การปฏิสัมพันธ์ของวัตถุเสมือนในสภาวะแวดล้อมเสมือนจริง

THE INTERACTION OF VIRTUAL OBJECTS IN VIRTUAL ENVIRONMENT

โดย

นายขจรพฐ สว่างดี

นายภิญโญ เทียมหิรัณย์โสภิต

อาจารย์ที่ปรึกษา

ดร.สมศักดิ์ วลัยรัชต์

ดร.อรัญญา วลัยรัชต์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

ปริญญานิพนธ์ ปีการศึกษา 2546

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การปฏิสัมพันธ์ของวัตถุเสมือนในสภาวะแวดล้อมเสมือนจริง

THE INTERACTION OF VIRTUAL OBJECTS IN VIRTUAL ENVIRONMENT

คณะผู้จัดทำ นายขจรพฏ สว่างดี รหัส 44015317

นายภิญโญ เทียมหิรัณย์โสภิต รหัส 44015340

..... อาจารย์ที่ปรึกษา

(ดร.สมศักดิ์ วลัยรัชต์)

..... อาจารย์ที่ปรึกษา

(ดร.อรัญญา วลัยรัชต์)

การปฏิสัมพันธ์ของวัตถุเสมือนในสภาวะแวดล้อมเสมือนจริง

นายขจรพฏ สว่างดี 44015317
นายภิญโญ เทียมหิรัณย์โสภิต 44015340
ดร.สมศักดิ์ วลัยรัชต์ อาจารย์ที่ปรึกษา
ดร.อรัญญา วลัยรัชต์ อาจารย์ที่ปรึกษา
ปีการศึกษา 2546

บทคัดย่อ

ปัจจุบันการสร้างโลกเสมือน (Virtual world) มีอยู่ทั่วไปทั้งที่เอาใช้งานหรือใช้เพื่อความสนุกสนาน โดยสามารถทำการจำลองวัตถุต่างๆหรือแม้แต่สภาพแวดล้อมที่เกิดขึ้นจากสถานที่ต่างๆมาไว้ในที่เดียวกัน โดยการใช้กฎทางกายภาพเข้ามาช่วยในการจำลองการเคลื่อนที่หรือแม้แต่โครงสร้างของวัตถุเองก็ตาม ซึ่งทุกอย่างนั้นสามารถพิสูจน์ออกมาได้ด้วยทฤษฎีฟิสิกส์ ปริญญานิพนธ์ฉบับนี้จึงขอนำเสนอระบบปฏิสัมพันธ์ของวัตถุเสมือนในสภาวะแวดล้อมเสมือนจริง

เนื้อหาในปริญญานิพนธ์ฉบับนี้แบ่งเป็น 2 ส่วนคือ ส่วนของการพัฒนาซอฟท์แวร์ และส่วนของการพัฒนาฮาร์ดแวร์ โดยในส่วนซอฟท์แวร์จะเป็นการสร้างโลกเสมือนในรูปแบบคอมพิวเตอร์กราฟิก 3 มิติที่จำลองโลกเสมือนขึ้น และในส่วนฮาร์ดแวร์เป็นส่วนที่พัฒนาอุปกรณ์สำหรับการอ่านตำแหน่งนิ้วมือที่สามารถสร้างแรงต้านกลับได้มีชื่อเรียกว่า SPIDAR (SPace Interface Device for Artificial Reality)

ระบบนี้พัฒนาขึ้นบนระบบปฏิบัติการวินโดว์ และใช้ชุดคำสั่งเพื่อการพัฒนาโปรแกรม DirectX ภาษาคอมพิวเตอร์ในการพัฒนาโปรแกรมคือ Visual C++ ลักษณะการทำงานระบบนี้อนุญาตผู้ใช้หนึ่งคนสวมแหวน 2 วงที่ SPIDAR ผู้ใช้จะเห็นภาพโลกเสมือนบนจอภาพคอมพิวเตอร์ซึ่งเป็นภาพคอมพิวเตอร์กราฟิก 3 มิติซึ่งมีวัตถุรูปทรงต่างๆอยู่ เมื่อผู้ใช้เคลื่อนนิ้วมือ จะส่งผลให้นิ้วมือเสมือนในจอภาพเคลื่อนที่ตาม และเมื่อไปสัมผัสกับวัตถุที่อยู่ในภาพวัตถุนั้นจะเกิดการเคลื่อนไปตามทิศทางและแรงที่กระทำ พร้อมทั้งผู้ใช้จะสามารถรับรู้ผลที่เกิดขึ้นได้โดยจะมีแรงต้านกลับ (Force Feed Back) มายังผู้ใช้ เช่น กรณีที่ลูกบอลกระเด็นมาสัมผัสนิ้วมือ

THE INTERACTION OF VIRTUAL OBJECTS IN VIRTUAL ENVIRONMENT

Mr. Khajonpot Sawangdee	44015317
Mr. Pinyo Thiamhirunsophit	44015340
Dr. Somsuk Varairuch	Advisor
Dr. Arrunya Varairuch	Advisor
Academic Year 2003	

ABSTRACT

Nowadays, the construct of virtual word is in general for works or entertain. It's can simulate objects or environments from many place to in one place by use the theory of Physically-based Interaction for help to simulate to move or the structure of objects. That can to explain by theory of physicals. This thesis is proposed The interaction of virtual objects in virtual environment

The content of the thesis can be divided into 2 parts. First part is describing about the development of the software to create virtual world with computer graphics 3D. The second part explains about the implementation of hardware system including the part of generating user's feedback. It's name SPIDAR (Space Interface Device for Artificial Reality).

The system is designed based on Windows operating system with DirectX library for graphics display. Software is implemented by using Microsoft Visual C++. One user is allowed to operate the system by put 2 rings are in SPIDAR. User can see virtual world on screen and manipulate the virtual objects. And can receive force feedback when touch them.

กิตติกรรมประกาศ

ปริญญานิพนธ์นี้เสร็จสมบูรณ์ได้จากความช่วยเหลือ และร่วมมือจากหลายๆ ฝ่ายด้วยกัน บุคคลแรกที่ต้องกล่าวถึงเพราะเป็นส่วนสำคัญที่ทำให้วิทยานิพนธ์นี้เสร็จลงได้ก็คือ อาจารย์สมศักดิ์ วลัยรัชต์ อาจารย์ที่ปรึกษา อาจารย์อรัญญา วลัยรัชต์ อาจารย์ที่ปรึกษาร่วม ที่ให้ความเอาใจใส่ แนะนำ และช่วยเหลือเสมอมา ซึ่งต้องขอขอบพระคุณเป็นอย่างมาก

ขอขอบคุณทางภาควิชาวิศวกรรมคอมพิวเตอร์ที่ได้จัดเตรียมสิ่งอำนวยความสะดวก เพื่อช่วยให้การวิจัยเป็นไปด้วยความสะดวกและรวดเร็ว รวมทั้งยังมีอินเตอร์เน็ตความเร็วสูงให้บริการสำหรับค้นหาข้อมูลความรู้

