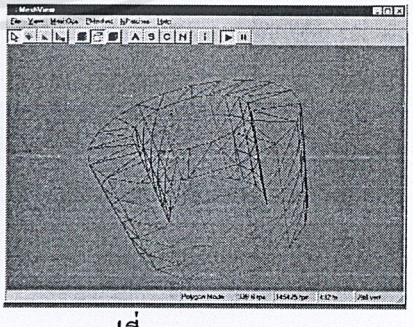


รูปที่ 2.4 Triangle Strip

จากรูปที่3 จะเห็นว่า Triangle Strip จะใช้จุดเพียงแค่ 7 จุดในการสร้างสามเหลี่ยม 5 รูป ซึ่งปกติเราใช้จุด 3 จุดสร้างสามเหลี่ยม 1 รูป ดังนั้นจึงต้องใช้จุดทั้งหมด 15 จุดในการสร้างสามเหลี่ยม 5 รูป จะเห็นได้ว่าประหยัดจุดกว่าปกติถึง 8 จุด โดยรูปสามเหลี่ยมแบบนี้เหมาะกับการสร้างรูปทรงประเภททรงกระบอก กล่องสี่เหลี่ยม ทรงกลม รูปหลอด

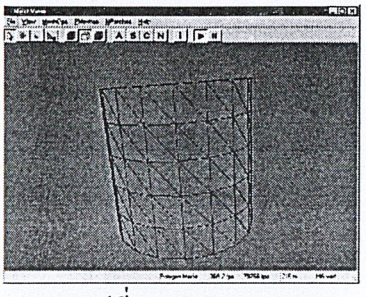


รูปที่ 2.5 สี่เหลี่ยมลูกบาศก์



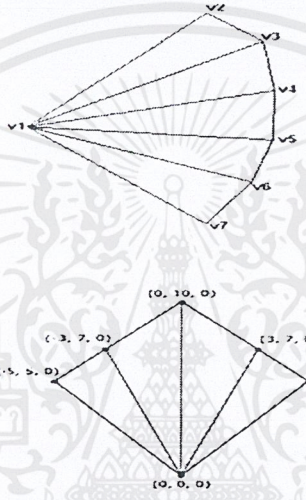
รูปที่ 2.6 หลอด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



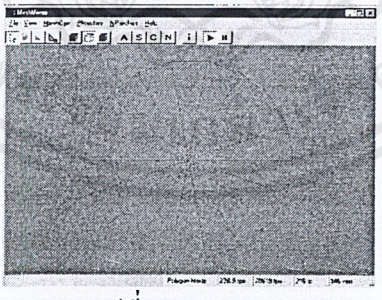
รูปที่ 2.7 ทรงกระบอก

2.3.2 Triangle fan เป็นรูปสามเหลี่ยมที่สร้างขึ้นโดยมีจุดร่วม 1 จุด คล้ายๆกับ Triangle Strip แต่จะสร้างสามเหลี่ยมจาก จุดหลังสุดที่จุดลงไปกับ จุดเริ่มต้น และ จุดใหม่ที่จุดลงไปดังรูป



รูปที่ 2.8 Triangle Fan

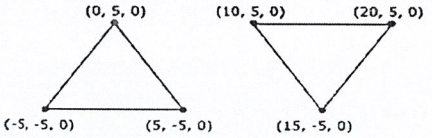
รูปสามเหลี่ยมประเภทนี้เหมาะสำหรับสร้าง รูปวงกลม รูปปีรามิด รูปกรวย ที่ไม่มีฝาปิด



รูปที่ 2.9 วงกลม

2.3.3 TriangleList เป็นสามเหลี่ยมที่มีความซับซ้อนน้อยที่สุดยืดหยุ่นที่สุดแต่เป็นสามเหลี่ยมที่ใช้จุดมากที่สุดในการสร้างรูปทรงต่างๆสามารถสร้างรูปทรงได้ทุกชนิดโดยสามเหลี่ยมที่สร้างขึ้นมานั้นจะติดกัน หรือไม่ติดกันก็ได้ดังรูป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.10 Triangle List

นอกจากสามเหลี่ยมพื้นฐาน 3 แบบนี้แล้วยังมี รูปแบบพื้นฐานอีก 3 อย่างคือ Point List ซึ่งส่วนใหญ่ใช้ทำหมอก หรือ Particle ต่างๆ Line Strip และ Line list ใช้สร้าง Objectที่ ลักษณะเป็นเส้น เช่น ผม หรือขนตา นอกจากนี้แล้วยังมี Object ที่ซับซ้อนมากขึ้นเช่น Spline , Patch ,Nurb

2.4 Matrix

จากการกำหนดจุดในแต่ละชนิดของสามเหลี่ยม พื้นฐานในหัวข้อที่แล้วเมื่อนำสามเหลี่ยมหลายๆรูปมาประกอบกันก็จะได้รูปทรงต่างๆขึ้น ในหัวข้อนี้ จะอธิบายเรื่องการเคลื่อนย้ายวัตถุที่สร้างขึ้น ซึ่งมีพื้นฐานการเคลื่อนที่อยู่ 3 รูปแบบคือ การ Translation ,Rotation และการ Scaling

2.4.1 Translation

คือการย้ายวัตถุที่สร้างขึ้นไปยังที่ใดๆบนระนาบ Global โดยสามารถอธิบายได้โดยสมการ

$$(x', y', z') = (x + \Delta x, y + \Delta y, z + \Delta z)$$

เมื่อ Δx คือระยะที่ต้องการเคลื่อนที่ในแนวแกน x

Δy คือระยะที่ต้องการเคลื่อนที่ในแนวแกน y

Δz คือระยะที่ต้องการเคลื่อนที่ในแนวแกน z

2.4.2 Rotation

คือการหมุนวัตถุที่สร้างขึ้นเป็นมุม θ รอบแกน X , Y หรือแกน Z โดยสามารถอธิบายได้โดยสมการดั่งนี้

หมุนรอบแกน X

$$(x', y', z') = (x, y\cos\theta - z\sin\theta, y\sin\theta + z\cos\theta)$$

หมุนรอบแกน Y

$$(x', y', z') = (x\cos\theta + z\sin\theta, y, -x\sin\theta + z\cos\theta)$$

หมุนรอบแกน Z

$$(x', y', z') = (x\cos\theta - y\sin\theta, x\sin\theta + y\cos\theta, z)$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.4.3 Scaling

คือการย่อขยายวัตถุในแนวแกนต่างๆ สามารถอธิบายได้โดยสมการดังนี้

$$(x', y', z') = (K_x x, K_y Y, K_z z)$$

โดย K_x คือจำนวนเท่าที่ต้องการย่อขยายตามแนวแกน x

K_y คือจำนวนเท่าที่ต้องการย่อขยายตามแนวแกน y

K_z คือจำนวนเท่าที่ต้องการย่อขยายตามแนวแกน z

จากพื้นฐานการเคลื่อนที่ทั้ง 3 แบบ ดังกล่าวเพื่อให้ง่าย ในทางปฏิบัติแล้ว เราจะกำหนด Matrixขึ้นมา เพื่อช่วยในการคำนวณการเคลื่อนที่ดังนี้

Translation Matrix

$$[x'y'z'1] = [x\ y\ z\ 1]\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ T_x & T_y & T_z & 1 \end{bmatrix}$$

Rotation Matrix around x-axis

$$[x'y'z'1] = [x\ y\ z\ 1]\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation Matrix around y-axis

$$[x'y'z'1] = [x\ y\ z\ 1]\begin{bmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation Matrix around z-axis

$$[x'y'z'1] = [x\ y\ z\ 1]\begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Scaling Matrix

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

$$[x'y'z'1]=[x\ y\ z\ 1]\begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

ถ้าต้องการการเคลื่อนที่มากกว่า 1 อย่างก็สามารถทำได้โดยการคูณเมตริก เช่นถ้าต้องการ เลื่อนจุดหมุนก็สามารถทำได้ดังนี้

ตัวอย่าง การหมุนสามเหลี่ยม สมมุติมีสามเหลี่ยมตั้งอยู่ที่พิกัด (5, 5, 0) ,(6, 6, 0) และ (7, 5, 0) เราต้องการ ให้สามเหลี่ยมนี้หมุนรอบตัวเอง สามารถทำได้โดย

1. ย้ายสามเหลี่ยมมาที่จุดศูนย์กลางของแกน โดยใช้ Matrix Translation โดยตำแหน่งลงมา 6 หน่วยแล้วย้ายไปทางซ้ายอีก 6 หน่วย ซึ่ง Matrix Translation มีลักษณะคือ

$$(x',y',z',1)=(x,y,z,1)\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -6 & -6 & 0 & 1 \end{bmatrix}$$

แล้วทำการคูณ Matrix ไปทีละจุดก็จะได้สามเหลี่ยมย้ายมาอยู่ที่ตำแหน่ง (-1, -1, 0) , (0, 0, 0) และ(1, -1, 0)

2.จากนั้นทำการหมุน วัตถุโดยคูณ Matrix Rotation เข้าไปดังนี้

$$(x',y',z',1)=(x,y,z,1)\begin{bmatrix} \cos t & \sin t & 0 & 0 \\ -\sin t & \cos t & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

จะได้สามเหลี่ยมที่หมุนรอบแกน Z ด้วยความเร็วเชิงมุม t

3.ย้ายตำแหน่งกลับคืนไปที่เดิมโดยคูณด้วย Matrix Translation ที่มีค่าเท่ากับ

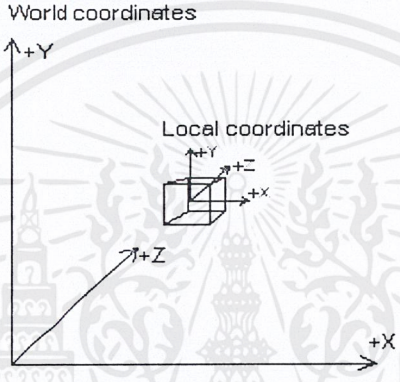
$$(x',y',z',1)=(x,y,z,1)\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 6 & 6 & 0 & 1 \end{bmatrix}$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จะได้สามเหลี่ยมที่หมุนรอบจุด(6, 6, 0) หรือจะทำการคูณ Matrix ทั้ง 3 ก่อนแล้วค่อยทำการคูณค่าเข้ากับจุดก็ได้

2.4.4 World ,View ,Projection Matrix

ในการนำวัตถุหลายๆชิ้นมาวางไว้ในระนาบเดียวกัน จะต้องมีการปรับค่าการอ้างอิงของแกนของแต่ละวัตถุให้อยู่แกนอ้างอิงเดียวกัน เรียกว่าการ Transformation Matrix โดยการหาว่าวัตถุเดิมที่ตั้งอยู่ในตำแหน่ง X1 ,Y1 ,Z1 เมื่อเทียบกับจุด 0 ,0 ,0 เดิมนั้นต้องตั้งอยู่ที่ใดเมื่อมีการเปลี่ยนแกนอ้างอิงไปเป็น X2,Y2,Z2 ซึ่งสามารถหาได้โดยการ นำค่าX1-X2,Y1-Y2,Z1-Z2 ก็จะได้พิกัดของจุดใหม่ในในแกนอ้างอิงใหม่ดังรูป

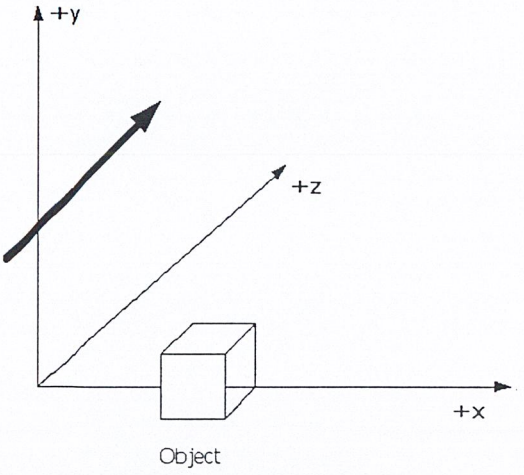


รูปที่ 2.11 World Coordinate

2.4.5 View Matrix

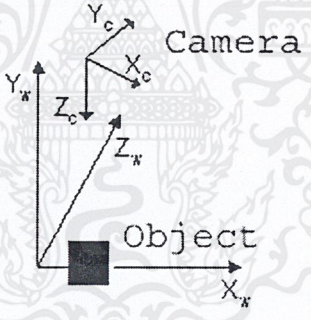
เมื่อทำการ Transformation Matrix ให้อยู่ใน World Matrix แล้ว ขั้นต่อไปต้องกำหนดตำแหน่งและจุดในการมองวัตถุ เรียกว่าการกำหนด View Transformation ซึ่งก็เป็นการเปลี่ยนจุดของ วัตถุอย่างหนึ่งแต่จริงๆแล้ววัตถุไม่ได้มีการเคลื่อนที่ไปด้วย เพียงแต่เป็นการบ่งบอกถึงตำแหน่งของภาพที่แสดงออกมาเท่านั้น ซึ่งก็คล้ายๆกับ การเปลี่ยนจาก Local axis ไปเป็น World axis แต่ View Transformation เป็นการเปลี่ยนจาก World Space ไปยัง Camera Space โดยมีการกำหนดให้ตำแหน่งของ Camera นั้นเป็นตำแหน่ง 0 ,0 ,0 เช่นถ้าวัตถุอยู่ที่ตำแหน่ง 0 ,0 ,0 และกล้องตั้งอยู่ที่ตำแหน่ง 10,20,10 นั่นหมายความว่าเราจะต้อง เลื่อนตำแหน่งวัตถุ ไปเป็นระยะ –10 ตามแนวแกน X เลื่อนตำแหน่งวัตถุไปเป็น –20 ตามแนวแกน Y และเลื่อนตำแหน่งวัตถุเป็น –10 ตามแนวแกน Z เป็นต้น โดยกล้องจะมอง ไปในระยะ +Z เสมอดังรูปถ้าไม่กำหนดจุดที่ตั้งกล้อง และ จุดที่กล้องมองให้ดีเราก็จะมองไม่เห็นวัตถุดังรูป ลูกศรสีดำแสดงถึงทิศทางการมอง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.12 การตั้งกล้องและตั้งทิศทางการมองที่ผิดพลาด

และเมื่อกำหนดจุดมองกล้องในลักษณะที่ เงยขึ้น หรือเงยลง ภาพที่เห็นเปรียบเสมือนการหมุนวัตถุ เช่นถ้าวัตถุตั้งอยู่ ณ.จุด 0 ,0 ,0 และกล้องตั้งอยู่เหนือวัตถุ ดังรูป ถ้ามิได้กำหนดจุดมองกล้อง ทิศทางการมองของกล้องจะมองไปในทิศทาง + Z ซึ่งเราจะมองไม่เห็นวัตถุแต่ถ้ากำหนดทิศทางในการมองวัตถุให้มองลงมาก็จะมองเห็นวัตถุซึ่งเปรียบเสมือนว่ามีการหมุนวัตถุขึ้นมานั่นเอง ดังนั้น View Transform Matrix ก็คือการ Translation กับ Rotation



รูปที่ 2.13 การตั้งทิศและที่ตั้งที่สามารถมองเห็นวัตถุได้

ซึ่ง เราสามารถอธิบาย View Matrix ได้จากสมการนี้

$$V = T \cdot R_z \cdot R_y \cdot R_x$$

เมื่อ T คือ Translation Matrix

R_x คือ Rotation Matrix ในแนวแกน X

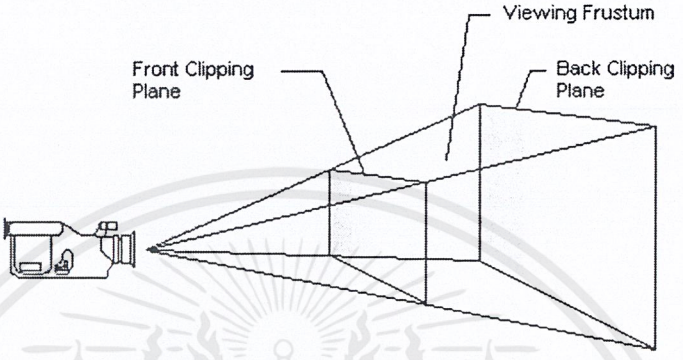
R_y คือ Rotation Matrix ในแนวแกน Y

R_z คือ Rotation Matrix ในแนวแกน Z

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.4.6 Projection Matrix

เมื่อวัตถุ 2 อันมีขนาดเท่ากัน ตั้งอยู่ในระยะทางที่ต่างกัน วัตถุที่อยู่ใกล้กว่า จะมองเห็นว่ามีขนาดใหญ่กว่า และวัตถุที่อยู่ไกลกว่าก็จะมองเห็นว่ามีขนาดเล็กกว่า Projection Matrix เป็น Matrix ที่ทำหน้าที่ในการ Scale ขนาดของวัตถุตามระยะทางที่ห่างจากจุดที่กล้องตั้งอยู่ มาตั้งไว้ที่เรตินาของดวงตาเราสามารถอธิบายได้คือการ Map ภาพจาก Back Clipping Plane ไปยัง Front Clipping Plane