และต้องขอบคุณ พี่แบด รุ่นพี่ป.โท ที่ให้คอยแนะนำการทำงาน และเพื่อนๆ ที่คอยสร้างความสนุกสนานในการทำงาน และที่จะขาดไม่ได้ห้อง MML ที่ให้สถานที่ในการทำงาน และฝ่ายสเบียงที่ช่วยให้ท้องหายหิวจนปริญญานิพนธ์สำเร็จลุล่วงด้วยดี

สุดท้ายขอขอบพระคุณบุคคลสำคัญที่สุดที่ทำให้ข้าพเข้ามีวันนี้ คือ บิดา มาดา อันเป็นที่เคารพยิ่ง ซึ่งได้เลี้ยงดู ข้าพเจ้ามาเป็นอย่างดี พร้อมทั้งให้โอกาสในการศึกษาอย่างเต็มที่ และยังให้กำลังใจ เอาใจใส่เสมอมา ในทุกๆ ด้านอันหาที่เปรียบมิได้ ข้าพเจ้าขอระลึกในพระคุณอันสุดประมาณ และขอกราบขอบพระคุณมา ณ ที่นี้

ขจรพฎ สว่างดี
ภิญโญ เทียมหิรัณย์โสภิต

สารบัญ

สารบัญรูปภาพ

หน้า

บทที่ 1
บทนำ

1.1 แรงจูงใจ

ในทุกวันนี้การพัฒนาสื่อทางด้านมัลติมีเดียได้รับการพัฒนาไปอย่างรวดเร็ว ไม่ว่าจะเป็นด้านการแสดงผลด้วยกราฟิก 3 มิติ ระบบเสียงรอบทิศทาง หรือจะเป็นอุปกรณ์ที่ใช้ในการสร้างความเสมือนจริง (Virtual reality เรียกสั้นๆ ว่า VR) ที่มีหลากหลายมากขึ้น และยังมีการพัฒนาระบบจำลอง (Simulation system) โดยผนวกเข้ากับ VR ยิ่งทำให้ได้รับการสนใจเพิ่มขึ้นไปอีก

โดยเฉพาะการแสดงผลด้วยกราฟิก 3 มิติที่ได้รับความสนใจนำไปใช้งานอย่างแพร่หลาย ไม่ว่าจะเป็นด้านการออกแบบผลิตภัณฑ์ ด้านความบันเทิง ศิลปะ ด้านการวิจัย ด้วยเหตุผลที่กราฟิก 3 มิติสามารถแสดงภาพออกมาให้เห็นนั้นมีความเหมือนหรือใกล้เคียงกับวัตถุจริงๆ มาก ต่างก็แค่เพียงจะเป็นแค่ภาพที่อยู่ในจอภาพเท่านั้นไม่สามารถจับต้องวัตถุ 3 มิตินั้นได้

นี่จึงเป็นแรงจูงใจให้มีการทำโครงงาน การปฏิสัมพันธ์ของวัตถุเสมือนในสภาวะแวดล้อมเสมือนจริง (The Interaction of virtual objects in virtual environment) นี้ขึ้นมา ซึ่งระบบที่สร้างนี้จะมี 2 ส่วนหลักๆ ที่สำคัญๆ คือ ส่วนที่หนึ่งซอฟท์แวร์ระบบจำลองโลกเสมือนที่จะแสดงผลออกมาเป็นคอมพิวเตอร์กราฟิก 3 มิติ โดยจะมีวัตถุรูปทรงพื้นฐานที่เป็นวัตถุเสมือนอยู่ภายใน และที่วัตถุเสมือนนี้จะมีการจำลองให้เป็นวัตถุแข็งที่มีลักษณะทางกายภาพเหมือนวัตถุจริง และส่วนที่สองคือส่วนของฮาร์ดแวร์ที่สามารถอ่านตำแหน่งของนิ้วโดยอาศัยอุปกรณ์อินพุท 3 มิติที่มีชื่อเรียกว่า SPIDAR (SPace Interface Device for Artificial Reality) โดยอุปกรณ์นี้สามารถสร้างแรงต้านกลับ (Forced feedback) มายังผู้ใช้เมื่อสัมผัสวัตถุเสมือนในโลกเสมือนจริง

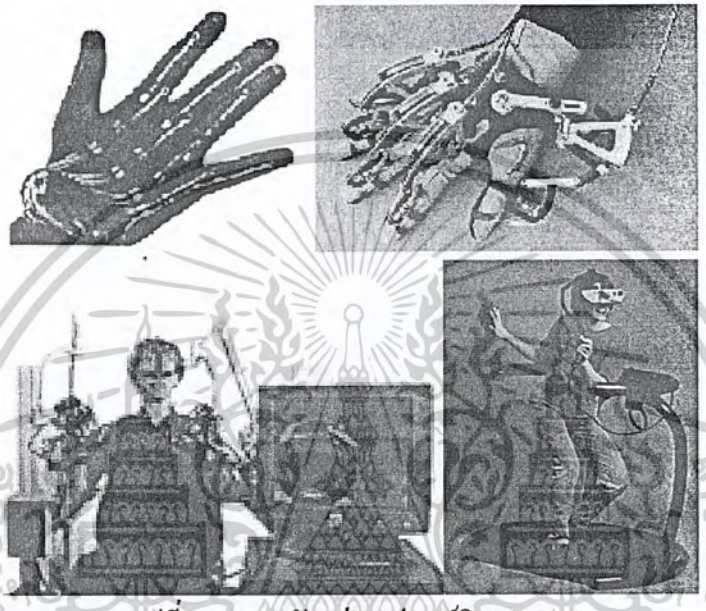
การทำงานของอุปกรณ์จะอนุญาตให้ ผู้ใช้ 1 คนสวมแหวนครอบที่ปลายนิ้วโป้งและนิ้วชี้ (หรือนิ้วใดก็ได้แล้วแต่ความถนัด) เคลื่อนไหวนิ้วมือได้อย่างอิสระ ตัวระบบจำลองจะคำนวณตำแหน่งของนิ้วมือทั้งสองในโลกเสมือนขณะที่เคลื่อนที่ไปตามทิศทางของนิ้วผู้ใช้ เมื่อผู้ใช้ไปจับหรือสัมผัสวัตถุเสมือนอุปกรณ์จะสร้างแรงต้านกลับเพื่อให้ผู้ใช้รู้สึกว่าเสมือนกับได้สัมผัสกับวัตถุนั้นจริงๆ

1.2 ความเสมือนจริง (Virtual Reality)

ความเสมือนจริงหรือที่เรียกกันว่า VR ถือได้ว่าเป็นศาสตร์แขนงหนึ่งที่รวมเอาความรู้และเทคโนโลยีต่างๆ เข้าไว้ด้วยกัน เช่น การแสดงภาพทั้งในรูปแบบ 2 มิติ และ 3 มิติ การแสดงเสียง การได้ยิน การสัมผัสที่ผ่านอุปกรณ์ติดต่อเช่น เมาส์ จอยสติก หรือถุงมือ เป็นต้น

อาจสรุปได้ว่า VR คือวิธีการที่มนุษย์จะใช้ในการมองเห็น จัดการ และโต้ตอบกับคอมพิวเตอร์และข้อมูลที่มีความซับซ้อนสูง โดยในส่วนของการมองเห็นจะเป็นภาพกราฟิกที่สร้างด้วยคอมพิวเตอร์

(Computer Graphic) ที่อาจจะเป็น 2 มิติ หรือ 3 มิติก็ได้ ผนวกเข้ากับการได้ยินเสียงหรือส่วนรับความสัมผัส ความรู้สึก ที่เป็นผลลัพธ์ออกมาจากโลกเสมือนในคอมพิวเตอร์ ซึ่งอาจจะเป็นโมเดล 3 มิติ ที่จำลองขึ้นตามหลักการทางวิทยาศาสตร์ โดยหัวใจหลักของ VR คือการทำการโต้ตอบระหว่างโลกเสมือนกับผู้ใช้โดยอาศัยการอ่านตำแหน่งของวัตถุต่างๆ ในโลกจริง เช่น มือ เท้า ดวงตา โดยที่ผู้ใช้สามารถโต้ตอบกับโลกเสมือนและสามารถจัดการกับวัตถุในโลกเสมือนได้โดยผ่านวัตถุจริง ซึ่งอาจมีการจำลองการเคลื่อนไหวทางกายภาพของวัตถุเสมือนเพื่อให้มีการตอบสนองให้ใกล้เคียงกับโลกจริงมากที่สุด



รูปที่ 1-1 แสดงตัวอย่างอุปกรณ์อินพุต VR

1.3 ระบบจำลอง (Simulation System)

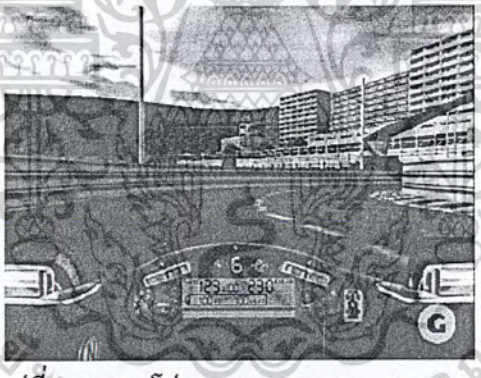
ปัจจุบันระบบจำลอง มีอยู่ทั่วไปทั้งที่นำใช้ในการทำงาน การวิจัย หรือใช้เพื่อความสนุกสนาน โดยสามารถทำการจำลองวัตถุต่างๆหรือแม้แต่สภาพแวดล้อมที่เกิดขึ้นจากสถานที่ต่างๆมาไว้ในที่เดียวกัน โดยทำขึ้นเพื่อให้ใกล้เคียงกับความเป็นจริงมากที่สุด และจากสิ่งที่ VR สามารถทำได้จึงได้มีการนำเอา VR ไปใช้งานในระบบจำลองในหลายๆด้าน โดยงานที่มีการนำเอาการจำลองกฎทางกายภาพนั้นมีมากมายในปัจจุบัน ซึ่งตัวอย่างการนำไปใช้มีดังต่อไปนี้

- เพื่อความบันเทิง เช่น โรงหนัง Simulation theater ที่สร้างเก้าอี้ที่สามารถหมุนหรือโยกเพื่อให้ผู้นั่งรู้สึกเหมือนได้เข้าไปในภาพยนตร์จริงๆ หรือเกมส์ขับรถที่มีพวงมาลัยที่สามารถสร้างแรงต้าน เพื่อทำให้รู้สึกว่าต้องใช้แรงในการหมุนที่เหมือนจริง

รูปที่ 1-2 โรงภาพยนตร์ Simulation Theater

รูปที่ 1-3 เกมส์รถแข่งที่มีที่นั่งและพวงมาลัยเลียนแบบของจริง

- ทางด้านการกีฬา เช่น ระบบจำลองการขับรถแข่งของนักแข่ง F1 เพื่อใช้วิเคราะห์สภาพสนามและแนวทางในการวิ่ง อย่างเช่นการเข้าโค้งเพื่อลดความเสี่ยงไปด้วยในตัว

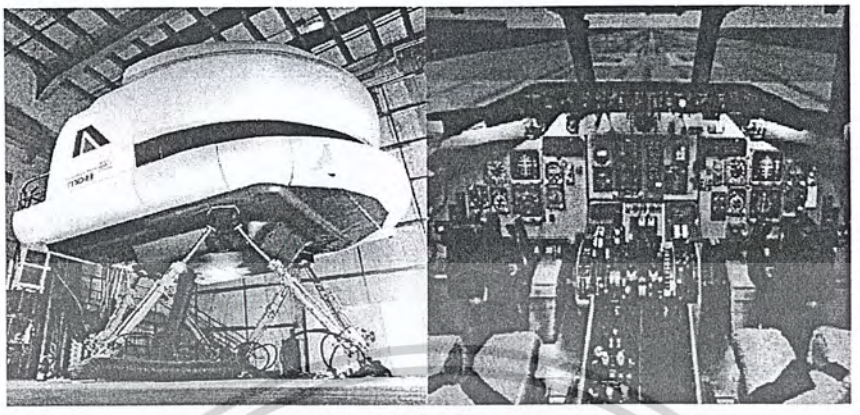


รูปที่ 1-4 แสดงโปรแกรม F1 Racing simulator

- เพื่อใช้ในการวิจัย เช่น การจำลองการเดินทางในอวกาศ การสร้างสภาวะไร้น้ำหนักหรือสุญญากาศเพื่อทดสอบชุดนักบินอวกาศ

รูปที่ 1-5 ระบบจำลองการเดินบนดาวอังคารของ NASA

- ทางการทหาร เช่นการฝึกหัดนักบิน โดยการสร้างสภาพจำลองทางอากาศรูปแบบต่างๆ เพื่อให้เสมือนการบินจริงๆ

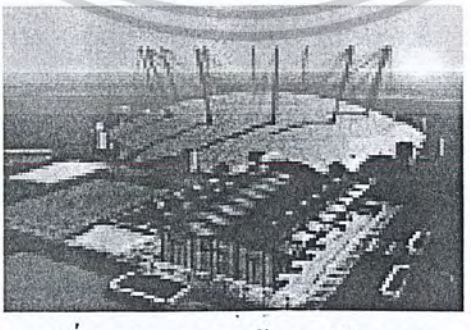


รูปที่ 1-6 ระบบจำลองการบินของทางทหาร

- ทางการแพทย์ เช่นการฝึกการผ่าตัดสามารถทำได้โดยจำลองการผ่าตัด โดยที่ไม่ต้องไปฝึกในเหตุการณ์จริงๆ

รูปที่ 1-7 ระบบจำลองการผ่าตัด

- งานที่มีความเสี่ยงสูง เช่นการกู้วัตถุระเบิด หรือการจำลองการสร้างตึกเพื่อคำนวณโครงสร้างป้องกันตึกถล่ม



รูปที่ 1-8 จำลองการสร้างสถาปัตยกรรม

1.4 วัตถุประสงค์ของโครงงาน

- สร้างอุปกรณ์ติดต่อสื่อสารระหว่างผู้ใช้สำหรับการทำงานในโลกเสมือนจริง (SPIDAR)
- ออกแบบสร้างสภาพแวดล้อมเสมือนจริงในรูปแบบกราฟิก 3 มิติ โดยใช้ Direct3D
- ศึกษาและจำลอง การปฏิสัมพันธ์ทางกายภาพของวัตถุเสมือนในสภาวะแวดล้อมเสมือนจริง โดยใช้หลักการฟิสิกส์ (Physically-Based Simulation)