รูปที่ 2.14 ลักษณะของภาพเมื่อมองผ่านกล้อง

ดังนั้นภาพที่จะปรากฏ บนกล้องจะอยู่ระหว่าง Front Clipping และ Back Clipping ซึ่งก็คือ Viewing Frustum นั้นเอง ซึ่งเราสามารถอธิบายการ Projection Transformation ได้ว่า เป็นการมอง วัตถุโดยผ่านหน้าต่าง ที่มีขนาดเท่ากับ Front Clipping และมีขอบเขตการมองอยู่ที่ Back Clipping ดังนั้นจะเห็นวัตถุที่อยู่ในปีรามิดเท่านั้น Projection Matrix มีลักษณะดังนี้

$$\begin{bmatrix} w & 0 & 0 & 0 \\ 0 & h & 0 & 0 \\ 0 & 0 & Q & 1 \\ 0 & 0 & -QZ_n & 0 \end{bmatrix}$$

โดย $$Q = \frac{Z_f}{Z_f - Z_n}$$

$$w = \frac{2Z_n}{V_w}$$

$$h = \frac{2Z_n}{V_h}$$

โดยที่ V_w , V_h คือความกว้างและความยาวของ ViewPort ใน Camera Space

Z_f , Z_n คือตำแหน่งของ Clipping plane อันไกล และอันใกล้ ตามลำดับ

สังเกตได้ว่า Projection Transformation คือการ Scaling และการ Translation

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.5 Material

เป็นการบรรยายว่า พื้นผิว Polygon มีการสะท้อนแสงอย่างไร มีการดูดกลืนหรือมีการปล่อยแสงออกมาอย่างไร คุณสมบัติของ Material ประกอบไปด้วยคุณสมบัติดังนี้

2.5.1 Diffuse Reflect

เป็นลักษณะของการสะท้อนแสงที่ตกลงบนพื้นผิววัตถุแล้ว ทำให้เกิดการสะท้อนออกมาทุกทางด้วยปริมาณที่เท่ากัน การสะท้อนแสงในลักษณะนี้ในธรรมชาติเกิดขึ้นจาก การที่แสงตกกระทบวัตถุที่มีผิวไม่เรียบ หรือไปตกกระทบกับรงค์วัตถุในวัตถุนั้นซึ่งมีขนาดเล็ก และมีการกระจายอยู่ทั่วไป และมีทิศทางที่แตกต่างกันออกไปจึงทำให้แสงสะท้อนออกไปทุกทิศทุกทางโดยในขณะที่แสงตกกระทบแสงบางความถี่จะถูกดูดกลืนเข้าไป ดังนั้นสีของแสงที่สะท้อนในลักษณะนี้จึงมักเป็นสีที่ได้รับอิทธิพลมาจากสีของวัตถุ

2.5.2 Diffuse Transmission

จะมีลักษณะคล้าย Diffuse Reflect แต่แสงที่มาตกกระทบนั้นจะทะลุผ่านออกไปอีกด้านหนึ่งที่ตรงข้ามกับด้านที่แสงตกกระทบ เช่นกรณีของการตกกระทบในพลาสติกแผ่นบางๆที่มีความขุ่น

2.5.3 Specular Reflect

เป็นลักษณะการสะท้อนแสงที่แสงจะสะท้อนกลับออกมาเป็นมุมเดียวกับมุมที่กระทบโดยเทียบกับ Normal vector ของพื้นผิวที่จุดนั้นๆ

2.5.4 Specular Transmission

แสงที่มากระทบวัตถุนั้นไม่จำเป็น บางทีอาจทะลุผ่านวัตถุไปได้ โดยแสงที่ทะลุเข้ามานั้นจะมีการหักเหตามความหนาแน่นของวัตถุ

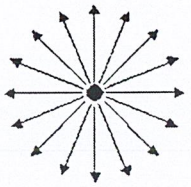
2.6 Lighting

ชนิดของ light ที่ใช้ปกติทั่วไปมีอยู่ 3 ชนิดคือ

2.6.1 Point light

คือแสงชนิดเป็นจุดคล้ายหลอดไฟ โดยจะส่องแสงออกมาทุกทิศทุกทางดังรูป

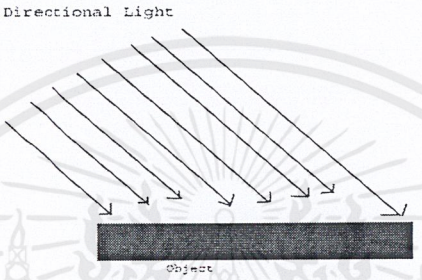
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.15 Point light

2.6.2 Directional Light

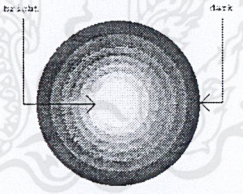
คือแสงชนิดที่ส่องขนานๆกันลงมาลงทั่งทั้งบริเวณที่ กำหนดดังรูป



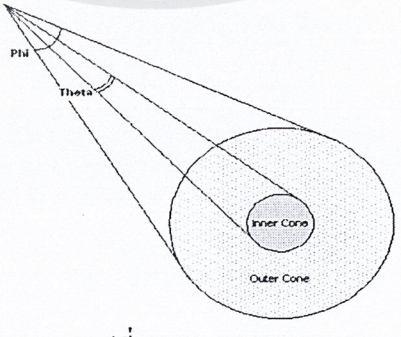
รูปที่ 2.16 Directional Light

2.6.3 Spot Light

คือแสงที่ส่องลงมาจากจุดใดจุดหนึ่งโดยมีความสว่างจากในมากที่สุดและที่ขอบน้อยที่สุดดังรูป



รูปที่ 2.17 ลักษณะของแสงประเภท Spot light

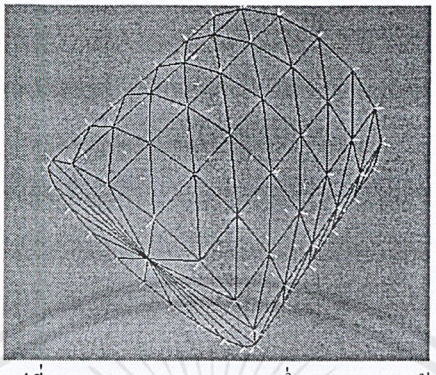


รูปที่ 2.18 Spot light

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.7 Normal vector

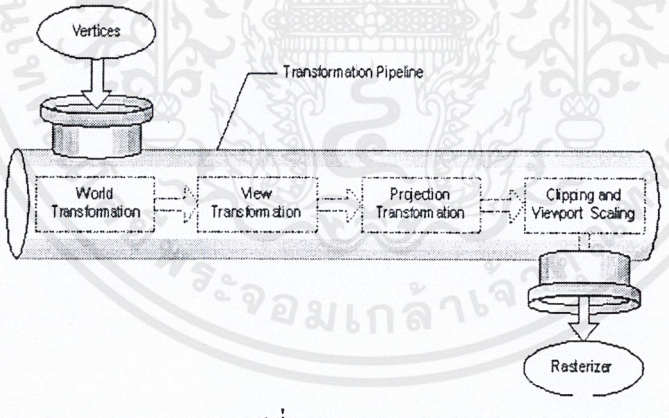
เป็นเวกเตอร์หนึ่งหน่วยที่ตั้งฉากกับพื้นผิวเกิดจากการรวมกันของเวกเตอร์ในแนวแกน X X , Y และ Z ที่มีขนาดต่างกันช่วยในการคำนวณการให้สีของการตกกระทบของแสงดังรูป



รูปที่ **2.19** Normal Vector ซึ่งแสดงจากเส้นสีเหลือง

การให้ Normal vector ที่ถูกต้อง จะช่วยให้วัตถุดูสมจริงยิ่งขึ้น สังเกตที่ขอบของทรงกระบอก จะมี normal vector 2 อันมีทิศทางตั้งฉากกันจะทำให้ภาพที่ออกมานั้นจะมีขอบที่คม

2.8 Render Pipeline



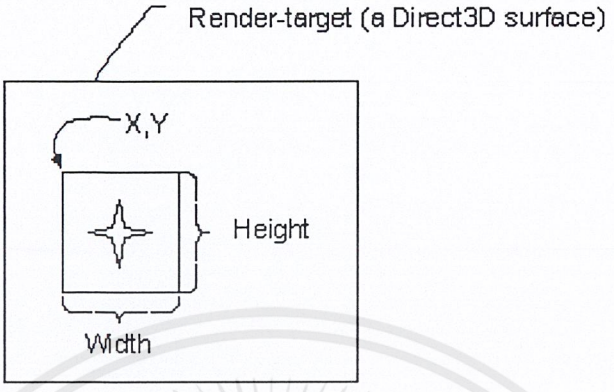
รูปที่ **2.20** Render Pipeline

จากรูปเป็นขั้นตอนการประมวลผลจากกลุ่ม Vertice ที่เราสร้างขึ้นมาจนเป็น polygon แล้ว ผ่านไปยัง ขั้นตอน World Transform ----------→ View Transform ----------→ Projection Transform ----------→ Clipping and Viewport Scaling ----------→ Rasterizer

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.9 Clipping And Viewport Scaling

คือการกำหนดขอบเขตการมองเห็นว่าให้เห็นรูปภาพมีขนาดเป็นอย่างไรคล้ายๆกับเป็นการกำหนดผืนผ้าที่จะ Render ภาพลงไปโดยต้องกำหนดค่า X,Y และความกว้างความยาวดังรูป



รูปที่ **2.21** Viewport

สามารถอธิบายได้โดย การ Transform Matrix ตามนี้

$$\begin{bmatrix} lWidth/2 & 0 & 0 & 0 \\ 0 & -lHeight/2 & 0 & 0 \\ 0 & 0 & maxz - minz & 0 \\ lX + lWidth/2 & lHeight/2 + lY & minz & 1 \end{bmatrix}$$

โดย Minz,Maxz คือระยะ ต่ำสุดของ Z และสูงสุดของ Z ที่ต้องการ Render

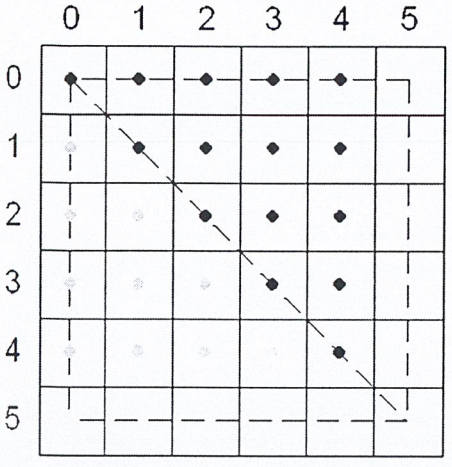
lX , lY คือจุดกำหนดพื้นที่ๆจะ Render

lWidth , lHeight คือ ความกว้างและความยาวของ Viewportที่จะ Render

2.10 Rasterize

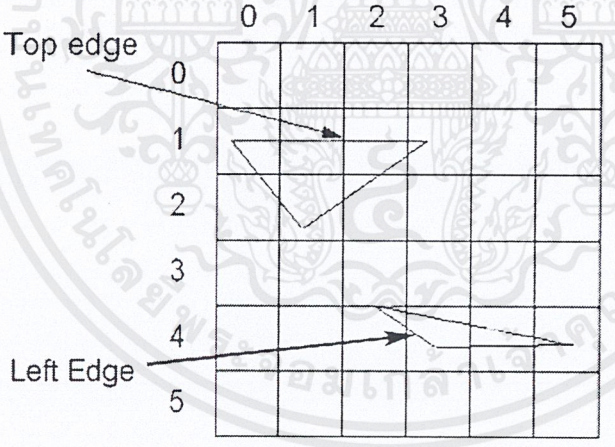
คือ ขั้นตอนสุดท้ายในการประมวลผลภาพ โดยทำการเปลี่ยนข้อมูลของ Object ที่อยู่ในรูปแบบ Array ของค่าพิกัดต่างๆและค่าสี ให้เป็น รูปภาพ 2 มิติที่ปรากฏบนจอคอมพิวเตอร์ ที่มีลักษณะเป็น Pixel ซึ่งบางครั้ง สามเหลี่ยมที่ Map มานั้นไม่พอดีกับ Pixel โดยต้องมีการตัดสินว่า Pixel ใดควรจะรับค่าจากสามเหลี่ยมใด ดังนั้นจึงต้องมีกฎในการ Map ค่าเรียกว่าการ Triangle Rasterization โดยมีหลักในการพิจารณา คือ ถ้า Pixel ใดอยู่ใน Polygon ในก็ให้ Map ค่าจาก Polygon นั้นโดยตัดสินจากจุดศูนย์กลางของ Pixel ว่าจุดศูนย์กลางอยู่ภายใน Polygon ใดก็ให้ Map จาก Pixel นั้นแต่ถ้ามี Polygon 2 อันวางอยู่ในลักษณะดังรูป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.22 แสดงวิธีการกำหนดการ Map ภาพไปยัง Polygon

จะเห็นสามเหลี่ยมทั้งคู่ผ่านจุดกึ่งกลางของ Pixel ในกรณีนี้จะมีวิธีการตัดสินด้วยกฎ top-left filling หมายความว่า Pixel ที่อยู่ระหว่าง Polygon 2 อันจะทำการ Map กับ Polygon ที่เป็นเจ้าของขอบทางด้านซ้ายหรือด้านบนเชื่อมกับอีก Polygon หนึ่งจากรูปจุดสีดำหมายถึง Pixel นั้นเป็นของสามเหลี่ยม อันบนขวาส่วนจุดเทาจะเป็น Pixel ที่ Map กับสามเหลี่ยมอันล่างซ้าย โดยจะเรียก Polygon ที่เป็นเจ้าของขอบทางด้านซ้ายหรือด้านบนว่า Left edge กับ Top edge ดังรูป



รูปที่ 2.23 Top-Left edga

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

2.11 Shading

คือการเกลี่ยค่าสีของ Pixel ให้มีความต่อเนื่องขึ้น มีหลายประเภทแต่ในที่นี้จะอธิบายถึง Gouraud Shading โดยจะเปรียบเทียบกับ Flat Shading จะเห็นว่า Gouraud Shading นั้นจะดูมีความโค้งมนมากกว่า เพราะมีการ Interpolation กันของค่า Intensity ของ Polygon โดยจะทำการเกลี่ยค่าความเข้มของแสงสีระหว่างจุดของ Polygon ทั้ง 3 จุดก่อนจากนั้นทำการเกลี่ยสีภายในตัว polygon แล้วค่อยนำมา Rasterization ให้เป็น Pixel จะทำให้ภาพดูสมจริงยิ่งขึ้นส่วน Flat shading นั้นจะทำการ Map ค่าแสงสีลงไปใน แต่ละ pixel โดยตรงซึ่งค่าแสงสีที่ Map ลงไปนั้นมีการเปลี่ยนแปลงอย่างทันทีทันใดทำให้ Object ดูเป็นเหลี่ยมๆดังรูป นอกจากจะมี Flat , Gourade แล้วยังมี Phong shading อีกโดยจะมีการ Interpolate ค่า Normal vector ลงไปก่อนแล้วค่อยทำการ Interpolate ค่า แสงสีซึ่งจะทำให้วัตถุมีความสมจริงมากขึ้นไปอีกระดับแต่ก็จะเวลาในการประมวลผลมากขึ้นเช่นกัน



รูปที่ 2.24 Flat Shading



รูปที่ 2.25 Gourade Shading

2.12 Texture Mapping

เป็นการ Map รูปภาพลงบนวัตถุ แล้วค่อยคำนวณค่าแสงสี (Diffuse , Specular)เพื่อให้มีลวดลายสมจริงยิ่งขึ้น โดยจะอธิบายรายละเอียด เรื่องอีกครั้งในหัวข้อ การเขียน Program โดยใช้ Direct3d

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

45874

บทที่ 3

DirectX

เป็น Window API ของไมโครซอฟท์พัฒนาเพื่อใช้จัดเตรียมอินเตอร์เฟสสำหรับควบคุมฮาร์ดแวร์มัลติมีเดียบนระบบ Windows และเป็นเครื่องมือให้ ทำงานกับคำสั่งและโครงสร้างข้อมูลในระดับใกล้ชิดฮาร์ดแวร์ โดยไม่ต้องสร้างโค้ดติดต่อในระดับล่าง ซึ่งวิธีติดต่อจะแตกต่างกันไปตามประเภทของอุปกรณ์ การเขียนโปรแกรมที่เป็นอิสระจากอุปกรณ์ลักษณะนี้ ช่วยให้สามารถสร้างซอฟต์แวร์เพื่อทำงานดังกล่าวได้เป็นอย่างดี แม้จะมีการเปลี่ยนแปลงตัวอุปกรณ์ตัวใหม่ก็ตาม

โปรแกรมที่เขียนด้วย DirectX สามารถ Run ได้อย่างรวดเร็วด้วย API ของ DirectX ซึ่งสร้างขึ้นมาบนพื้นฐานของ HAL (Hardware Abstraction Layer) สามารถซ่อนลักษณะของดีไวซ์ที่อยู่กับฮาร์ดแวร์ และยังได้รับการออกแบบให้มีความสามารถเพิ่มขยายได้ในอนาคต

ส่วนถ้าตัวการ์ดจอแสดงผลไม่มี Function ที่โปรแกรมที่เขียนขึ้นต้องการเรียกใช้ ก็มีโปรแกรมจำลองการทำงานของการ์ดจอผ่านทาง Hel(Hardware Emulation Layers)