1.5 ขอบเขตของการพัฒนา

- ฮาร์ดแวร์ที่สามารถสร้างแรงต้านกลับได้โดยใช้ SPIDAR ซึ่งจัดว่าเป็น String-Based Haptic Display โดยใช้มอเตอร์ดึงนิ้วมือผู้ใช้ด้วยเชือก โดยอาศัยความยาวเชือกและตำแหน่งของมอเตอร์เพื่อใช้ในการคำนวณตำแหน่ง 3 มิติของนิ้วมือผู้ใช้
- สร้างโลกเสมือนด้วยคอมพิวเตอร์กราฟิกแบบ 3 มิติ ด้วยชุดคำสั่งกราฟิกไลบารี่ Direct3D
- ระบบ Simulation ที่สามารถแสดงการตอบสนองระหว่างผู้ใช้กับวัตถุจำลอง 3 มิติตามหลักคุณสมบัติทางฟิสิกส์ (Physically-based Interaction)

1.6 เนื้อหาของปริญญานิพนธ์

ปริญญานิพนธ์ฉบับนี้จะเริ่มเนื้อหาที่บทที่ 2 ที่จะอธิบายถึงทฤษฎีและหลักการของภาพ 3 มิติแล้วจะเข้าสู่การแนะนำตัว Direct3D ซึ่งเป็นไลบารี่หลักที่ใช้ในการแสดงผลภาพ 3 มิติ และเมื่อเข้าใจถึงหลักการของภาพ 3 มิติแล้วส่วนต่อไปก็จะเป็นการอธิบายทฤษฎีที่ใช้ในการจำลองวัตถุโดยใช้คุณสมบัติของฟิสิกส์ และการตรวจสอบการชนกันของวัตถุ ในส่วนบทที่ 3 จะกล่าวถึงการพัฒนาระบบในส่วนของฮาร์ดแวร์ โดยจะพูดถึงความเป็นมา ส่วนประกอบของฮาร์ดแวร์ และหลักการทำงานของฮาร์ดแวร์ ต่อมาในบทที่ 4 จะกล่าวถึงการพัฒนาในส่วนซอฟท์แวร์ขั้นตอนการสร้างโลกเสมือน ขั้นตอนการสร้างการจำลองการกระทำระหว่างวัตถุ ผลการทำงานของระบบโดยกำหนดรูปแบบการกระทำให้หลากหลาย เช่น การจับวัตถุ ยก หมุน กด เป็นต้น และในบทสุดท้ายจะเป็นการสรุปการพัฒนาโครงงานนี้ ปัญหาและอุปสรรคและแนวทางการพัฒนาต่อในอนาคต

บทที่ 2
ทฤษฎีและหลักการ

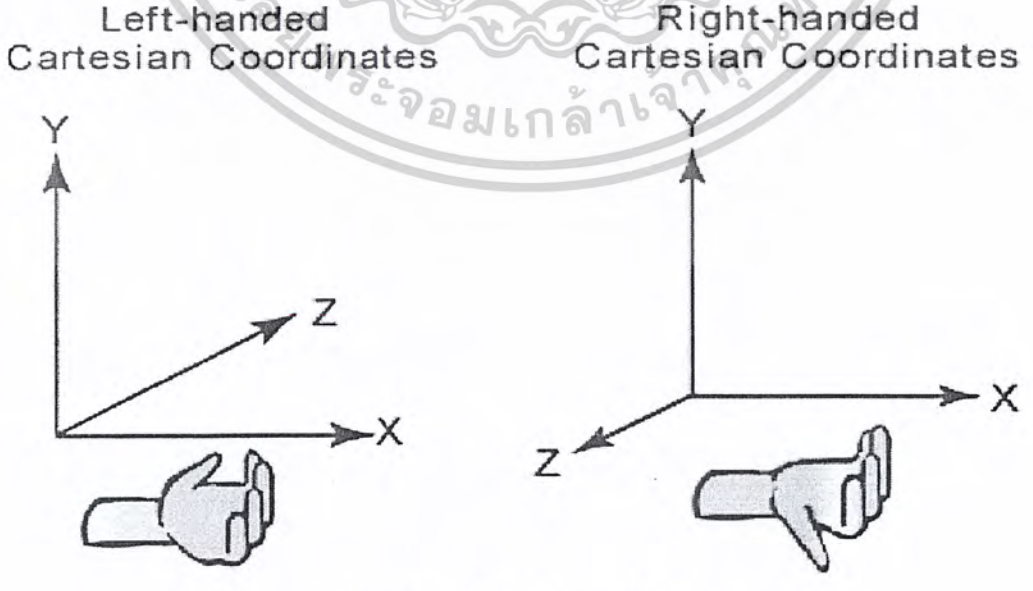
ในบทนี้จะแบ่งเนื้อหาออกเป็น 4 ส่วนโดยส่วนที่ 1 จะกล่าวถึงทฤษฎีและหลักการของภาพ 3 มิติ ส่วนที่ 2 จะเป็นการแนะนำ Direct3D ซึ่งไลบารี่สำคัญที่ใช้ในการแสดงผลภาพ 3 มิติ ในส่วนที่ 3 จะเป็นทฤษฎีการจำลองสภาพแวดล้อมตามคุณสมบัติของฟิสิกส์ และส่วนสุดท้ายจะกล่าวถึงการตรวจสอบการชนกันของวัตถุ

2.1 ทฤษฎีและหลักการภาพ 3 มิติ

ในการพัฒนาโปรแกรมกราฟิก 3 มิติ จำเป็นอย่างยิ่งที่จะต้องเข้าใจถึงทฤษฎีพื้นฐานที่จำเป็นสำหรับนำไปประยุกต์ใช้ โดยเฉพาะทฤษฎีทางด้านคอมพิวเตอร์กราฟิก ซึ่งเป็นพื้นฐานสำหรับการพัฒนาโปรแกรม 3 มิติ

2.1.1 ระบบพิกัด 3 มิติ

ระบบพิกัด หมายถึงทิศทางของค่าในแนวแกน x, y และ z ของจุดกำเนิด (Origin) ในระบบ 3 มิติ ซึ่งจุดกำเนิดจะมีค่า x, y และ z เป็น (x=0, y=0, z=0) และโดยปกติแล้วทิศทางของค่า x จะเพิ่มขึ้นในทิศทางไปทางด้านขวา และลดลงไปทางด้านซ้ายของจุดกำเนิด ส่วนค่า y จะเพิ่มขึ้นไปทางด้านบน และลดลงไปทางด้านล่าง แต่ค่า z จะมีให้เลือกอยู่ 2 แบบ คือแบบแรกจะเพิ่มขึ้นในทิศทางเข้าไปในจอภาพ และแบบที่สองจะเพิ่มขึ้นในทิศทางออกจากจอภาพ โดยแบบแรกเรียกว่า Left-Handed Cartesian Coordinate System ส่วนแบบที่สองจะเรียกว่า Right-Handed Cartesian Coordinate System