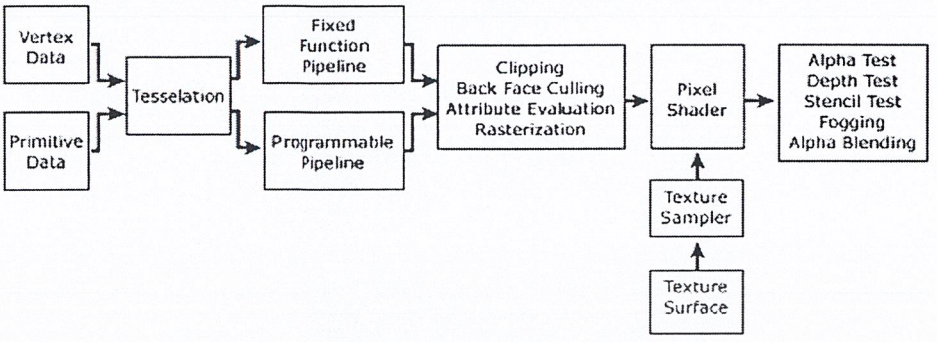
โดยตัว Directx นั้นประกอบไปด้วย API ประเภทต่างๆเพื่อใช้ในการติดต่อ กับ ฮาร์ดแวร์ชนิดต่างๆที่ต่ออยู่กับ Computer เช่น Soundcard , Joystick และการ์ดจอดังนี้

Direct Graphic และ Direct3D เป็นตัวที่ช่วยในการเขียนโปรแกรมกราฟฟิกเพื่อติดต่อใช้งานฮาร์ดแวร์ทั้งในการวาดเพื่อแสดงผล ควบคุมข้อมูล Bitmap หน่วยความจำ การใช้งานตัวเร่งความเร็ว 3 มิติ การ Flipping และ การ Blitting ภาพ Direct3D มีการทำงานอยู่ 2 โหมดด้วยกัน คือ โหมด Immediate ซึ่งเป็นโหมดที่มีความซับซ้อนในการใช้งานแต่มีความยืดหยุ่นและสามารถ Run Application ได้เร็วมาก และ โหมด Retain ซึ่งสามารถใช้งานได้ง่ายกว่าแต่มีขีดจำกัดของการทำงานมากกว่า Run Application ได้ช้ากว่า Immediate Mode โดย Retain Mode นั้นทาง Microsoft ได้เลิกการพัฒนาไปแล้วตั้งแต่ Directx 6 ดังนั้นจึงได้มีการศึกษาการทำงานใน Immediete Mode เท่านั้นนอกจาก Direct3D แล้วยังมี API อื่นๆที่ช่วยในการอินเตอร์เฟสกับฮาร์ดแวร์เช่น DirectPlay DirectSound ,DirectInput ,Directmusic

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

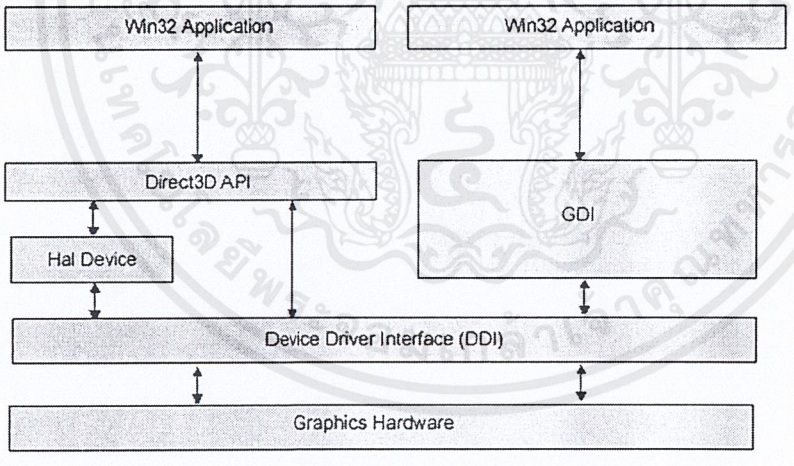
3.1 การทำงานของ Direct3D

มีลำดับขั้นตอนดังรูป



รูปที่ 3.1 ขั้นตอนการ Render

การ Tessellation คือการแตกภาพออกเป็นรูปสี่เหลี่ยมเล็กๆเพื่อใช้ในการประมวลผลส่วน Fix Function Pipeline และ Programmable pipeline จะเป็นส่วนการ Transformation Pipeline และ Pixel Shader เป็นกระบวนการคำนวณค่าสีเกลี่ยค่าสีให้กับตัว วัตถุโดยอาจรับค่าสีมาจาก Texture ใดๆหรือ ถ้าไม่มีการกำหนดการ Mapping Texture ลงไปในวัตถุก็จะทำการ Shading ให้กับตัว Material ของวัตถุเท่านั้นและผ่านไปยังขั้นตอนสุดท้ายซึ่งเป็นการRenderโดยวิธีพิเศษต่างๆเช่นเงา Lens,Flare,Fog



รูปที่ 3.2 ความสัมพันธ์ระหว่าง GDI กับ Hal

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.2 การสร้างภาพใน Direct3D8

มีขั้นตอนคือ

1. สร้างอุปกรณ์หรือเครื่องมือที่ใช้ในการวาดภาพ และจัดการข้อมูลต่างๆ เตรียมเงื่อนไขที่ต้องการใช้
2. สร้างภาพที่ต้องการจะนำมาแสดง
3. กำหนดตำแหน่งที่ต้องการจะวางภาพ และกำหนดตัวแปรที่ใช้ในการ Transformation Matrix
4. กำหนด Lighting และ Material
5. ทำการ Mapping Texture ลงบนวัตถุ
6. ทำการ Render

3.2.1 การสร้างอุปกรณ์หรือเครื่องมือที่ใช้ในการวาดภาพ และจัดการข้อมูล

เป็นขั้นตอนแรกในการสร้างภาพโดยต้องทำการสร้าง Object แม่แบบก่อนคือ Object Directx8 ซึ่งเป็น Object แม่ซึ่งจะเป็นตัวสร้าง Object ทุกชนิดที่มีใน DirectX API นี้ ต่อมาให้สร้าง Object Direct3D8 ซึ่งเป็น Object แม่ของ Object ทุกตัวที่เกี่ยวกับการสร้างภาพ ขั้นตอนต่อมาคือการสร้าง Device ที่ใช้ในการ Render ภาพ คือ Direct3DDevice8 สามารถเขียนเป็น Code ใน Visual Basic ได้ดังนี้

```
Dim g_DX As New DirectX8                    เป็นการกำหนดชนิดของตัวแปร
Dim g_D3D As Direct3D8                      เป็นการกำหนดชนิดของตัวแปร
Dim g_D3DDevice As Direct3DDevice8          เป็นการกำหนดชนิดของตัวแปร
Set g_D3D = g_DX.Direct3DCreate()           ทำการสร้าง Object Direct3D8
Dim  d3dpp As D3DPRESENT_PARAMETERS
```

ทำการสร้างDeviceที่ใช้ในการ Render โดยใช้ Method Createdevice

```
Set g_D3DDevice = g_D3D.CreateDevice(D3DADAPTER_DEFAULT, D3DDEVTYPE_HAL,
hWnd, _ D3DCREATE_SOFTWARE_VERTEXPROCESSING, d3dpp)
```

ค่าตัวแปรต่างๆมีความหมายดังนี้

D3DADAPTER_DEFAULT เป็นการ set ให้ Device(g_D3Ddevice)เป็นตัวหลักในการทำงาน

D3DDEVTYPE_HAL เป็นการบอกชนิด ว่าใช้การประมวลผลแบบใด(มี Hal กับHel)

HWnd เป็นตัวแปรที่บอกค่า handle ว่าเป็นของ Objectใด

D3DCREATE_SOFTWARE_VERTEXPROCESSING เป็น flag ที่ใช้ในการกำหนดพฤติกรรมของ Device นอกจากนี้ยังมี D3DCREATE_HARDWARE _VERTEXPROCESSING D3DCREATE_MIXED_VERTEXPROCESSING จะต้อง Set เป็นค่าหนึ่งใน 3 ค่านี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

D3dpp เป็นค่า Parameter ที่กำหนดลักษณะของ Device ที่สร้างขึ้น จะเป็นตัวกำหนดคุณสมบัติเกี่ยวกับ ความกว้าง ความยาว ของ Window ทั้งใน Window mode และใน Full Screen mode และกำหนดว่า Render ใน mode full screen หรือ window กำหนดวิธีการ Flipping ภาพ กำหนดการใช้ Zbuffer และอีกหลายอย่างที่จำเป็น โดยมีลักษณะข้อมูลดังนี้

```
Type D3DPRESENT_PARAMETERS
  AutoDepthStencilFormat As CONST_D3DFORMAT
  BackBufferCount As Long                          จำนวนของ backbuffer
  BackBufferFormat As CONST_D3DFORMAT              รูปแบบของ back buffer
  BackBufferHeight As Long          ความสูงของ Back buffer(ในfull mode screen mode)
  BackBufferWidth  As Long          ความกว้างของ back buffer(ในfull mode screen mode)
  EnableAutoDepthStencil As Long
  Flags As Long                                    set เป็น 0
  FullScreen_PresentationInterval As Long
  FullScreen_RefreshRateInHz As Long               ค่าความถี่ refresh หน้าจอ
  hDeviceWindow As Long                            ค่า Handle ของ Object
  MultiSampleType As CONST_D3DMULTISAMPLE_TYPE
  SwapEffect As CONST_D3DSWAPEFFECT                คือลักษณะของการ Flip page
  Windowed As Long                                 ถ้าเป็น window mode set 1
End Type
```

ต่อมาทำการ set Render state ตามความจำเป็น จาก Method SetRenderState เช่น การกำหนด ให้มี Z-buffer หรือ กำหนดให้ใช้ Lighting (ควรจะ Set)มีวิธีดังนี้

```
D3DDevice.SetRenderState D3DRS_ZENABLE, 1
D3DDevice.SetRenderState D3DRS_LIGHTING, 1
```

นอกจากนี้ยังมีการ Set ค่าต่างๆ นอกเหนือจาก Zbuffer , lighting อีกเช่น

```
D3DRS_CLIPPING                  = 136 (&H88)
D3DRS_LIGHTING                  = 137 (&H89)
D3DRS_AMBIENT                   = 139 (&H8B)
D3DRS_FOGVERTEXMODE             = 140 (&H8C)
D3DRS_COLORVERTEX               = 141 (&H8D)
D3DRS_LOCALVIEWER               = 142 (&H8E)
D3DRS_NORMALIZENORMALS          = 143 (&H8F)
D3DRS_DIFFUSEMATERIALSOURCE     = 145 (&H91)
D3DRS_SPECULARMATERIALSOURCE    = 146 (&H92)
D3DRS_AMBIENTMATERIALSOURCE     = 147 (&H93)
D3DRS_EMISSIVEMATERIALSOURCE    = 148 (&H94)
D3DRS_VERTEXBLEND               = 151 (&H97)
D3DRS_CLIPPLANEENABLE           = 152 (&H98)
D3DRS_SOFTWAREVERTEXPROCESSING = 153 (&H99)
D3DRS_POINTSIZE                 = 154 (&H9A)
D3DRS_POINTSIZE_MIN             = 155 (&H9B)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
D3DRS_POINTSPRITEENABLE        = 156 (&H9C)
D3DRS_POINTSCALEENABLE         = 157 (&H9D)
D3DRS_POINTSCALE_A             = 158 (&H9E)
D3DRS_POINTSCALE_B             = 159 (&H9F)
D3DRS_POINTSCALE_C             = 160 (&HA0)
D3DRS_MULTISAMPLEANTIALIAS     = 161 (&HA1)
D3DRS_MULTISAMPLEMASK          = 162 (&HA2)
D3DRS_PATCHEDGESTYLE           = 163 (&HA3)
D3DRS_PATCHSEGMENTS            = 164 (&HA4)
D3DRS_DEBUGMONITORTOKEN        = 165 (&HA5)
D3DRS_POINTSIZE_MAX            = 166 (&HA6)
D3DRS_INDEXVERTEXBLENDENABLE   = 167 (&HA7)
D3DRS_COLORWRITEENABLE         = 168 (&HA8)
D3DRS_TWEENFACTOR              = 170 (&HAA)
D3DRS_BLENDOP                  = 171 (&HAB)
D3DRS_ZWRITEENABLE             =  14
D3DRS_ALPHATESTENABLE          =  15
D3DRS_LASTPIXEL                =  16 (&H10)
D3DRS_SRCBLEND                 =  19 (&H13)
D3DRS_DESTBLEND                =  20 (&H14)
D3DRS_CULLMODE                 =  22 (&H16)
D3DRS_ZFUNC                    =  23 (&H17)
D3DRS_ALPHAREF                 =  24 (&H18)
D3DRS_ALPHAFUNC                =  25 (&H19)
D3DRS_DITHERENABLE             =  26 (&H1A)
D3DRS_ALPHABLENDENABLE         =  27 (&H1B)
D3DRS_FOGENABLE                =  28 (&H1C)
D3DRS_SPECULARENABLE           =  29 (&H1D)
D3DRS_ZVISIBLE                 =  30 (&H1E)
D3DRS_FOGCOLOR                 =  34 (&H22)
D3DRS_FOGTABLEMODE             =  35 (&H23)
D3DRS_FOGSTART                 =  36 (&H24)
D3DRS_FOGEND                   =  37 (&H25)
D3DRS_FOGDENSITY               =  38 (&H26)
D3DRS_EDGEANTIALIAS            =  40 (&H28)
D3DRS_ZBIAS                    =  47 (&H2F)
D3DRS_RANGEFOGENABLE           =  48 (&H30)
D3DRS_STENCILENABLE            =  52 (&H34)
D3DRS_STENCILFAIL              =  53 (&H35)
D3DRS_STENCILZFAIL             =  54 (&H36)
D3DRS_STENCILREF               =  57 (&H39)
D3DRS_STENCILMASK              =  58 (&H3A)
D3DRS_STENCILWRITEMASK         =  59 (&H3B)
D3DRS_TEXTUREFACTOR            =  60 (&H3C)
```

แล้วแต่ว่าต้องใช้ Function อะไรบ้าง

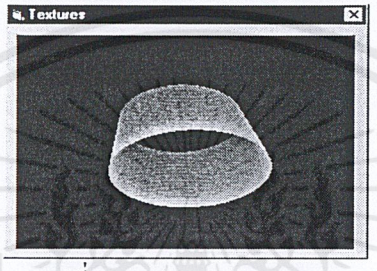
3.2.2 วิธีการสร้างภาพ มี 2 วิธี

1. แบบ manual เป็นการสร้างภาพขึ้นมาโดยกำหนดจุดขึ้นมาทีละจุดโดยกำหนดว่าใช้ Primitive Object แบบใด(Triangle list , Triangle strip, Triangle fan) โดยเก็บไว้ใน Array ตัวหนึ่งชื่อว่า Vertexซึ่ง vertex นี้ ต้องมีข้อมูลของพิกัดXYZและเป็นอย่างน้อย(อาจมีค่าNormal vector ค่าสีของจุด,หรือ ค่าพิกัดของ Texture ด้วยก็ได้) ตัวอย่างการสร้างภาพทรงกระบอกจาก Triangle Strip

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
For i = 0 To 49
    theta = (2 * g_pi * i) / (50 - 1)
    Vertices(2 * i + 0).postion = vec3(Sin(theta), -1, Cos(theta))
    Vertices(2 * i + 0).color = &HFFFFFFFF  'white
    Vertices(2 * i + 1).postion = vec3(Sin(theta), 1, Cos(theta))
    Vertices(2 * i + 1).color = &HFF808080  'grey
    Next
```

ตัวแปล vertex เป็น array ของ ค่า XYZ และ ค่าสี(เป็น ตัวเลขฐาน16 แสดงถึงค่า RAGB) จะได้ทรงกระบอกมีรัศมี 1 หน่วย สูง 2 หน่วยไม่มีฝาปิดทั้งสองด้าน ดังรูป



รูปที่ 3.3 รูปทรงกระบอก

ตัวอย่างการสร้าง Box จาก Triangle Strip

```
Vertices2(0).postion = vec3(-2, 0, -1.6)
Vertices2(1).postion = vec3(-2, 0, 1.6)
Vertices2(2).postion = vec3(20, 0, -1.6)
Vertices2(3).postion = vec3(20, 0, 1.6)
Vertices2(4).postion = vec3(20, 1.2, -1.6)
Vertices2(5).postion = vec3(20, 1.2, 1.6)
Vertices2(6).postion = vec3(-2, 1.2, 1.6)
Vertices2(7).postion = vec3(-2, 1.2, -1.6)
Vertices2(8).postion = vec3(-2, 0, 1.6)
Vertices2(9).postion = vec3(-2, 0, -1.6)
Vertices2(10).postion = vec3(20, 0, -1.6)
Vertices2(11).postion = vec3(-2, 1.2, -1.6)
Vertices2(12).postion = vec3(20, 1.2, -1.6)
Vertices2(13).postion = vec3(-2, 1.2, 1.6)
Vertices2(14).postion = vec3(20, 1.2, 1.6)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

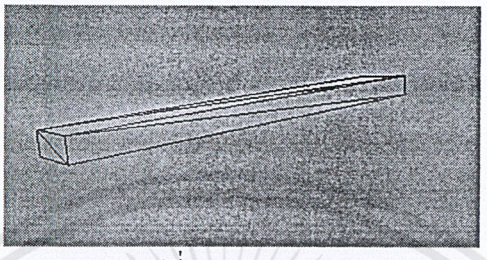
Vertices2(15).postion = vec3(-2, 0, 1.6)

Vertices2(16).postion = vec3(20, 0, 1.6)

Vertices2(17).postion = vec3(20, 1.2, -1.6)

Function Vec3 เป็น Function ช่วยในการกำหนดค่าของ X ,Y ,Z

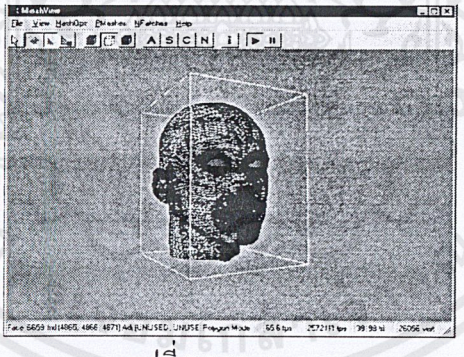
จะได้รูปกล่องสี่เหลี่ยมที่มีขนาด ความกว้างเท่ากับ3.2 หน่วย และมีความยาวและความสูงเท่ากับ 22 ,1.2 หน่วยตามลำดับแสดงได้ดังรูป



รูปที่ 3.4 รูปกล่อง

ซึ่งเราจะเห็นว่าวิธีนี้เป็นวิธีที่ยุ่งยากมากเพราะถ้าไม่มีความรู้ทางคณิตศาสตร์ก็ต้อง Plot ค่าจุดลงไปทีละจุด

2.การใช้ Mesh เป็นวิธีการสร้างภาพขึ้นมาก่อนโดยโปรแกรมสำเร็จรูป เช่น 3DStudio Max หรือ Maya แล้ว export เป็น X-file.X แล้วทำการ Import เข้ามาใน Program ซึ่งทำให้ได้ภาพที่มีรูปทรงซับซ้อนมากกว่าวิธีแรกดังรูป

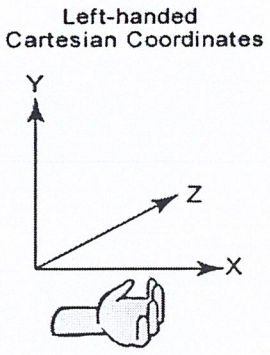


รูปที่ 3.5 Mesh

3.2.3 การกำหนดตำแหน่งของวัตถุ

เมื่อสร้างวัตถุขึ้นแล้ว วัตถุที่สร้างขึ้นก็จะอยู่ในตำแหน่งตามที่ได้กำหนดไว้ เช่นถ้าสร้างสามเหลี่ยม 1 รูปอยู่ที่ (0,1,0) , (-1,0,0) ,(1,0,0) ก็จะได้สามเหลี่ยมตั้งอยู่ตามพิกัดที่ได้กำหนดไว้โดยแกนอ้างอิงที่ใช้ใน Direct3D มีลักษณะดังรูปคือระยะบวก X มีทิศไปทางขวาระยะบวกYมีทิศทางขึ้นและระยะบวก Z พุ่งเข้าสู่หน้าจอ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.6 แสดงถึง Coordinate ที่ใช้ใน Direct3D

ในกรณีที่มีวัตถุมากชิ้นขึ้น การอ้างอิงตำแหน่งจะทำได้ยากทั้งในการสร้างและการเคลื่อนย้าย จึงต้องใช้ Matrix ในการช่วยการอ้างอิงตำแหน่ง โดยแต่ละวัตถุจะมี Matrix เป็นของตัวเองหรือมีแกนอ้างอิงเป็นของตัวเองซึ่งจะทำการเคลื่อนย้าย การหมุน การปรับขนาดทำได้ง่ายขึ้น การอ้างอิงตำแหน่งวัตถุนั้นจะอ้างอิงกับ World Space เป็นแกนหลักจากนั้นทำการนำวัตถุไปวางใน World Space โดยใช้คำสั่ง D3Dmetrix .SETTRANSFORM D3DTS_WORLD, matWorldโดย Matworld คือ Matrix ที่ระบุว่าวัตถุอยู่ในตำแหน่งใด โดยมีค่า Default เป็น

$$\text{MatWorld} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

ถ้าต้องการวางวัตถุไว้ที่ ตำแหน่ง (1,1,1) ก็จะได้ Matrix Matworld เป็น $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$

หรือใช้คำสั่ง D3DXMatrixTranslation Matworld ,x ,y ,z ซึ่งมีหลักการเหมือนกับ การ Translation Matrix โดย Matrix 1 อันควรใช้กำหนดตำแหน่ง Object หนึ่งอันเพื่อให้ง่ายต่อการเปลี่ยนแปลงในภายหลัง

เมื่อทำการ Map ตำแหน่งวัตถุไปยัง World Space แล้วต่อไปต้องกำหนด View Matrix ซึ่งเป็น Matrix ที่กำหนดที่ตั้งจุดมองของกล้องโดยทำการกำหนดจากคำสั่ง D3DXMatrixLookAtLH(_ MOut As D3DMATRIX, VEye As D3DVECTOR, VAt As D3DVECTOR, _ VUp As D3DVECTOR) ซึ่ง Veyes หมายถึงตำแหน่งของกล้อง Vat คือ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตำแหน่งที่เรามองวัตถุ ส่วน Vup ให้ Set ค่าเป็น (0,1,0) เสมอ และ Mout คือ Matrix ลัพธ์ที่จะนำไปกำหนดค่ามุมมองจาก คำสั่ง g_D3DDevice.SetTransform D3DTS_VIEW, Mout

จากนั้นทำการกำหนดค่า Projection Matrix ซึ่งเป็นการกำหนดค่า Focus และ ช่วงระยะ Z min และ Z max ซึ่งมีวิธีในการกำหนดค่าอยู่ 2 แบบหลักๆ คือกำหนดจากค่า FOV (field of-view)และกำหนดค่า Width และ Height ดังนี้

D3DXMatrixPerspectiveLH(Mout As D3DMATRIX, W As Single,h As Single, zn As Single, zf As Single) เป็นคำสั่งกำหนดค่า Projection Matrix จากค่า ความกว้างและความยาว

D3DXMatrixPerspectiveFovLH(Mout As D3DMATRIX, Fovy As Single, aspect As Single, Zn As Single, Zf As Single) เป็นการกำหนดค่า Projection Matrix จาก ค่า FOV และกำหนดค่า ความยาวและความกว้างจาก ค่า Aspect ratio ซึ่งก็คือ อัตราความกว้างต่อความยาวของจอภาพ

แล้ว Set ค่า Mout ให้เป็น Projection Matrix จาก

g_D3DDevice.SetTransform D3DTS_PROJECTION, Mout

3.2.4 การกำหนดค่า Lighting และ Material

สามารถกำหนดค่า Lighting ได้จากคำสั่งDirect3DDevice8.SetLight index, Light โดย Index หมายถึงค่าลำดับของ Light และ Light หมาย ถึง ตัวแปรประเภท D3DLIGHT8 ซึ่งมีค่า Parameter ต่างๆที่ต้องกำหนดตามชนิดของ แสงดังนี้

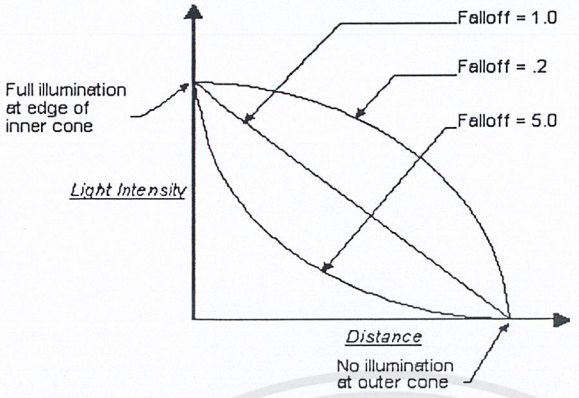
Type D3DLIGHT8

Ambient As D3DCOLORVALUE	ระบุถึงสีAmbient ของแสง
diffuse As D3DCOLORVALUE	ระบุถึงสี Diffuse ของแสง
specular As D3DCOLORVALUE	ระบุถึงสี Specular ของแสง
type As CONST_D3DLIGHTTYPE	ระบุถึงชนิดของแสงมี Point,Spot,Direction
Attenuation0 As Single	ค่า Attenuation จะบอกถึงอัตราการเปลี่ยนแปลงความเข้มของแสงต่อระยะทางห่างจากแหล่งกำเนิด
Attenuation1 As Single	
Attenuation2 As Single	
Direction As D3DVECTOR	ค่าทิศทางของแสงประเภท Direction และ Spot
Falloff As Single	อัตราการลดลงของแสงจาก ภายในไปยังภายนอกของแสงประเภท Spot ปกติจะเป็น 1
Phi As Single	มุมของ Cone ภายนอกของ Spot light
position As D3DVECTOR	ตำแหน่งของแสง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Range As Single ช่วงการส่องสว่างของแสง

Theta As Single มุมของ Cone ภายนอกของ Spot light



รูปที่ 3.7 กราฟแสดงความสัมพันธ์ของค่า Fall off กับค่า Intensity

ส่วน Material นั้นสามารถกำหนดได้จากคำสั่ง

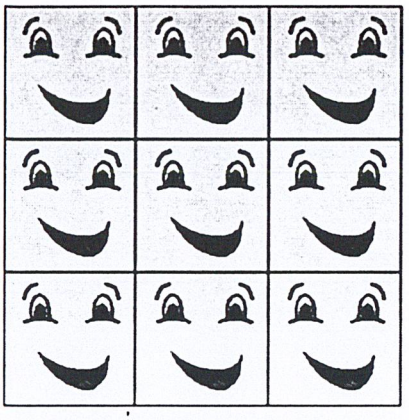
g_D3DDevice.SetMaterial mtrl โดย mtrl นั้นหมายถึงตัวแปลประเภท D3DMATERIAL ซึ่งสามารถกำหนดคุณสมบัติเป็นค่า Diffuse , Specular ,Emission ,Ambient , Power

3.2.5 Texture Mapping

การ Map Texture ลงบนภาพจะต้องกำหนด Coordinate Texture ก่อนดังนั้นข้อมูลของ vertex จะมี ค่าพิกัด XYZ และต้องมีพิกัด ของ Texture เพิ่มขึ้นมา โดยมีค่าอยู่ในช่วง 0.0-1.0 ซึ่งเมื่อ Map Texture ลงไปแล้ว 1 ภาพจะปิดพอดีกับวัตถุ แต่ถ้า Texture Coordinate มีค่ามากกว่า 1.0 เช่น 3.0 จะทำให้มีการ Map ภาพลงไปมากกว่า 1 ภาพ ขึ้นอยู่กับการกำหนดค่า Address ของ Texture ดังนี้

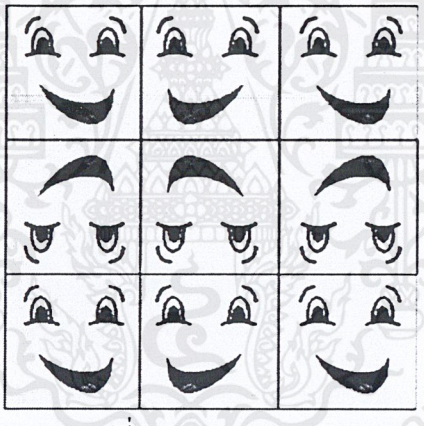
1. Wrap Texture Address Mode ถ้ามีการ Set ค่า Texture Coordinate ที่มากกว่า 1.0 เมื่อทำการ Map ภาพจนถึง พิกัดของ Texture ที่ 1.0 แล้วจะเริ่มทำการ Map รูปลงไปซ้ำอีกครั้งหนึ่ง เช่นมีการ Set ค่าของ Coordinateอยู่ระหว่าง 0.0 – 3.0 จะได้ผลดังรูป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.8 Wrap mode

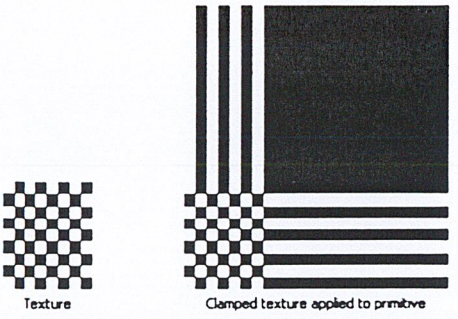
2. Mirror Texture Address Mode ก็คล้ายกรณี ของWrap แต่จะทำการกลับข้าง เช่น ถ้า Tu(ค่า Cordinate ของ Texture ในแกน X) มีค่าเกิน 1.0 เมื่อใดก็จะทำการกลับข้างจากซ้ายไปขวา หรือ ขวาไปซ้าย แต่ถ้าTv (ค่า Cordinate Texture ในแกน Y) มีค่าเกินหนึ่งเมื่อใดในการ Map ภาพครั้งต่อไปจะกลับหัวกลับหาง คล้ายกับว่าเป็นภาพที่เกิดจากกระจกสะท้อนไปมาดังรูป



รูปที่ 3.9 Mirror mode

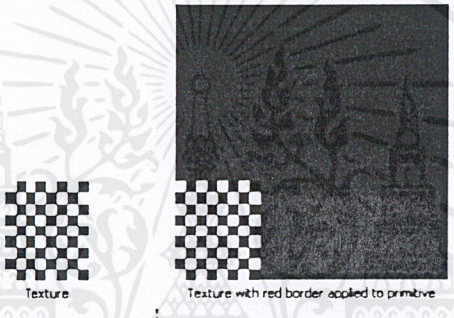
3. Clamp Texture Address mode เช่นเดียวกับ 2 กรณีที่แล้วแต่ Mode นี้เมื่อ Tu หรือ Tv มีค่าถึง1.0แล้วยัง Map ต่อไม่หมดจะทำการ Map ค่าสุดท้ายที่ Map ลงไปดังรูป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.10 Clamp Texture mode

4. Border Texture Address Mode เหมือนกับกรณีที่ผ่านๆมาแต่ การ Map จะ Map ลงถึงแค่ Tu,Tv=1.0 เท่านั้นถ้านอกเหนือบริเวณนี้ จะไม่มีการ Map เกิดขึ้น ดังนั้นจะแสดงออกมาแต่สีของ Material ที่ตั้งไว้เท่านั้นดังรูปตั้งค่า Material เป็นสีแดง



รูปที่ 3.11 Border Texture mode

การ Map Texture ลงบนวัตถุจะต้องกำหนดTexture ที่จะนำมา Map ก่อน โดยใช้คำสั่ง

Set g_Texture = g_D3DX.CreateTextureFromFile(g_D3DDevice, "TextureName.bmp")

โดย g_texture เป็นตัวแปรประเภท Direct3Dtexture จากนั้นทำการ กำหนดค่า Cordinate ของแต่ละ Vertices ที่เราสร้างขึ้นเช่นกรณีวัตถุเป็นรูปทรงกระบอก สามารถกำหนด Tu ,Tv ได้ดังนี้

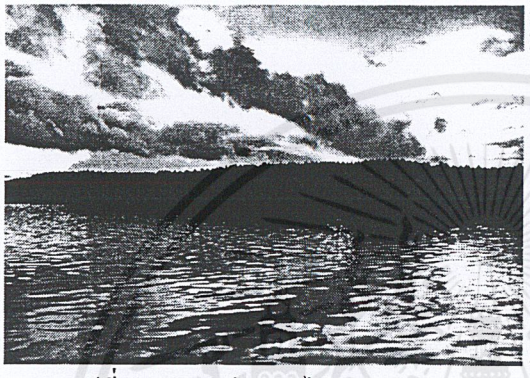
```
For i = 0 To 49
    theta = (2 * g_pi * i) / (50 - 1)
    Vertices(2 * i + 0).postion = vec3(Sin(theta), -1, Cos(theta))
    Vertices(2 * i + 0).color = &HFFFFFFFF  'white
    Vertices(2 * i + 0).tu = i / (50 - 1)
    Vertices(2 * i + 0).tv = 1
    Vertices(2 * i + 1).postion = vec3(Sin(theta), 1, Cos(theta))
    Vertices(2 * i + 1).color = &HFF808080  'grey
    Vertices(2 * i + 1).tu = i / (50 - 1)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

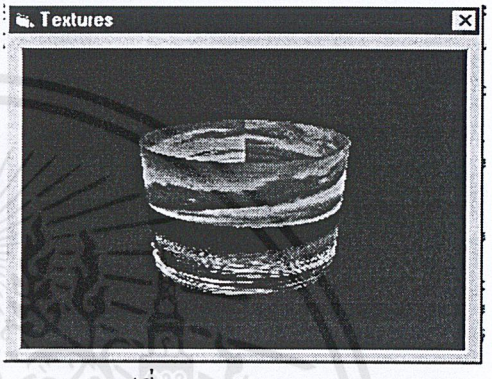
```
Vertices(2 * i + 1).tv = 0