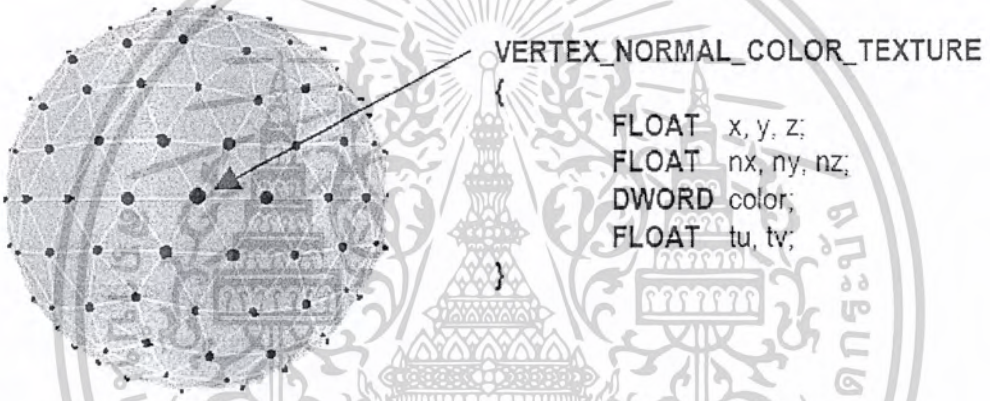


รูปที่ 2-1 แสดงระบบพิกัดคาร์ทีเซียนแบบมือซ้ายและแบบมือขวา

โดยทั่วไป Direct3D นั้นระบบพิกัดที่ใช้จะเป็นระบบมือซ้าย ในการแสดงผลภาพ 3 มิตินั้นเราจะต้องส่งเวอร์เท็กซ์อาเรย์ที่ต้องการแสดงผลเข้าไปใน Geometry Pipeline สุดท้ายก็จะได้ภาพ 2 มิติออกมาแสดงผลบนจอภาพ เหตุที่ต้องแปลงข้อมูลที่เป็น 3 มิติให้เป็น 2 มิตินั้นเพราะหน้าจอของเราไม่สามารถแสดงภาพ 3 มิติตรงๆ ได้จึงต้องทำการส่งข้อมูลเข้า Geometry Pipeline เพื่อให้ได้ผลลัพธ์เป็นภาพ 2 มิติที่สอดคล้องกับข้อมูลนั้นบนมุมมองที่ต้องการ

2.1.2 เวอร์เท็กซ์ (Vertex)

เวอร์เท็กซ์เป็นจุดบนพิกัดในระบบ 3 มิติ ซึ่งใช้ในการอ้างอิงตำแหน่ง ซึ่งในการประมวลผลทางด้านกราฟิกนั้นมักจะกำหนดคุณสมบัติเพิ่มเติมให้กับเวอร์เท็กซ์ เช่น Color, Texture Coordinate เป็นต้น ซึ่งคุณสมบัติเหล่านี้เมื่อประกอบกับตำแหน่งของเวอร์เท็กซ์แล้วจะนำไปใช้ในการเก็บข้อมูลที่แทนแบบจำลอง 3 มิติได้ โดยแสดงดังรูปตัวอย่าง

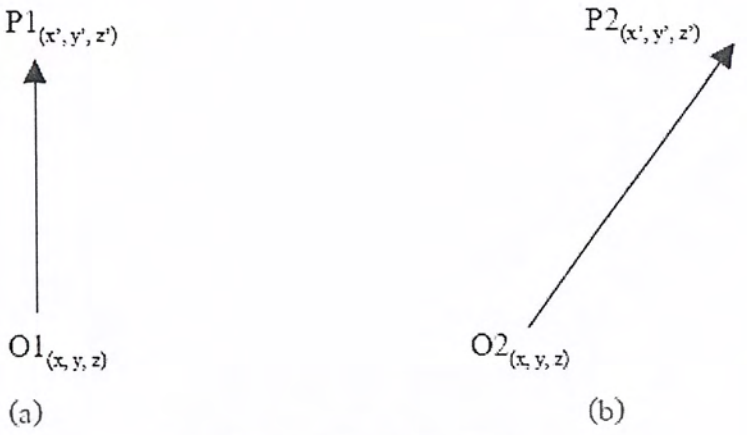


รูปที่ 2-2 แสดงตัวอย่างการนำเวอร์เท็กซ์มาใช้ในการเก็บข้อมูลที่แทนแบบจำลอง 3 มิติ

2.1.3 เวกเตอร์ (Vector)

เวกเตอร์เป็นปริมาณที่มีทั้งขนาดและทิศทาง เป็นองค์ประกอบทางคณิตศาสตร์ที่มีความสำคัญมากด้านการประยุกต์ใช้ในงานการประมวลผลกราฟิก ซึ่งจะต่างกับปริมาณสเกลาร์ตรงที่เวกเตอร์นั้นจะมีทิศทาง หรือกล่าวได้อีกอย่างนึ่งว่าเวกเตอร์มีองค์ประกอบมากจากปริมาณสเกลาร์ n ตัว เพื่อแทนขนาดและทิศทางในระบบ n มิติ

เรามักจะแทนเวกเตอร์โดยใช้สัญลักษณ์ $\overrightarrow{OP}$ โดยหมายถึงเวอกเตอร์นี้มีทิศทางจากจุด O ไปยังจุด P และมีขนาดเท่ากับระห่างระหว่างจุด O และจุด P



รูปที่ 2-3 แสดงตัวอย่างเวกเตอร์

หากเราทำการบวกเวคเตอร์ทั้งสองจะได้ค่าดังนี้

$$R = V1 + V2$$
$$R = (V1_x + V2_x, V1_y + V2_y, V1_z + V2_z)$$

เมื่อทำการคูณเวกเตอร์ด้วยปริมาณสเกลาร์ ซึ่งทำได้โดยการคูณปริมาณสเกลาร์กับทุกคอมโพเน็นต์ของเวกเตอร์ๆ จะได้

$$V * s = (V_x * s, V_y * s, V_z * s)$$

สำหรับขนาด (Magnitude/length) ของเวกเตอร์ใดๆ นั้น สามารถเขียนแทนด้วย $|V|$ ซึ่งสามารถหาได้โดยการใช้กฎพีทาโกรัส ได้สมการดังนี้

$$|V| = \sqrt{V_x^2 + V_y^2 + V_z^2}$$

เวกเตอร์ที่ขนานกัน (Parallel Vector) หมายถึง เวกเตอร์คู่ใดๆ ที่มีทิศทางเดียวกันหรือตรงกันข้ามกัน

เวกเตอร์ร่วมระนาบ (Coplanar Vector) หมายถึง เวกเตอร์ตั้งแต่ 3 เวกเตอร์ขึ้นไปที่อยู่ในระนาบเดียวกัน

2.1.4 เมทริกซ์ (Matrix)