Next
```

g_D3DDevice.SetTexture 0, g_Texture เป็นการกำหนดให้ใช้ Texture ที่อยู่ใน g_texture Map ลงบนวัตถุ จะเห็นว่าได้ตั้งค่า Tu ,Tv มีค่าไม่เกินหนึ่งดังนั้นจึงได้ภาพที่ Map ลงไปภาพเดียวโดยจะยืดออกตามความยาวของวัตถุที่ได้สร้างขึ้นและได้ภาพที่ทำการ Map แล้วดังนี้



รูปที่ 3.12 ภาพก่อนนำไป Map



รูปที่ 3.13 หลังจากการ Map ภาพ

3.2.6 Render

เมื่อสร้างวัตถุและกำหนดคุณสมบัติให้แก่วัตถุแล้วก็ทำการ Render เพื่อแสดง ภาพ โดยมีลำดับขั้นตอนตามนี้

1.ทำการเคลียร์ Viewport และ Z-buffer โดยใช้คำสั่ง

g_D3DDevice.Clear 0, ByVal 0, D3DCLEAR_TARGET Or D3DCLEAR_ZBUFFER, &HFF&, 1#, 0

2. กำหนด การเริ่มต้นสร้างหน้าจอ โดยใช้คำสั่ง

g_D3DDevice.BeginScene

3.ทำการกำหนด Matrix ต่างๆที่ต้องการ(World ,View ,Projection)

4.วาดวัตถุ

5.จบหน้าจอทำการสลับ Back buffer ขึ้นมา โดยใช้คำสั่ง

g_D3DDevice.EndScene

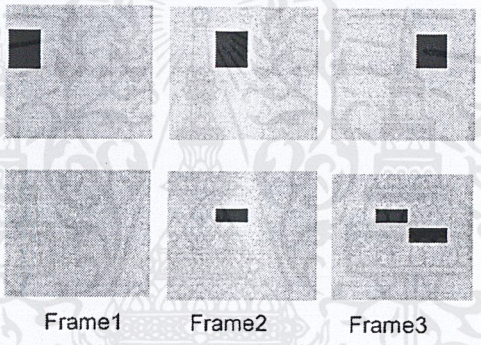
g_D3DDevice.Present ByVal 0, ByVal 0, 0, ByVal 0

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.3 การสร้างภาพเคลื่อนไหว

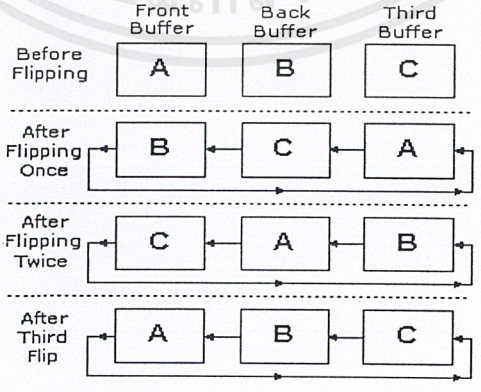
เป็นการกำหนดการเคลื่อนไหวให้กับวัตถุ ซึ่งมีเทคนิคอยู่หลายอย่างในสมัยแรกในการสร้างการ์ตูน ก็จะใช้ การวาดภาพลงไปทีละภาพ หรือใช้การปั้นหุ่นขึ้นมาแล้วถ่ายภาพ จากนั้นก็ทำการเปลี่ยนแปลงท่าทางทีละนิด แล้วก็ถ่ายภาพทีละ Shot หรือเรียกว่า Stop-Motion เนื่องจากการประมวลผลของข้อมูลทางด้านภาพนั้นใช้ทรัพยากรมหาศาล คอมพิวเตอร์ในสมัยนั้นยังไม่มีประสิทธิภาพเพียงพอในการประมวลผล แต่เมื่อ Computerได้พัฒนามามากขึ้นในปัจจุบันจึงได้มีการนำ Computer เข้ามามีส่วนช่วยในการจัดการและสร้างภาพเคลื่อนไหวมากขึ้น

โดยวิธีการสร้างภาพเคลื่อนไหวใน DirectX นั้นจะใช้การ Flipping Page โดยเราจะสร้างพื้นผิวซ่อนเอาไว้เรียกว่า Back Buffer แล้วขั้นตอนการสลับมุมมองจากภาพที่เราสร้างขึ้นที่ Back buffer กับ ภาพที่เราแสดงอยู่บนสกรีนเราเรียกว่า Flipping ขั้นตอนดังกล่าวจะใช้พื้นผิวมากกว่าหนึ่งผืน เพื่อสร้างพื้นผิวเพียงผิวเดียว ขณะกำลังวาดพื้นผิวหน้าจออยู่ด้วย เพื่อป้องกันการฉีกขาดของ Frame ซึ่งจะเกิด Imageเมื่อมีการวาดภาพดังรูป



รูปที่ 3.14 Tearing

โดยสามารถกำหนดจำนวนของ Back Buffer ได้จาก D3DPRESENT_PARAMETER ถ้ามี Back Buffer มากกว่า 1 อันวิธีการ Flipping Page จะเป็นดังรูปซึ่งค่า Default จะเป็น D3DSWAPEFFECT_FLIP



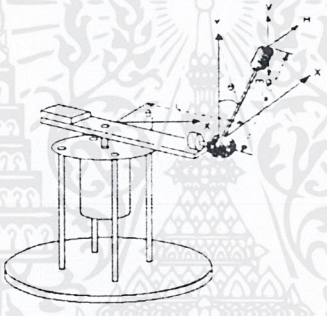
รูปที่ 3.15 การ Flipping Page

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4
สมการคุณลักษณะของ Inverted pendulum

Mathematical Modeling of Physical System

ในการจะทำการควบคุมระบบใดที่ได้สร้างแบบจำลองขึ้นมานั้นจำเป็นที่จะต้องเรียนรู้สมการรูปแบบทางคณิตศาสตร์ของระบบนั้นๆ ด้วย ซึ่งในการหาสมการรูปแบบทางคณิตศาสตร์ของระบบต่างๆ นั้นมีหลายระบบ เช่น ระบบทางไฟฟ้า ระบบที่เกี่ยวกับระดับของของเหลว ระบบความร้อน และ อื่นๆ แต่ระบบที่จะนำเสนอต่อไปนี้ เป็นการหาสมการรูปแบบทางคณิตศาสตร์ของระบบเชิงกล (Mechanical System) ของ การควบคุมระบบอินเวอร์ทเพนดูลัม ที่ได้สร้างไว้ เพื่อจะได้สมการรูปแบบทางคณิตศาสตร์ ของ การควบคุมเพนดูลัม เอาไว้ใช้ทำการควบคุมต่อไปเมื่อ



รูปที่ 4.1 การแตกแรงเพื่อหาสมการรูปแบบทางคณิตศาสตร์

4.1 โมเดลทางคณิตศาสตร์ของ Pendulum

โดยที่ l_1 : ความยาวของแขนขับนับจากจุดหมุน (m)

l_2 : ความยาวของก้านลูกตุ้ม (m)

m : มวลของลูกตุ้ม (kg)

θ_1 : มุมที่เปลี่ยนไปของแขนขับ (rad)

θ_2 : มุมที่เปลี่ยนไปของก้านลูกตุ้ม (rad)

V : แรงในแนวตั้งที่ทำกับก้านลูกตุ้ม (N)

H : แรงในแนวนอนที่ทำกับก้านลูกตุ้ม (N)

จากรูปที่ 4 0สามารถหาฟังก์ชันถ่ายโอนได้ดังนี้

จากกฎข้อที่สองของนิวตัน จะได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

พิจารณาในแกน Y :

$$V - mg = m\frac{d^2}{dt^2}(l_2 \cos\theta_2) \tag{4.1}$$

พิจารณาในแกน X :

$$H = m\frac{d^2}{dt^2}(l_2 \sin\theta_2 + \theta_1 l_1) \tag{4.2}$$

เมื่อ $\theta_1 l_1$ คือระยะทางที่แขนเคลื่อนที่ไปเป็นวงกลมด้วยมุม θ_1

จากหนังสือ" Modern Control Engineering" หน้า 103 เราจะได้ความสัมพันธ์ คือ

$$J_2\ddot{\theta}_2 + b_2\dot{\theta}_2 = Hl_2 \cos\theta_2 - Vl_2 \sin\theta_2 \tag{4.3}$$

โดยที่ J_2 เป็นค่า Moment of Initial ของก้านลูกตุ้ม

b_2 เป็นค่าความฝืด (Viscous Friction)

นำสมการที่ (4.1),(4.2) แทนลงใน (4.3)

$$J_2\ddot{\theta}_2 + b_2\dot{\theta}_2 = \left[m\frac{d^2}{dt^2}(l_2\sin\theta_2 + l_1\theta_1) \right] l_2\cos\theta_2 - \left[mg + m\frac{d^2}{dt^2}(l_2\cos\theta_2) \right] l_2\sin\theta_2 \tag{4.4}$$

และ

$$\frac{d^2}{dt^2}\sin\theta = -(\sin\theta)\dot{\theta}^2 + (\cos\theta)\ddot{\theta} \tag{4.5}$$

$$\frac{d^2}{dt^2}\cos\theta = -(\cos\theta)\dot{\theta}^2 - (\sin\theta)\ddot{\theta} \tag{4.6}$$

จะได้

$$(J_2 - ml_2^2)\ddot{\theta}_2 + b_2\ddot{\theta}_2 + mgl_2 \sin\theta_2 = \ddot{\theta}_1 ml_1 l_2 \cos\theta_2 \tag{4.7}$$

เราประมาณค่า $\sin\theta_2 \approx 0$ และ $\cos\theta_2 \approx 1$ เมื่อให้ค่า θ_2 เข้าใกล้ 0 จะได้

$$(J_2 - ml_2^2)\ddot{\theta}_2 + b_2\dot{\theta}_2 = \ddot{\theta}_1 ml_1 l_2 \tag{4.8}$$

Laplace Transform จะได้ฟังก์ชันการถ่ายโอน ของความสัมพันธ์ระหว่าง θ_1 และ θ_2 เป็น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

$$\frac{\theta_2(s)}{\ddot{\theta}_1(s)} = \frac{ml_1l_2}{(J_2 - ml_2^2)s^2 + b_2 s} \tag{4.9}$$

4.2 โมเดลคณิตศาสตร์ของดีซีมอเตอร์

ดีซีมอเตอร์ที่ใช้ร่วมกับดีซีแอมปลิไฟทั้งในระบบการบังคับตำแหน่งและการบังคับความเร็วมักจะได้รับการประยุกต์ใช้เป็นส่วนประกอบสร้างกำลังานในระบบการนำร่อง และระบบการบังคับต่างๆ และเนื่องจากวิทยาการเกี่ยวกับสารแม่เหล็ก และการขยายด้วยโซลิดสเตท ทำให้ดีซีมอเตอร์แบบแม่เหล็กถาวรได้รับความนิยมใช้เป็นส่วนประกอบการขับเคลื่อนในระบบการบังคับแบบปิดลูปต่างๆ มากขึ้น การออกแบบและการชดเชยระบบดังกล่าวได้อย่างเหมาะสม จะต้องใช้โมเดลคณิตศาสตร์ของส่วนประกอบทั้งหมดในระบบ ในหัวข้อนี้เราจะได้พัฒนาลิเนียร์โมเดลของดีซีมอเตอร์แบบแม่เหล็กถาวร และแบบฟิลด์แยกกระตุ้น



4.2.1 โมเดลอิเล็คโทรแมคคานิคอล

ส่วนสำคัญของดีซีมอเตอร์แบบฟิลด์แยกกระตุ้นมีโมเดลดังแสดงในรูปที่ 2.2

โดยที่ R_a : ความต้านทานของอาร์มาเจอร์

L_a : อินดัคแต้นซ์ของอาร์มาเจอร์

V_g: โวลท์เต็จกำเนิดในอาร์มาเจอร์ (โวลท์เต็จย้อนกลับ)

R_f : ความต้านทานของฟิลด์

L_f : อินดัคแต้นซ์ของฟิลด์

ϕ : ช่องว่างอากาศของเส้นแรงสนามแม่เหล็ก

ω : ความเร็วของเพลาอาร์มาเจอร์

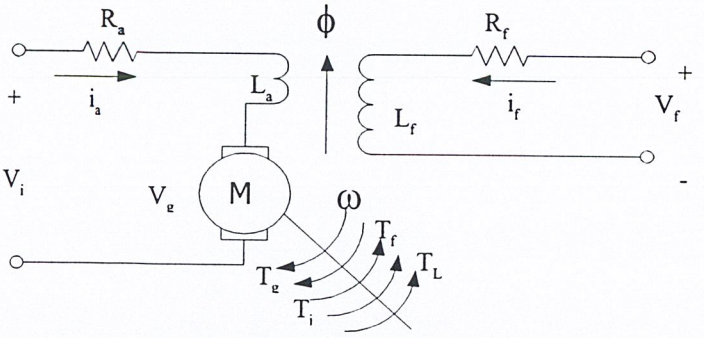
T_g : แรงบิดที่พัฒนาขึ้นในมอเตอร์

T_f : แรงบิดเสียดทานของมอเตอร์

T_j : แรงเฉื่อยของมอเตอร์

T_L : แรงบิดโหลดบนเพลาของมอเตอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.2 โมเดลของดีซีมอเตอร์แบบฟิลด์แยกกระตุ้น

ขั้นแรกเราจะหาสมการพื้นฐานโมเดลของดีซีมอเตอร์ได้จากลูฟของอาร์มาเจอร์

$$V_i(t) = R_a i_a(t) + L_a \frac{di_a(t)}{dt} + V_g(t) \tag{4.10}$$

เทอมโวลท์เต็จ $V_g(t)$ ในสมการ (4.10) คือ โวลท์เต็จย้อนกลับของมอเตอร์ ซึ่งเกิดขึ้นเมื่อเส้นลวดตัวนำของอาร์มาเจอร์หมุนตัดเส้นแรงแม่เหล็กซึ่งเกิดขึ้นโดยกระแสของฟิลด์ (I_f) ตามกฎของฟาราเดย์ลูฟของเส้นลวดตัวนำหมุนในฟิลด์แม่เหล็กคงที่จะมีการเหนี่ยวนำโวลท์เต็จขึ้นในขดลวดนั้น

$$V(t) = \frac{d\lambda(t)}{dt} \tag{4.11}$$

เมื่อ λ(t) เส้นแรงแม่เหล็กที่รั่วไปยังขดลวดและ t คือ เวลาในการหมุนของคอมมิวเตเตอร์ของมอเตอร์ การควบคุมวงจรของแต่ละส่วนของตัวนำในโรเตอร์จะเกิดโวลท์เต็จขึ้นในส่วนของ ตัวนำนั้นตามสมการ (2.11) เมื่อ dλ(t)/dt จะเป็นสัดส่วนต่อเส้นแรงแม่เหล็กในช่องว่างอากาศและความเร็วเชิงมุม ω(t) หรือเราจะได้ว่า

$$V_g(t) = K\phi(t)\omega(t) \tag{4.12}$$

สมมติให้กระแสของฟิลด์มีค่าคงที่ และไม่คิดถึงส่วนการเปลี่ยนแปลงในเส้นแรงฟิลด์เนื่องจากอาร์มาเจอร์รีแอคชั่นเส้นแรงฟิลด์ ก็จะมีค่าคงที่ ดังนั้น สมการ (4.12) ก็จะกลายเป็น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

$$V_g(t) = K_e\omega(t) \tag{4.13}$$

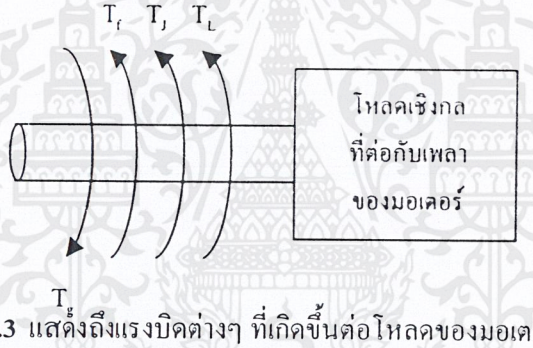
เมื่อเราสมมติให้เส้นแรงของฟิลด์มีค่าคงที่ แรงบิดของแม่เหล็กไฟฟ้าซึ่งเกิดขึ้นแก่โรเตอร์ของมอเตอร์ จะเป็นสัดส่วนกับกระแสอาร์มมาเจอร์

$$T_g(t) = K_t i_a(t) \tag{4.14}$$

โดยที่ K_t คือค่าคงที่ของแรงบิดของมอเตอร์กำลังงานเชิงกลที่เกิดขึ้นในโรเตอร์ คือผลคูณของแรงบิดที่เกิดขึ้นและความเร็วเชิงมุม

$$P_g(t) = T_g(t)\omega(t) \tag{4.15}$$

กำลังงานเชิงกลที่เกิดขึ้นในโรเตอร์ทั้งหมดนี้จะจ่ายไปยังโหลดที่ต่ออยู่กับเพลาของมอเตอร์แต่กำลังงานนี้บางส่วนจะสูญเสียไปในมอเตอร์ การสูญเสียเนื่องมาจากแรงเสียดทานหมายถึงหน่วงเนื่องจากลมที่มีต่อโรเตอร์ แรงเสียดทานตัวรองรับโรเตอร์ กระแสที่ไหลวนในเหล็กของโรเตอร์และฮีสเตอร์รีซิส จากรูปที่ 42 แสดงให้เห็นถึงแรงบิดต่างๆ ที่เกิดขึ้นต่อโหลดของมอเตอร์ คือ



รูปที่ 4.3 แสดงถึงแรงบิดต่างๆ ที่เกิดขึ้นต่อโหลดของมอเตอร์

โดยที่ $T_g(t)$: แรงบิดของมอเตอร์

$T_f(t)$: แรงบิดที่ต้องชนะการสูญเสียเนื่องจากการเสียดทาน

$T_j(t)$: แรงบิดเพื่อใช้เพิ่มอัตราเร่งแก่ ความเฉื่อยของโหลด

$T_L(t)$: แรงบิดโหลด

ในช่วงเวลาใดๆ ก็ตาม แรงบิดของมอเตอร์จะต้องเท่ากับ และมีทิศทางตรงข้ามกับผลรวมของแรงบิด $T_f(t)$, $T_j(t)$ และ $T_L(t)$ ดังนั้น

$$T_g(t) = T_f(t) + T_L(t) + J\frac{d\omega(t)}{dt} \tag{4.16}$$

โดยที่ J คือผลรวมของโมเมนท์แรงเฉื่อยของโรเตอร์และโหลดที่ต่ออยู่ที่เพลามอเตอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ผลรวมของแรงบิดเสียดทานที่ประกอบกันขึ้นที่เพลาของมอเตอร์ ซึ่งเป็นลิเนียร์ฟังก์ชันกับความเร็วเชิงมุมของโรเตอร์ เรียกว่า ส่วนประกอบของวิสคอสฟริคชั่น และมักจะอยู่ในเทอมที่แยกออกจากฟริคชั่นอื่นๆ ซึ่งแสดงได้ด้วยสมการ ต่อไปนี้

$$T_g(t) = T_f(t) + T_L(t) + J\frac{d\omega(t)}{dt} + B\omega(t) \tag{4.17}$$

เมื่อ B คือสัมประสิทธิ์ของวิสคอสฟริคชั่นของมอเตอร์และโหลดที่ต่ออยู่กับเพลาของมอเตอร์ $T_f(t)$ คือผลรวมของฟริคชั่นของโหลด และของมอเตอร์ทั้งหมด มีแรงต้านของลมและการสูญเสียกำลังในเหล็กของเพลามอเตอร์ ยกเว้นวิสคอสฟริคชั่น

สมการ (4.10) (4.13) (4.14) และ (4.17) เป็นชุดสมการพื้นฐานของดีซีมอเตอร์โมเดล และจากสมการเหล่านี้ เราสามารถจะหาทรานสเฟอร์ฟังก์ชันของดีซีมอเตอร์ได้ โดยใส่ลาปลาสทรานส์ ฟอร์ม ทั้งสองข้างของชุดสมการพื้นฐาน และเขียนได้ใหม่เป็น

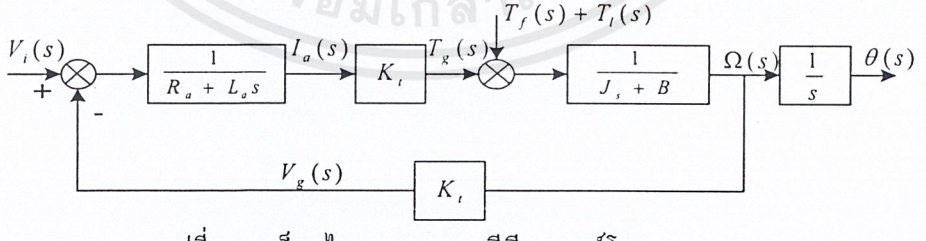
$$V_i(s) - V_g(s) = (R_a + sL_a)I_a(s) \tag{4.18}$$

$$V_g(s) = Ke\,\omega \tag{4.19}$$

$$T_g(s) = K_t\,I_a(s) \tag{4.20}$$

$$T_g(s)\;T_f(s)\;T_L(s) = (B + sJ)\,\omega \tag{4.21}$$

บล็อคไดอะแกรมที่แสดงถึงสมการพื้นฐานเหล่านี้ แสดงได้ดังในรูปที่ 2.4



รูปที่ 4.4 บล็อคไดอะแกรมของดีซีมอเตอร์โมเดล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ข้อควรสังเกต

สมมติว่าโวลท์เต็จที่ป้อนให้กับวงจรอาร์มาเจอร์ของมอเตอร์มีค่าคงที่ดังนั้นมอเตอร์จะหมุนด้วยความเร็วคงที่คือทำงานอยู่ที่สภาวะสงบนิ่งด้วยโหลดที่คงที่กำลังเชิงกลที่เกิดขึ้นโดยโรเตอร์จะหาได้จากสมการ (2.15) และเมื่อร่วมกับสมการ (4.14) ก็จะได้

$$P_g = T_g\omega = K_t\ I_a\omega \tag{4.22}$$

เมื่อทุกเทอมในสมการสุดท้ายมีค่าคงที่ เนื่องจากมอเตอร์ทำงานอยู่ที่สภาวะสงบนิ่ง กำลังไฟฟ้าที่ถูกดูดกลืนโดยอาร์มาเจอร์ต้องเท่ากับผลคูณของโวลท์เต็จคร่อมอาร์มาเจอร์ และกระแสที่ไหลผ่าน ดังนั้น

$$P = V_g\ I_a = K_e\ \omega I_a \tag{4.23}$$

ดังนั้นเราจะได้ว่า กำลังงานเชิงกลที่เกิดขึ้นต้องเท่ากับกำลังงานไฟฟ้าที่ถูกดูดกลืนในโรเตอร์ คือสรุปได้ว่า $K_t = K_e$

4.3 ทรานสเฟอร์ฟังก์ชันของดีซีมอเตอร์

บล็อคไดอะแกรมของรูปที่ 44 แสดงถึงระบบที่มีสองอินพุทและมีเอาท์พุทเป็นทั้งความเร็วเชิงหมุน ω และการเคลื่อนที่แบบเชิงมุม θ จากรูปที่ 2.4 ความเร็วเอาท์พุทของระบบเขียนได้เป็น

$$\omega(s) = G_1(s)V_1(s) + G_2(s)\left[T_f(s) + T_L(s)\right] \tag{4.24}$$

เมื่อ

$$G_1(s) = \frac{\omega(s)}{V_1(s)} \qquad ;\ Tf(s)\ +\ TL(s)\ =\ 0 \tag{4.25}$$

$$G_1(s) = \frac{\omega(s)}{T_f(s) + T_L(s)} \qquad ;\ V1(s)\ =\ 0 \tag{4.26}$$

โดยที่ $G_1(s)$ คือทรานสเฟอร์ฟังก์ชันระหว่างโวลท์เต็จและความเร็ว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

$$G_1(s) = \frac{\omega(s)}{V_1(s)} = \frac{K_t}{(L_a s + R_a)(Js + B) + K_t K_c}$$

$$= \frac{K_m}{\alpha s^2 + \beta s + 1} \tag{4.27}$$

โดยที่

$$K_m = \frac{K_t}{R_a B + K_t K_c} \tag{4.28}$$

$$\alpha = \frac{L_a J}{R_a B + K_t K_c} \tag{4.29}$$

$$\beta = \frac{R_a J + L_a B}{R_a B + K_t K_c} \tag{4.30}$$



รูปที่ 4.5 ตัวอย่างคอนโทรลมอเตอร์และเยนเนอร์เรเตอร์

สมการ (4.27) เป็นโวลท์เต็จทรานสเฟอร์ฟังก์ชันของดีซีมอเตอร์ในเมื่อสมมติว่า T_f และ T_L มีค่าเป็นศูนย์ สมการ (4.27) สามารถเขียนใหม่ได้เป็น

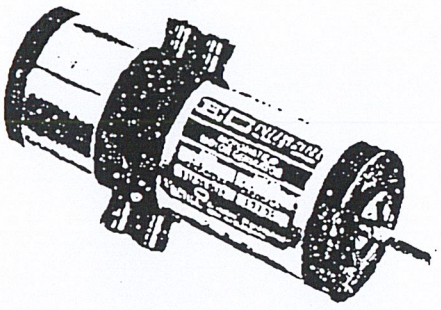
$$G_1(s) = \frac{K_t}{R_a B(1 + \tau_e s)(1 + \tau_m s) + K_t K_c} \tag{4.31}$$

โดยที่

τ_e = L_a/R_a = ไทม์คอนสแต้นท์ทางไฟฟ้า

τ_m = J/B = ไทม์คอนสแต้นท์ทางเชิงกล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.6 ดีซีมอเตอร์ และทาโคมิเตอร์สำหรับใช้ในระบบบังคับมีการป้อนกลับ

ถ้าอินดัคแต้นท์ของอาร์มาเจอร์มีค่าน้อย ไทม์คอนสแต้นท์ทางไฟฟ้าสามารถตัดทิ้งได้และสมการ (4.27) ได้เป็น

$$G_v(s) = \frac{\omega(s)}{V_1(s)} = \frac{K_t}{R_2(Js+B) + K_t K_c} \tag{4.32}$$

$$= \frac{K_m}{\tau s + 1} \tag{4.33}$$

โดยที่ $\tau = \dfrac{R_a J}{R_a B + K_t K_c}$

ในสมการ (4.33) ค่าคงที่ K_m อาจเรียกได้ว่าเป็นค่าคงที่ของมอเตอร์ ทรานสเฟอร์ฟังก์ชันแรงบิดโหลด $G_2(s)$ หาได้เป็น

$$G_2(s) = \frac{\omega(s)}{T_f(s) + T_L(s)} = \frac{-\dfrac{1}{J_s + B}}{1 + \dfrac{K_t K_c}{(Js+B)(L_a s + R_a)}}$$

$$G_2(s) = \frac{-\dfrac{R_a}{R_t} Km \left[\dfrac{L_a}{R_a} s + 1 \right]}{\alpha s^2 + \beta s + 1} \tag{4.34}$$

ซึ่งถ้าอินดัคแต้นท์ของอาร์มาเจอร์ไม่นำมาคิดถึง สมการ (4.34) ก็ลดลงได้

$$G_L(s) = \frac{\omega(s)}{T_f(s) + T_L(s)} = \frac{-\dfrac{R_a}{K_t} K_m}{\tau s + 1} \tag{4.35}$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.4 โมเดลทางคณิตศาสตร์ของPID คอนโทรลเลอร์

มีสมการคือ

$$M_v = K_p Error + K_i \int errordt + K_d \frac{Derror}{Dt}$$

4.5 Simulation Inverted Pendulum

จากสมการทางคุณลักษณะของ Moter , Pendulum , PID สามารถนำมาเขียนโปรแกรมเพื่อจำลองกระบวนการ ในการควบคุม Invert Pendulum ได้โดยเปลี่ยนแปลงสมการคุณลักษณะโดยการแทนที่ การดิฟเฟอร์เรนเชียล ด้วยสมการข้างล่าง

$$\frac{Dv}{Dt} = \frac{v(n) - v(n-1)}{T} \tag{4.36}$$

ซึ่งจะได้สมการคุณลักษณะใหม่สามารถนำมาเขียนโปรแกรมได้ดังนี้ จากสมการคุณลักษณะของ Moter

$$\frac{\omega}{V_i} = \frac{K_m}{\alpha s^2 + \beta s + 1}$$

ย้ายข้าง

$$\alpha\omega s^2 + \beta\omega s + \omega = K_m V_i \tag{4.37}$$

จาก $\frac{Dv}{Dt} = Vs - v(0)$ และ $\frac{D^2 v}{D^2 t} = Vs^2 - sv(0) - v'(0)$

และกำหนดให้ $v(0) = 0$ และ $v'(0) = 0$

จะได้ว่า $\frac{D^2 v}{D^2 t} = Vs^2$ และ $\frac{Dv}{Dt} = Vs$ (4.38)

เมื่อแทนค่าในสมการที่ 4.38 จะได้สมการใหม่คือ

$$\alpha \frac{D^2\omega}{D^2 t} + \beta \frac{D\omega}{Dt} + \omega = K_m V$$

จากนั้นแทนค่าจากสมการที่ 4.36 จะได้สมการใหม่คือ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

$$\alpha\left(\frac{\omega(n)-2\omega(n-1)+\omega(n-2)}{T^2}\right)+\beta\left(\frac{\omega(n)-\omega(n-1)}{T}\right)+\omega(n)=K_m V$$

และทำการย้ายข้างจะได้ผลลัพธ์คือ

$$\omega(n)=\frac{T^2 K_m V(n)+\omega(n-1)(T\beta+2\alpha)-\alpha\omega(n-2)}{T^2+T\beta+\alpha} \quad (4.39)$$

ซึ่งจะใช้สมการนี้แทนมอเตอร์

จากสมการคุณลักษณะของ Pendulum

$$J_m\ddot{\theta}_2-ml_2^2\ddot{\theta}_2+b_2\dot{\theta}_2+mgl_2\sin\theta_2=ml_1l_2\ddot{\theta}_1$$

ให้ $\omega=\dot{\theta}_2$ (4.40)

กำหนดให้ $K_M=(J_m-ml_2^2)$

$K_c=ml_1l_2$

$K_I=mgl_2$

$K_M\dot{\omega}+b_2\omega+K_I\sin\theta=K_c\ddot{\theta}_1$

เมื่อทำการย้ายข้างจะได้ผลลัพธ์คือ

$$\omega=\frac{(-b_2\omega-K_I\sin\theta_2+K_c\ddot{\theta}_1)}{K_M} \quad (4.41)$$

จากการประมาณค่าของรุงเง-กุตต้าในสมการที่ 4.40

จะได้ K_1,K_2,K_3,K_4ของสมการที่ 4.40 คือ

$K_1=\omega_n$

$K_2=\omega_n+0.5K_1$

$K_3=\omega_n+0.5K_2$

$K_4=\omega_n+0.5K_3$

จะได้

$$\theta_{n+1}=\theta_n+\frac{1}{6}(K_1+2K_2+2K_3+K_4) \quad (4.42)$$

และจากการประมาณค่าของรุงเง-กุตต้าในสมการที่ 4.41

จะได้ K_1, K_2, K_3, K_4 ของสมการที่ 4.41 คือ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

$$K_1 = \frac{(-b_2\omega_n - K_I \sin\theta_2 + K_c \ddot{\theta}_1)}{K_M}$$

$$K_2 = \frac{(-b_2(\omega_n + 0.5K_1) - K_I \sin\theta_2 + K_c \ddot{\theta}_1)}{K_M}$$

$$K_3 = \frac{(-b_2(\omega_n + 0.5K_2) - K_I \sin\theta_2 + K_c \ddot{\theta}_1)}{K_M}$$

$$K_4 = \frac{(-b_2(\omega_n + K_3) - K_I \sin\theta_2 + K_c \ddot{\theta}_1)}{K_M}$$

จะได้

$$\omega_{n+1} = \omega_n + \frac{1}{6}(K_1 + 0.5K_2 + 0.5K_3 + K_4) \qquad (4.43)$$

จากสมการ PID

$$M_v = K_p Error + K_i \int errordt + K_d \frac{Derror}{Dt}$$

Diff ทั้งสมการ 1 ครั้ง

$$\frac{DM_v}{Dt} = K_p \frac{Derror}{Dt} + K_i error + K_d \frac{D^2 error}{D^2 t}$$

แทนค่าจากสมการ 4.36 และ 4.38 จะได้

$$\frac{M_v(n) - M_v(n-1)}{T} = error(n)\left(\frac{K_p + TK_i}{T}\right) - \frac{K_p error(n-1)}{T} + \frac{K_d(error(n) - 2error(n-1) + error(n-2}{T^2}$$

เมื่อทำการย้ายข้างแล้วจะได้

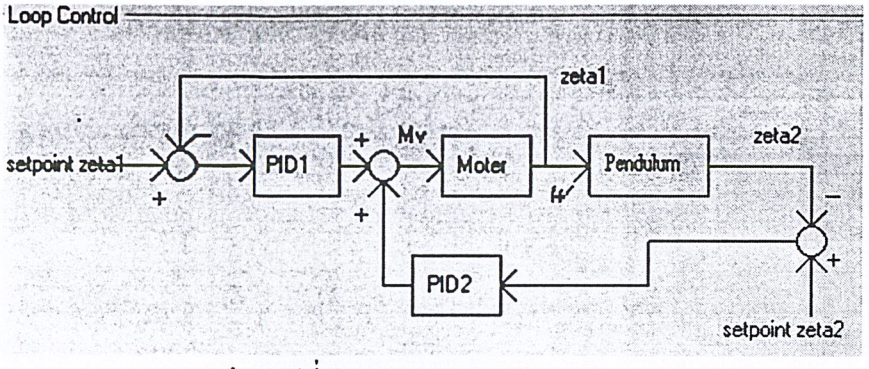
$$M_v(n) = error(n)\left(K_p + Tk_i + \frac{K_d}{T}\right) + M_v(n-1) - error(n-1)\left(K_p + \frac{2K_d}{T}\right) + \frac{K_d error(n-2)}{T}$$

(4.44)

จะได้สมการที่ใช้ในการ Simulation คือ สมการที่ 3.39 ,3.42,4.43 และ 3.44

โดย Block Diagram ในการควบคุมดังนี้แล้วนำค่า Zeta1 ,Zeta2 ไปแสดงผลบนโปรแกรม โดยสร้าง Object ขึ้นมารับค่าไปแสดงการเคลื่อนไหวบนคอมพิวเตอร์ดังรูป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

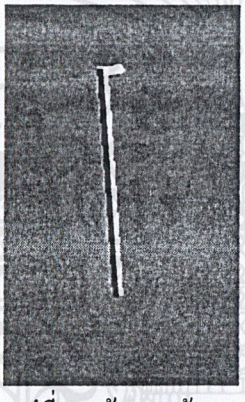


รูปที่ 4.7 Block Diagram

4.6 โมเดลภาพ 3 มิติ

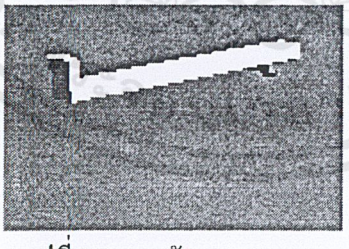
แบ่งได้ 3ส่วนคือ

1.ก้านลูกตุ้ม



รูปที่ 4.8 ก้านลูกตุ้ม

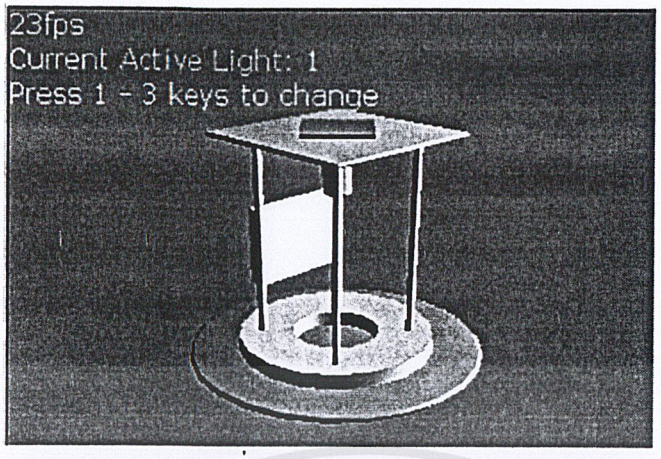
2.แขนขับ



รูปที่ 4.9 แขนขับ

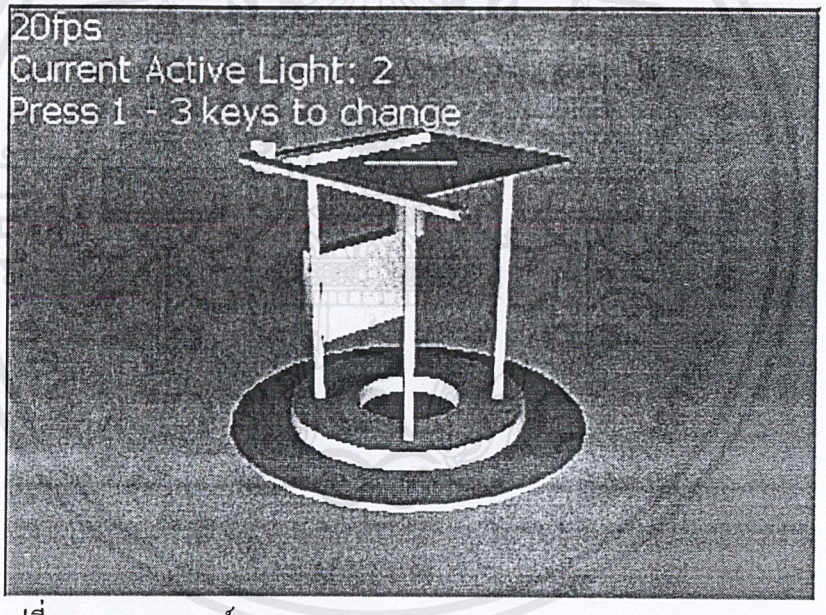
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.ส่วนฐาน



รูปที่ 4.10 ส่วนฐาน

โดยกำหนดให้ส่วนฐานอยู่กับที่ ส่วนแขนขับหมุนตามค่า Zeta1 ส่วนก้านลูกตุ้ม นั้นเคลื่อนไหวพร้อมกัน 2 แกนโดยใช้ Translation Matrix ในการ สร้างภาพเคลื่อนไหว



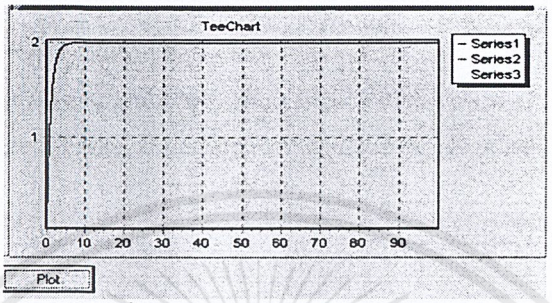
รูปที่ 4.11 ภาพสมบูรณ์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 5

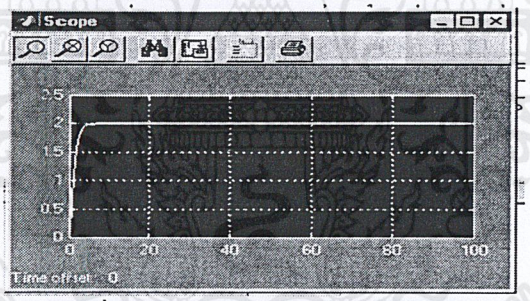
ผลการทดลอง

สามารถสร้างภาพ Animation 3D ได้แบบ Real Time และสามารถทำการ Simulate การควบคุมและการเคลื่อนไหวของ Motor ได้ ดังรูป



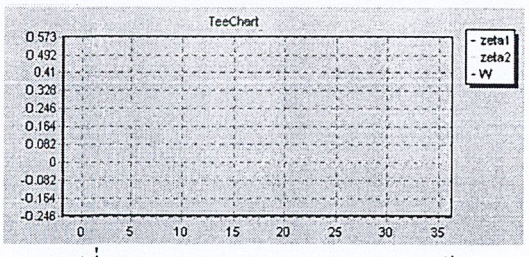
รูปที่ 5.1 กราฟแสดงค่ามุมของ Motor

จากค่า Parameter คือ ค่า P=1,I=0,D=1ค่า R คือ 0.5โอห์ม, L คือ 0 เฮนรี่, Kt คือ 0.15 นิวตันเมตร/แอมป์, Kc คือ0.01 โวลต์-วินาที/เรเดียน , b คือ 0.000006 Kg-m^2/sec และJmotor คือ 0.0015 Kg-m^2 ตามลำดับ เมื่อเปรียบเทียบผลจาก MathLab



รูปที่ 5.2 กราฟแสดงค่ามุมของ Motor

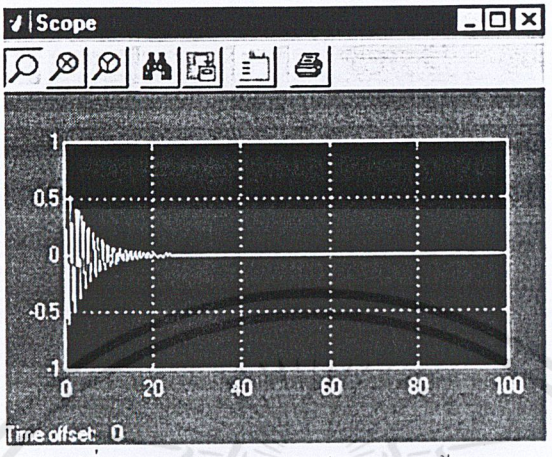
ที่ Transfer function ของ Motor เท่ากับ $\frac{100}{0.5s+1}$ และสามารถ Simulate การเคลื่อนที่ของตัว Pendulum ได้ดังรูป



รูปที่ 5.3 กราฟแสดงการแก่วงของลูกตุ้ม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ซึ่ง Parameter ของ Pendulum มีค่าดังนี้ มวลของลูกตุ้มและก้านมีค่า 0.2 Kg โมเมนต์ของก้านมีค่า 0.05 Kg-m^2 ค่าความหน่วงมีค่า 0.01 Kg-m^2/sec ความยาวของแขนขับและก้านลูกตุ้มมีค่าเท่ากับ 0.2 และ 0.35 m เมื่อเทียบกับกราฟที่เป็นผลเฉลยของสมการ



รูปที่ 5.4 กราฟแสดงการแกว่งของลูกตุ้ม

จะเห็นได้ว่าผลที่ได้ต่างกันเนื่องมาจาก การทำ linearization ในขั้นตอนการหาผลเฉลย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 6
สรุปผลแนวทางการพัฒนาต่อและปัญหาที่พบ

เนื่องจากยังขาดสมการที่ใช้อธิบายการเคลื่อนที่ของ Pendulum ที่สมบูรณ์ การเคลื่อนไหวจึงยังไม่เหมือนจริงทั้งหมด ทำให้การคอนโทรลนั้นยังไม่เหมือนจริงเท่าที่ควร และข้อมูลของ Direct3D นั้นหาได้ค่อนข้างยาก เนื่องจากมีการ Update version ใหม่อยู่เสมอ และประกอบกับขาดความรู้ทางด้านภาพสามมิติของผู้จัดทำ ทำให้ใช้เวลาเกือบทั้งหมดไปกับการศึกษาความรู้ทางด้านภาพ 3 มิติ และเป็นผลทำให้เวลาในการทำโปรแกรมจำลองภาพลดลงไปด้วย

Project simulation นี้สามารถใช้เป็นแนวทางการเขียน โปรแกรมเพื่อจำลองระบบต่างๆด้วยภาพ 3 มิติได้เป็นอย่างดี ขาดเพียงแต่การสร้างเงา หมอกเงาภาพสะท้อนของเงาและ Particle ต่างๆ และนำไปจำลองระบบก่อนที่จะออกแบบระบบ ที่มีความซับซ้อนได้หากทราบสมการคุณลักษณะของระบบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรมจำลองอินเวอร์ทเพนดูลัม 3 มิติ