เมทริกซ์ $m \times n$ ประกอบด้วยจำนวนจริงที่เขียนเรียงเป็นแถว (Row) จำนวน m แถว และเขียนในแนวดิ่งจำนวน n หลัก (Column) ถูกปิดล้อมด้วยเครื่องหมาย [ ] หรือ ( ) จำนวนจริงแต่ละจำนวนในเมทริกซ์เรียกว่า สมาชิกของเมทริกซ์

$\begin{pmatrix} 2 & 1 & 0 \\ 5 & 3 & 2 \end{pmatrix}$ เป็นเมทริกซ์มิติ 2×3 $\quad$ $\begin{pmatrix} 1 \\ 0 \\ 2 \\ 4 \end{pmatrix}$ เป็นเมทริกซ์มิติ 4×1

ถ้าเมทริกซ์ที่มีจำนวนแถวและจำนวนหลักเท่ากันเท่ากับ n จะเรียกเมทริกซ์นั้นว่า เมทริกซ์จัตุรัสมิติ n และเรียก $a_{11}, a_{22}, ..., a_{nn}$ ว่าเป็นสมาชิกในแนวทแยงของ A เมื่อ A เป็นเมทริกซ์จัตุรัสมิติ n

a_{11}, a_{22}, a_{33} เป็นสมาชิกในแนวทแยงของ A ซึ่งสมาชิกทุกตัวนอกจากแนวทแยงเป็นศูนย์ทั้งหมด เรียก A ว่าเป็น เมทริกซ์ทแยง (Diagonal Matrix)

ให้ $A = (a_{ij})_{m \times n}$ และ $B = (b_{ij})_{m \times n}$ เป็นเมทริกซ์ที่มีมิติเท่ากัน เรียก A เท่ากับ B ก็ต่อเมื่อ $a_{ij} = b_{ij}$ สำหรับ $1 \le i \le m, 1 \le j \le n$ และเขียนแทนด้วย $A = B$

ถ้า $A = (a_{ij})_{m \times n}$ และ $B = (b_{ij})_{m \times n}$ แล้วผลบวกของ A และ B เขียนแทนด้วย $A + B$ หมายถึง $C = (c_{ij})_{m \times n}$ ซึ่ง

$$c_{ij} = a_{ij} + b_{ij} \quad (1 \le i \le m, 1 \le j \le n)$$

ให้ $A = (a_{ij})_{m \times n}$ เป็นแมทริกซ์ และ k เป็นค่าสเกลาร์จำนวนจริงใดๆ แล้ว ผลคูณสเกลาร์ kA จะเป็นเมทริกซ์ $(ka_{ij})_{m \times n}$ เช่น

ให้ $A = \begin{pmatrix} 2 & 1 & 0 \\ 5 & 3 & 2 \end{pmatrix}$ และ $k = 3$

ดังนั้น $kA = \begin{pmatrix} 3(2) & 3(1) & 3(0) \\ 3(5) & 3(3) & 3(2) \end{pmatrix} = \begin{pmatrix} 6 & 3 & 0 \\ 15 & 9 & 6 \end{pmatrix}$

ให้ $A=(a_{ij})_{m\times p}$ และ $B=(b_{ij})_{p\times n}$ แล้วผลคูณของ A และ B เขียนแทนด้วย AB หมายถึง

$$c=(a_{ij})_{m\times n} \text{ โดย } c_{ij}=a_{i1}b_{1j}+a_{i2}b_{2j}+\ldots+a_{ip}b_{pj}$$

$$c=\sum_{k=1}^{p}a_{ik}b_{kj} \quad (1\le i\le m,\ 1\le j\le n)$$

เช่น
$$AB=\begin{pmatrix}a_{11} & a_{12}\\ a_{21} & a_{22}\end{pmatrix}\begin{pmatrix}b_{11} & b_{12} & b_{13}\\ b_{21} & b_{22} & b_{23}\end{pmatrix}$$
$$AB=\begin{pmatrix}a_{11}b_{11}+a_{12}b_{21} & a_{11}b_{12}+a_{12}b_{22} & a_{11}b_{13}+a_{12}b_{23}\\ a_{21}b_{11}+a_{22}b_{21} & a_{21}b_{12}+a_{22}b_{22} & a_{21}b_{13}+a_{22}b_{23}\end{pmatrix}$$

ผลคูณ AB ไม่มีหรือไม่มีนิยามถ้า A เป็นเมทริกซ์ขนาด $m\times p$ และ B เป็นเมทริกซ์ขนาด $q\times n$ เมื่อ $p\neq q$

การคูณเมทริกซ์นั้นไม่มีคุณสมบัติการสลับที่ซึ่งหมายถึง $A\times B\neq B\times A$ เมื่อ A และ B เป็นเมทริกซ์จัตุรัส

เมทริกซ์เอกลักษณ์ (Identity Matrix) คือเมทริกซ์จัตุรัสที่มีสมาชิกในแนวทะแยงเป็น 1 ทั้ง หมด ซึ่งเขียนแทนด้วย I หรือ I_n แทนเมทริกซ์เอกลักษณ์มิติ n เช่น

$$\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}$$

ถ้า A เป็นเมทริกซ์มิติ $m\times n$ แล้วจะได้ว่า $AI_n=A$ และ $I_nA=A$

B เป็นอินเวิร์สการคูณของเมทริกซ์ A (B เป็นอินเวิรส์ของ A) เมื่อ B เป็นเมทริกซ์ซึ่ง $AB=BA=I$ โดยที่ A เป็นเมทริกซ์จัตุรัสใดๆ

ให้ A เป็นเมทริกซ์จัตุรัสมิติ n จะกล่าวว่า A มีตัวผกผัน (Invertible) หรือ A เป็นเมทริกซ์ซึ่งมิใช่เอกฐาน (Non-Singular Matrix) ถ้าสามารถหาเมทริกซ์จัตุรัส B ได้ซึ่งทำให้

$$AB=I_n=BA$$

และเรียกเมทริกซ์ B ว่าเป็นอินเวิร์สการคูณ ของ A เขียนแทนด้วยสัญลักษณ์ A^{-1} นั่นคือ ถ้า A เป็นเมทริกซ์ซึ่งมิใช่เมทริกซ์เอกฐานมิติ n แล้ว จะได้ว่า

$$AA^{-1}=A^{-1}A=I_n$$

และถ้า A ไม่มีอินเวิร์ส แล้วจะเรียก A ว่าเป็นแมทริกซ์เอกฐาน (Singular Matrix)

2.1.5 การทรานฟอร์มเมชั่น 3 มิติ

คือการเปลี่ยนแปลงวัตถุ 3 มิติจะแบ่งเป็น 3 แบบคือ

2.1.5.1 การเคลื่อนย้ายตำแหน่ง (Translation)

เป็นการเคลื่อนย้ายตำแหน่งของวัตถุ ซึ่งใช้ Translation Matrix ดังนี้

$$\begin{pmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

2.1.5.2 การหมุน (Rotation)

เป็นการหมุนวัตถุรอบแกน X, Y หรือ Z ซึ่งใช้ Rotation Matrix ดังนี้

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

หมุนรอบแกน X

$$\begin{pmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

หมุนรอบแกน Y

$$\begin{pmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

หมุนรอบแกน Z

2.1.5.3 การย่อหรือขยาย (Scaling)