```
'Constant
Public Kmoter As Single
Public alfa As Single
Public beta As Single
Public Kmoment As Single
Public Kdevide As Single
Public KML1L2 As Single
Public BTS As Single
Public KMGL As Single
Dim condition As Boolean
'*****************************************
'PID1&PID2
Dim speed As Single
Dim P1 As Single
Dim Ts As Single
Dim I1 As Single
Dim D1 As Single
Dim P2 As Single
Dim I2 As Single
Dim D2 As Single
Dim setpointzeta1 As Single
'-----------------------------------------------------
Dim zeta1 As Single
Dim zeta1o As Single
Dim zeta2 As Single
Dim zeta2o As Single
Dim zeta2oo As Single
Dim W1 As Single
Dim W1o As Single
Dim W1oo As Single
Dim k1 As Single
Dim k2 As Single
Dim k3 As Single
Dim k4 As Single
Dim Rotatealfa As Single
Dim rotatebeta As Single
Dim xdown As Single
Dim ydown As Single
Dim rotatexy As D3DVECTOR
'''''''
```

```
Const pi As Single = 3.14159265358979  'can be worked out using (4*atn(1))
Const Rad = pi / 180 'Multiply degrees by this much to get result in radians
'#########
'## FONTS ##
'#########
Dim MainFont As D3DXFont
Dim MainFontDesc As IFont
Dim TextRect As RECT
Dim fnt As New StdFont
'#########
'## MESH ##
'########
Dim Mesh As D3DXMesh
Dim Mesh2 As D3DXMesh
Dim mesh3 As D3DXMesh
Dim mesh4  As D3DXMesh
'######################
'## TEXTURES AND MATERIALS ##
'######################
Dim MeshMaterials() As D3DMATERIAL8  ' Mesh Material data
Dim MeshTextures() As Direct3DTexture8 ' Mesh Textures
Dim meshmaterialsgan() As D3DMATERIAL8
Dim meshtexturesgan() As Direct3DTexture8
Dim meshmaterialsgan2() As D3DMATERIAL8
Dim meshtexturesgan2() As Direct3DTexture8
Dim meshmaterialsfloor() As D3DMATERIAL8
Dim meshtexturesfloor() As Direct3DTexture8
Dim nMaterials As Long 'How may materials/textures we have....
Dim nmaterials2 As Long
Dim nmaterials3 As Long
Dim nmaterials4 As Long
'############
'## LIGHTING ##
'###########
Dim Lights(0 To 2) As D3DLIGHT8 'We're going to set up 3 lights.
Dim CurrentActiveLight As Integer 'So we can cycle through them....
'############
'## Background ##
Dim background As Direct3DSurface8
Dim bufferbackground As Direct3DSurface8
'###########
'############
'##  MATRICES ##
'############
Dim matWorld As D3DMATRIX '//How the vertices are positioned
Dim matView As D3DMATRIX '//Where the camera is/where it's looking
Dim matProj As D3DMATRIX '//How the camera projects the 3D world onto the 2D screen
Dim matworld2 As D3DMATRIX
Dim matworld3 As D3DMATRIX
'######################
'## FRAME RATE CALCULATIONS ##
'######################
Private Declare Function GetTickCount Lib "kernel32" () As Long
Dim FPS_LastCheck As Long
Dim FPS_Count As Long
Dim FPS_Current As Single
'// Initialise : This procedure kick starts the whole process.
'// It'll return true for success, false if there was an error.
Public Function Initialise() As Boolean
On Error GoTo ErrHandler:
Dim DispMode As D3DDISPLAYMODE '//Describes our Display Mode
Dim d3dwindow As D3DPRESENT_PARAMETERS '//Describes our Viewport
Set Dx = New DirectX8  '//Create our Master Object
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
Set D3D = Dx.Direct3DCreate() '//Make our Master
Object create the Direct3D Interface
Set D3DX = New D3DX8 '//Create our helper
library...
'//We're going to use Fullscreen mode because I prefer
it to windowed mode :)
   Dim mode As D3DDISPLAYMODE
   D3D.GetAdapterDisplayMode
D3DADAPTER_DEFAULT, mode
   ' Set up the structure used to create the D3DDevice.
Since we are now
   ' using more complex geometry, we will create a
device with a zbuffer.
   'the D3DFMT_D16 indicates we want a 16 bit z
buffer but
   'Dim d3dwindow As
D3DPRESENT_PARAMETERS
   d3dwindow.Windowed = 1
   d3dwindow.SwapEffect =
D3DSWAPEFFECT_COPY_VSYNC
   d3dwindow.BackBufferFormat = mode.Format
   d3dwindow.BackBufferCount = 1
   d3dwindow.EnableAutoDepthStencil = 1
   d3dwindow.AutoDepthStencilFormat =
D3DFMT_D16
   Set D3DDevice =
D3D.CreateDevice(D3DADAPTER_DEFAULT,
D3DDEVTYPE_HAL, Picture1.hWnd, _

D3DCREATE_SOFTWARE_VERTEXPROCESSIN
G, d3dwindow)

Set background =
D3DDevice.CreateImageSurface(400, 417,
mode.Format)
D3DX.LoadSurfaceFromFile background, ByVal 0,
ByVal 0, App.Path & "\background.bmp", ByVal 0,
D3DX_DEFAULT, 0, ByVal 0
Set bufferbackground = D3DDevice.GetBackBuffer(0,
D3DBACKBUFFER_TYPE_MONO)
'//Transformed and lit vertices dont need lighting
'   so we disable it...
D3DDevice.SetRenderState D3DRS_LIGHTING, 1
'//We need to enable our Z Buffer
D3DDevice.SetRenderState D3DRS_ZENABLE, 1
D3DDevice.SetRenderState D3DRS_AMBIENT,
&H202020
'The ambient value is a hex-RRGGBB code
D3DDevice.SetTextureStageState 0,
D3DTSS_MAGFILTER, D3DTEXF_LINEAR
D3DDevice.SetTextureStageState 0,
D3DTSS_MINFILTER, D3DTEXF_LINEAR
D3DDevice.SetTextureStageState 0,
D3DTSS_BORDERCOLOR, 0
'############
'##  MATRICES ##
'############
'//1. The World Matrix
D3DXMatrixIdentity matWorld
D3DDevice.SetTransform D3DTS_WORLD,
matWorld 'commit this matrix to the device
'//2. The View Matrix
D3DXMatrixLookAtLH matView, MakeVector(0,
100, 200), MakeVector(0, 0, 0), MakeVector(0, 1, 0)
D3DDevice.SetTransform D3DTS_VIEW, matView
'//3. The projection Matrix
D3DXMatrixPerspectiveFovLH matProj, pi / 4, 1, 0.1,
500
D3DDevice.SetTransform D3DTS_PROJECTION,
matProj
'##########
'## FONTS ##
'#########
fnt.Name = "Verdana"
fnt.Size = 12
Set MainFontDesc = fnt
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
Set MainFont = D3DX.CreateFont(D3DDevice,
MainFontDesc.hFont)
'############
'## TEXTURES ##
'############
If InitialiseGeometry() = False Then GoTo
ErrHandler: 'We must create the geometry - otherwise
there aint no point in going on
If SetupLights() = False Then GoTo ErrHandler: 'We
must also set up the lights, being a lighting example
it's pointless to run it unless they're working.
Initialise = True
Exit Function
ErrHandler:
'//We failed; for now we wont worry about why.
Debug.Print "Error Number Returned: " & Err.Number
Initialise = False
End Function
Public Sub Render1()
'//0. Any variables required
   Dim I As Long 'Loop variable
'//1. We need to clear the render device before we can
draw anything
'     This must always happen before you start
rendering stuff...
   'All rendering calls go between these two lines
   '//Draw the Object
   For I = 0 To nMaterials - 1
     ' //Setup the renderer for this part of the mesh
      D3DDevice.SetMaterial MeshMaterials(I)
      D3DDevice.SetTexture 0, MeshTextures(I)
      '//Draw the current part of the mesh
      Mesh.DrawSubset I
   Next I
   '//Draw the frame rate
   TextRect.Top = 0
   TextRect.bottom = 20
   TextRect.Right = 75
   D3DX.DrawText MainFont, &HFFFFCC00,
CStr(FPS_Current) & "msec", TextRect, DT_TOP Or
DT_LEFT
   TextRect.Top = 20
   TextRect.bottom = 40
   TextRect.Right = 250
   D3DX.DrawText MainFont, &HFFFFCC00,
"Current Active Light: " & CurrentActiveLight + 1,
TextRect, DT_TOP Or DT_LEFT
   TextRect.Top = 40
   TextRect.bottom = 60
   TextRect.Right = 250
   D3DX.DrawText MainFont, &HFFFFCC00, "Press
1 - 3 keys to change", TextRect, DT_TOP Or
DT_LEFT
   TextRect.Top = 250
   TextRect.bottom = 270
   TextRect.Right = 350
   D3DX.DrawText MainFont, &HFFFFCC00, "
Inverted Pendulum 3D Simulation", TextRect,
DT_TOP Or DT_LEFT
End Sub

Private Sub Command1_Click()
bRunning = False
Erase MeshMaterials
Erase MeshTextures
Erase meshmaterialsgan
Erase meshtexturesgan
Erase meshmaterialsgan2
Erase meshtexturesgan2
Erase meshmaterialsfloor
Erase meshtexturesfloor
Set Mesh = Nothing
Set Mesh2 = Nothing
Set mesh3 = Nothing
Set mesh4 = Nothing
Set D3DDevice = Nothing
Set D3D = Nothing
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
Set Dx = Nothing
Unload frmMain
End Sub
Private Sub Command2_Click()
P1 = Val(Text1.Text)
I1 = Val(Text2.Text)
D1 = Val(Text3.Text)
P2 = Val(Text4.Text)
I2 = Val(Text5.Text)
D2 = Val(Text6.Text)
End Sub
Private Sub Command3_Click()
form1.Show
condition = True

End Sub
Private Sub Command4_Click()
If condition = True Then
booleancontrol = True
Else
booleancontrol = False
MsgBox "Please set Motor and pendulum Parameter"
End If
End Sub
Private Sub Command5_Click()
booleancontrol = False
W1 = 0
W1o = 0
W1oo = 0
mv1o = 0
mv2o = 0
mv1 = 0
mv2 = 0
w = 0
wo = 0
wp = 0
wpo = 0
zeta2 = 0
zeta2new = 0
zeta1 = 0
zeta1o = 0
zeta2oo = 0
zeta2o = 0
error1 = 0
error2 = 0
error1o = 0
error2o = 0
error2oo = 0
End Sub
Private Sub Command6_Click()
zeta1 = Val(Text7.Text)
setpointzeta1 = Val(Text8.Text)
zeta2 = Val(Text9.Text)
zeta1 = Abs(zeta1) * pi / 180
setpointzeta1 = Abs(setpointzeta1) * pi / 180
zeta2 = Abs(zeta2) * pi / 180
zeta1o = zeta1
zeta2o = zeta2
zeta2oo = zeta2
End Sub
Private Sub Command7_Click()
Dim numcount1 As Single
TChart1.Series(1).Clear
TChart1.Series(2).Clear
T = 0
W1 = 0
W1o = 0
W1oo = 0
mv1o = 0
mv2o = 0
error1 = 0
error2 = 0
error1o = 0
error2o = 0
error2oo = 0
W1 = 0
zeta1 = 0
zeta1o = 0
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
w = 0
wo = 0
wp = 0
wpo = 0
zeta2 = Val(Text9.Text)
zeta2 = Abs(zeta2) * pi / 180
zeta2o = zeta2
zeta2oo = zeta2
Do While T <= 100
  pid1
  pid2
  tfmoter
  If Check1.Value = 1 Then
  tfpendulum1
  End If
  If Check1.Value = 0 Then
  tfpendulum
  End If
  'steptest
  TChart1.Series(1).AddXY T, zeta2, "", clblue
  'TChart1.Series(2).AddXY T, zeta1, "", clred
  T = T + 0.005
  Loop
  zeta1 = 0
End Sub
Private Sub Command8_Click()
speed = speed + 1
If speed > 10 Then
speed = 10
End If
Text10.Text = Str(speed)
End Sub
Private Sub Command9_Click()
speed = speed - 1
If speed < 1 Then
speed = 1
End If
Text10.Text = Str(speed)
End Sub
```

```
Private Sub Form_KeyUp(KeyCode As Integer, Shift
As Integer)
Select Case KeyCode
  Case vbKey1
    CurrentActiveLight = 0
  Case vbKey2
    CurrentActiveLight = 1
  Case vbKey3
    CurrentActiveLight = 2
End Select
End Sub
Private Sub Form_Load()
Dim RotateAngle As Single
Dim matTemp As D3DMATRIX '//To hold temporary
stuff...
Dim TIME As Single
Dim Time2 As Single
Dim numcount As Integer
Me.Show '//Make sure our window is visible
speed = 10
condition = False
W1 = 0
W1o = 0
W1oo = 0
mv1 = 0
mv1o = 0
mv2o = 0
mv2 = 0
error1 = 0
error2 = 0
error1o = 0
error2o = 0
error2oo = 0
w = 0
wo = 0
Ts = 0.005
booleancontrol = False
bRunning = Initialise()
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
Debug.Print "Device Creation Return Code : ",
bRunning 'So you can see what happens...
Do While bRunning
   'We're going to update our world matrix so that the
object does
   'something vaguely interesting... :)
   If booleancontrol = True Then
   For numcount = 1 To speed
   pid1
   pid2
   tfmoter
   tfpendulum
   'steptest
   Next
   End If
   Do While GetTickCount <= TIME + 50
   Loop
   TIME = GetTickCount
   FPS_Current = TIME - Time2 - 5
   Time2 = GetTickCount
   D3DDevice.BeginScene
   D3DDevice.Clear 0, ByVal 0,
D3DCLEAR_TARGET Or D3DCLEAR_ZBUFFER,
&HFFA0A0AF, 1#, 0
   '//Clear the screen black
   D3DDevice.CopyRects background, ByVal 0, 0,
bufferbackground, ByVal 0
'_____'
   D3DXMatrixIdentity matworld3 '//Reset our world
matrix
   D3DXMatrixIdentity matTemp
   D3DXMatrixRotationZ matTemp, Rotatealfa
   D3DXMatrixMultiply matworld3, matworld3,
matTemp
   D3DXMatrixIdentity matTemp
   D3DXMatrixRotationX matTemp, -11 / 7
   D3DXMatrixMultiply matworld3, matworld3,
matTemp
   D3DXMatrixIdentity matTemp
   D3DXMatrixScaling matTemp, 0.4, 0.4, 0.4
   D3DXMatrixMultiply matworld3, matworld3,
matTemp
   D3DDevice.SetTransform D3DTS_WORLD,
matworld3
   render4
   Render1
'_____'
   D3DXMatrixIdentity matWorld '//Reset our world
matrix
   D3DXMatrixIdentity matTemp
   D3DXMatrixRotationZ matTemp, Rotatealfa
   D3DXMatrixMultiply matWorld, matWorld,
matTemp
   D3DXMatrixIdentity matTemp
   D3DXMatrixRotationZ matTemp, zeta1
'**************change zeta1**************'
   D3DXMatrixMultiply matWorld, matWorld,
matTemp
   D3DXMatrixIdentity matTemp
   D3DXMatrixRotationX matTemp, -11 / 7
   D3DXMatrixMultiply matWorld, matWorld,
matTemp
   D3DXMatrixIdentity matTemp
   D3DXMatrixScaling matTemp, 0.4, 0.4, 0.4
   D3DXMatrixMultiply matWorld, matWorld,
matTemp
   D3DDevice.SetTransform D3DTS_WORLD,
matWorld
   '//2. Rendering the graphics...
   render2
   D3DXMatrixIdentity matworld2
   D3DXMatrixIdentity matTemp
   D3DXMatrixTranslation matTemp, 0, 0, -162.5
   D3DXMatrixMultiply matworld2, matworld2,
matTemp
   D3DXMatrixIdentity matTemp
   D3DXMatrixRotationY matTemp, zeta2
'"**************change zeta2**************""
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    D3DXMatrixMultiply matworld2, matworld2,
matTemp
    D3DXMatrixIdentity matTemp
    D3DXMatrixTranslation matTemp, 0, 0, 162.5
    D3DXMatrixMultiply matworld2, matworld2,
matTemp
    D3DXMatrixMultiply matworld2, matworld2,
matWorld
    D3DDevice.SetTransform D3DTS_WORLD,
matworld2
    render3
    D3DDevice.EndScene
'//3. Update the frame to the screen...
'     This is the same as the Primary.Flip method as
used in DirectX 7
'     These values below should work for almost all
cases...
D3DDevice.Present ByVal 0, ByVal 0, 0, ByVal 0
    '//Work out which light should be on...
    D3DDevice.LightEnable 0, 0 'Turn them all off
    D3DDevice.LightEnable 1, 0
    D3DDevice.LightEnable 2, 0
    D3DDevice.LightEnable CurrentActiveLight, 1
'Then turn on the correct one
    '//Update the frame...
    '//Frame rate calculations
    DoEvents '//Allow windows time to think; otherwise
you'll get into a really tight (and bad) loop...
Loop '//Begin the next frame...
'//If we've gotten to this point the loop must have been
terminated
'  So we need to clean up after ourselves. This isn't
essential, but it's
'  good coding practise.
On Error Resume Next 'If the objects were never
created;
'                    (the initialisation failed) we might
get an
'                    error when freeing them... which
we need to
'                    handle, but as we're closing
anyway...
Erase MeshMaterials
Erase MeshTextures
Erase meshmaterialsgan
Erase meshtexturesgan
Set D3DDevice = Nothing
Set D3D = Nothing
Set Dx = Nothing
Debug.Print "All Objects Destroyed"
'//Final termination:
'Unload Me
End
End Sub
Private Function InitialiseGeometry() As Boolean
On Error GoTo BailOut: '//Setup our Error handler
'//0. Any variables required  urt
    Dim mtrlBuffer As D3DXBuffer '//Holds some
useful data for us...
    Dim I As Long 'Loop variable
    Dim TextureFile As String 'the texture required
    Dim mtrlBuffer2 As D3DXBuffer
    Dim mtrlbuffer3 As D3DXBuffer
    Dim mtrlbuffer4 As D3DXBuffer
'//1. Get the data from the file
    Set Mesh = D3DX.LoadMeshFromX(App.Path &
"\basecompack.x", D3DXMESH_POINTS,
D3DDevice, Nothing, mtrlBuffer, nMaterials)
    If Mesh Is Nothing Then GoTo BailOut: '//Dont
continue if the above call did not work
'//2. Allocate the space required for the materials and
textures used:
    ReDim MeshMaterials(nMaterials) As
D3DMATERIAL8
    ReDim MeshTextures(nMaterials) As
Direct3DTexture8
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
'//3. Now we fill our arrays with the infformation
required
   For I = 0 To nMaterials - 1
      '//Get D3DX to copy the data that we loaded from
the file into our structure
      D3DX.BufferGetMaterial mtrlBuffer, I,
MeshMaterials(I)
      '//Fill in the missing gaps - the Ambient
properties
      MeshMaterials(I).Ambient =
MeshMaterials(I).diffuse
      '//get the name of the texture used for this part of
the mesh
      TextureFile =
D3DX.BufferGetTextureName(mtrlBuffer, I)
      '//Now create the texture
      If TextureFile <> "" Then 'Dont try to create a
texture from an empty string
         Set MeshTextures(I) =
D3DX.CreateTextureFromFileEx(D3DDevice,
App.Path & "\" & "Yellobrk.bmp", 128, 128,
D3DX_DEFAULT, 0, D3DFMT_UNKNOWN,
D3DPOOL_MANAGED, D3DX_FILTER_LINEAR,
D3DX_FILTER_LINEAR, 0, ByVal 0, ByVal 0)
      End If
   Next I
   ''''''
```

```
'//3. Now we fill our arrays with the information
required
  For I = 0 To nmaterials3 - 1
    '//Get D3DX to copy the data that we loaded from
the file into our structure
    D3DX.BufferGetMaterial mtrlbuffer3, I,
meshmaterialsgan2(I)
    '//Fill in the missing gaps - the Ambient
properties
    meshmaterialsgan2(I).Ambient =
meshmaterialsgan2(I).diffuse
    '//get the name of the texture used for this part of
the mesh
    TextureFile =
D3DX.BufferGetTextureName(mtrlbuffer3, I)
    '//Now create the texture
    If TextureFile <> "" Then 'Dont try to create a
texture from an empty string
      Set meshtexturesgan2(I) =
D3DX.CreateTextureFromFileEx(D3DDevice,
App.Path & "\" & "brkrun.bmp", 128, 128,
D3DX_DEFAULT, 0, D3DFMT_UNKNOWN,
D3DPOOL_MANAGED, D3DX_FILTER_LINEAR,
D3DX_FILTER_LINEAR, 0, ByVal 0, ByVal 0)
    End If
  Next I
  ''''''
```

```
k3 = Ts * ((-BTS * (wo + 0.5 * k2)) - KMGL *
Sin(zeta2) + Kc * Cos(zeta2))
k3 = k3 / Kmoment
k4 = Ts * ((-BTS * (wo + k3)) - KMGL * Sin(zeta2) +
Kc * Cos(zeta2))
k4 = k4 / Kmoment
w = wo + (1 / 6) * (k1 + 2 * k2 + 2 * k3 + k4)
If Abs(Abs(zeta2) - 3.1459) < 0.005 Then
zeta2new = 3.141592654
End If
Do While zeta2new > 6.28318
zeta2new = zeta2new - 6.28318
Loop
Do While zeta2new < 0
zeta2new = zeta2new + 6.28318
Loop
zeta2 = zeta2new
wo = w
wpo = wp
End Sub
Sub tfmoter()
mvtotal = mv1 + mv2
W1 = (Ts * mvtotal * Ts * Kmoter) + (W1o * (Ts *
beta + 2 * alfa)) - (alfa * W1oo)
W1 = W1 / (Ts * Ts + Ts * beta + alfa)
zeta1 = zeta1o + (W1 * Ts)
End Sub
Sub steptest()
W1 = (19 * 0.005 * mv1) + (19 * W1o)
W1 = W1 / (19 + 0.005)
zeta1 = zeta1o + (W1 * 0.005)
zeta1o = zeta1
End Sub
Sub render4()
Dim I As Long 'Loop variable
'//1. We need to clear the render device before we can
draw anything
'     This must always happen before you start
rendering stuff...
'All rendering calls go between these two lines
'//Draw the Object
For I = 0 To nmaterials4 - 1
  ' //Setup the renderer for this part of the mesh
  D3DDevice.SetMaterial meshmaterialsfloor(I)
  D3DDevice.SetTexture 0, meshtexturesfloor(I)
  '//Draw the current part of the mesh
  mesh4.DrawSubset I
Next I
End Sub
Private Sub Picture1_MouseDown(Button As Integer,
Shift As Integer, x As Single, y As Single)
push = True
xdown = x
ydown = y
End Sub
Private Sub Picture1_MouseMove(Button As Integer,
Shift As Integer, x As Single, y As Single)
Dim delx As Single
If push = True Then
delx = x - xdown
delx = delx / 550
Rotatealfa = -delx
End If
End Sub
Private Sub Picture1_MouseUp(Button As Integer,
Shift As Integer, x As Single, y As Single)
push = False
End Sub
Sub tfpendulum1()
wp = W1
Kc = (KML1L2 * (wp - wpo)) / Ts
k1 = Ts * wo
k2 = Ts * (wo + 0.5 * k1)
k3 = Ts * (wo + 0.5 * k2)
k4 = Ts * (wo + k3)
zeta2new = zeta2 + (1 / 6) * (k1 + 2 * k2 + 2 * k3 +
k4)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
k1 = Ts * ((-BTS * wo) - KMGL * Sin(zeta2) + Kc *
Cos(zeta2))
k1 = k1 / Kmoment
k2 = Ts * ((-BTS * (wo + 0.5 * k1)) - KMGL *
Sin(zeta2) + Kc * Cos(zeta2))
k2 = k2 / Kmoment
k3 = Ts * ((-BTS * (wo + 0.5 * k2)) - KMGL *
Sin(zeta2) + Kc * Cos(zeta2))
k3 = k3 / Kmoment
k4 = Ts * ((-BTS * (wo + k3)) - KMGL * Sin(zeta2) +
Kc * Cos(zeta2))
k4 = k4 / Kmoment
w = wo + (1 / 6) * (k1 + 2 * k2 + 2 * k3 + k4)
If Abs(Abs(zeta2) - 3.1459) < 0.005 Then
zeta2new = 3.141592654
End If
zeta2 = zeta2new
wo = w
wpo = wp
End Sub
```


เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้