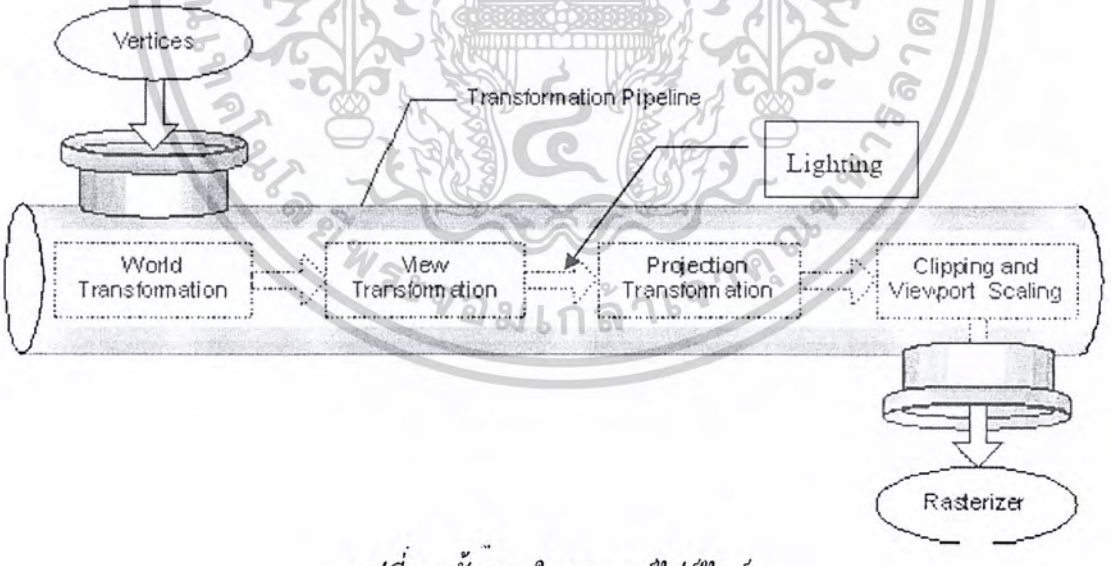
เป็นการย่อหรือขยายวัตถุ ซึ่งใช้ Scaling Matrix ดังนี้

$$\begin{pmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

2.1.6 กระบวนการเรนเดอร์ออบเจ็กท์ 3 มิติ

ในการพัฒนาโปรแกรมประยุกต์ที่ใช้ Direct3D นั้น จะใช้พื้นฐานอยู่บนทฤษฎีเกี่ยวกับ 3 มิติ ได้แก่ เวอร์เท็กซ์, โพลีกอนและคำสั่งที่ใช้ควบคุมข้อมูลเหล่านี้ รวมทั้งการเข้าถึงการแปลงพิกัดของวัตถุ ซึ่งเป็นความสามารถในการเข้าถึงไปป์ไลน์ของกราฟฟิก 3 มิติ ในส่วนของการเปลี่ยนแปลงตำแหน่ง (Transformation), การให้แสง (Lighting) และขั้นตอนการแสดงผลเพื่อสร้างภาพ (Rasterization)

ซึ่งขั้นตอนในกระบวนการเรนเดอร์ออบเจ็กท์ 3 มิติ แบ่งเป็น 2 ขั้นตอน คือ กระบวนการแปรพิกัดและให้แสง (Transformation and Lighting : T&L) ซึ่งเป็นกระบวนการแปรพิกัดของเวอร์เท็กซ์ที่มีรูปแบบทศนิยม ไปสู่พิกัดแบบพิกเซลตามมุมมองของกล้องจำลองภายในซีน รวมถึงกระบวนการให้แสง ขั้นตอนที่ 2 คือ กระบวนการแปรมาสู่พิกเซล (Rasterization) เป็นขั้นตอนลงจุด เส้น และ รูปสามเหลี่ยม ด้วยภาพกราฟฟิกที่ได้จากกระบวนการ T&L

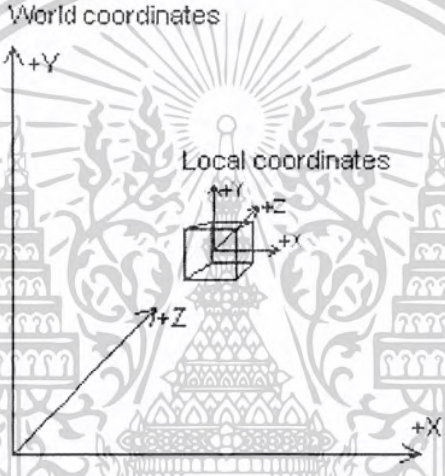


รูปที่ 2-4 ขั้นตอนในเรนเดอร์ไปป์ไลน์

2.1.6.1 กระบวนการแปรพิกัดและให้แสง (Transformation and Lighting : T&L)

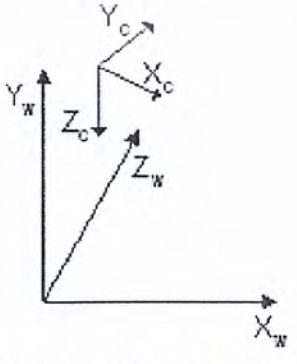
กระบวนการแปรพิกัด (Transformation) จะมีส่วนที่ทำหน้าที่ส่งอินพุทต่างๆ ผ่านไปยังฟังก์ชั่นการทำงานในรูปแบบไปป์ไลน์ของไดเร็กซ์ทรีดี เรียกว่า Transformation Engine ซึ่งมีขั้นตอนการคำนวณเพื่อแปลงพิกัดต่างๆ ดังนี้

- World Transformation เป็นการคำนวณเพื่อแปลงพิกัด (Coordinate) หรือตำแหน่งของเวอร์เท็กซ์ (Vertex) ต่างๆ ที่ประกอบขึ้นเป็นโมเดลที่อยู่ในโมเดลสเปซ (Model Space) ให้เป็นพิกัดในเวิร์ลสเปซ (World Space) กล่าวได้ว่า เป็นการแปลงตำแหน่งเวอร์เท็กซ์เดิมที่มีความสัมพันธ์อยู่ในท้องถิ่นตนเอง (Local Coordinate) ให้มีความสัมพันธ์อ้างอิงกับตำแหน่งเวอร์เท็กซ์อื่นๆ ที่อยู่ในโลก 3 มิติ ด้วยกัน



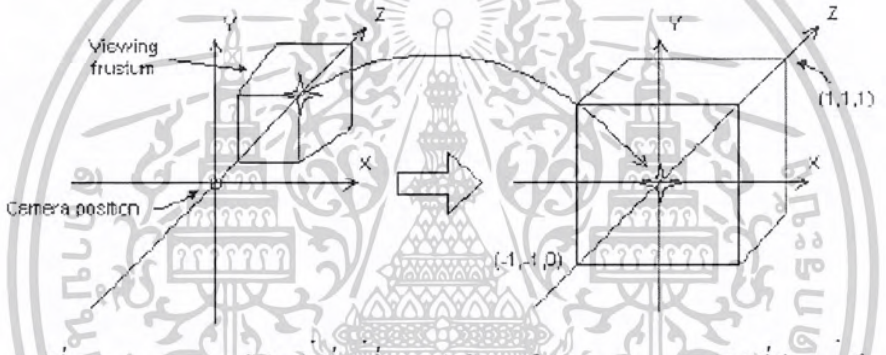
รูปที่ 2-5 แสดงความสัมพันธ์ระหว่างโมเดลในโมเดลสเปซที่มาอ้างอิงตำแหน่งใหม่ในเวิร์ลสเปซ

- View Transformation เป็นการคำนวณเพื่อแปลง ตำแหน่งของเวอร์เท็กซ์ต่างๆ บางส่วนของเวิร์ลสเปซ (World Space) ให้ไปเป็นพิกัด ในคาเมร่าสเปซ (Camera Space) ซึ่งถือได้ว่าเป็นการแปลงให้มาอยู่ในมุมมองของผู้ดู



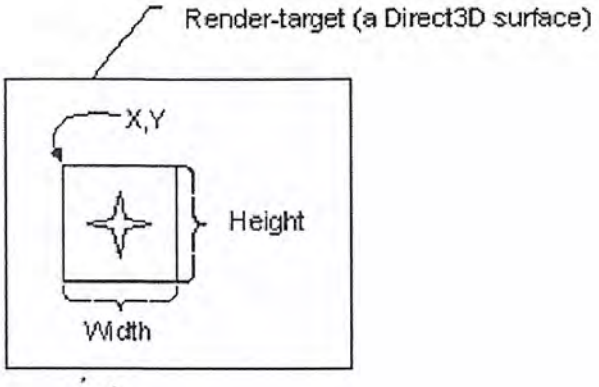
รูปที่ 2-6 แสดงความสัมพันธ์ระหว่างโมเดลในเวิร์ลสเปซกับมุมมองของกล้อง

- Lighting เป็นการคำนวณค่าสีต่างๆ ให้กับเวอร์เท็กซ์ เช่น สีที่ได้จากเท็กเซล (Texel) ของเท็กซ์เจอร์ (Texture) ที่นำมาลงให้ตรง, สีของจุดเวอร์เท็กซ์นั้นๆ หรือสีจากแหล่งกำเนิดแสงต่างๆ เป็นต้น
- Projection Transformation เป็นการคำนวณเพื่อแปลงสิ่งที่อยู่ในคาเมร่าสเปซ (Camera Space) ให้ไปอยู่ในโปรเจ็กชั่นสเปซ (Projection Space) โดยที่ขั้นตอนนี้จะมีการคำนวณตำแหน่งและขนาดของโมเดลให้ผิดแปลกไปจากเดิม เพื่อที่ว่าเมื่อถูกแปลงให้เห็นเป็นภาพ 2 มิติแล้ว จะได้เห็นเป็นแบบเพอร์สเปคทีฟ (Perspective) คือสิ่งที่อยู่ใกล้ตา จะใหญ่กว่าสิ่งที่อยู่ไกลออกไป ซึ่งจะทำให้มองเห็นเสมือนว่า ภาพ 2 มิตินั้นดูมีความลึก ซึ่งการเขียนโปรแกรมเพื่อทำงานในส่วน Transformation ทั้ง 3 แบบที่กล่าวมานี้จะต้องใช้เมทริกซ์ (Matrix)



รูปที่ 2-7 แสดงการทำโปรเจ็กชั่น ซึ่งจะเห็นโมเดลในมุมมองภาพ 2 มิติที่มีความลึก

- Clipping and Viewport Scaling เป็นการทำงานที่แทรกก่อนจะไปถึงการทำงานในส่วนของการลงสี (Rasterization) นั่นคือ การตัดส่วนของโมเดลที่ถูกโมเดลอื่นบัง หรือตัดส่วนที่อยู่ด้านหลังออกไป แล้วก็ลดหรือขยายขนาดโมเดลต่างๆ ให้พอดีกับขนาดของความละเอียด (Resoluion) ของหน้าจอที่ใช้ เป็นการช่วยลดการทำงานของขั้นตอนการลงสี คือไม่ต้องไปสนใจสิ่งที่มองไม่เห็นนั่นเอง



รูปที่ 2-8 หลังจากถูกวิวพอร์ตและขริบออก

2.1.6.2 กระบวนการลงสี (Rasterization)

ขั้นสุดท้ายในกระบวนการเรนเดอร์ไปป์ไลน์ จะเป็นการลงสี (Rasterization) โดยตัวที่ทำหน้าที่นี้จะเรียกว่า Rasterizer ซึ่งจะแปลงสิ่งที่อยู่ในโปรเจ็กชั่นสเปซ (Projection Space) ให้ไปอยู่ในสกรีนสเปซ (Screen Space) คือแปลงพิกัดของเท็กเซล (Texel) ต่างๆ ที่เป็นแบบ 3 มิติ ให้เป็น 2 มิติ เมื่อเทียบกับขนาดของจอภาพ หลังจากนี้เราก็จะได้ภาพปรากฏอยู่ในแบ็คบัฟเฟอร์ (Back Buffer) เพื่อที่เราจะได้สั่งให้ไดเร็กทรีดีดีไวซ์ออบเจ็กท์ (Direct3D Device Object : DDI) จัดการนำข้อมูลในแบ็คบัฟเฟอร์ (Back Buffer) นี้ไปสลับหรือก๊อปปี้เป็นฟร็อนท์บัฟเฟอร์ (Front Buffer) เพื่อให้ภาพไปปรากฏให้เห็นบนจอภาพต่อไป

2.2 DirectX คืออะไร

DirectX ตามความหมายจะหมายถึง ไลบรารีคำสั่ง (Run Time Library) ที่ช่วยทำงานด้านมัลติมีเดียกราฟิกโดยตัว DirectX Foundation จะมีส่วนประกอบที่เรียกว่า HAL (Hardware Abstraction Layer) จะใช้ซอฟต์แวร์ในการตรวจสอบความสามารถของฮาร์ดแวร์ ที่อยู่ในเครื่องคอมพิวเตอร์นั้นอย่างอัตโนมัติ แล้วนำมากำหนดพารามิเตอร์ของแอพพลิเคชัน ให้ตรงตามความเหมาะสมระหว่างเกมส์คอมพิวเตอร์ หรือโปรแกรมมัลติมีเดีย กับ ไดรเวอร์ของฮาร์ดแวร์ ที่มีส่วนเกี่ยวข้องกับเกมส์หรือโปรแกรมนั้น ทำให้การประมวลผลโปรแกรมทำงานได้เร็วขึ้น เนื่องจากขั้นตอนต่างๆ จะถูกนำไปประมวลผลโดยตรง ไม่ต้องอาศัยตัวกลางอย่างเช่น GDI (Graphic Device Interface) ทำให้นักพัฒนาเกมส์คอมพิวเตอร์สามารเขียนโปรแกรมให้สื่อสารกับ DirectX เท่านั้นก็เพียงพอ นอกจากนี้ DirectX Foundation ยังมีส่วนประกอบที่เรียกว่า HEL (Hardware Emulation Layer) ทำให้สามารถใช้โปรแกรมมัลติมีเดีย หรือโปรแกรมที่เกี่ยวข้องกับ 3 มิติ บนฮาร์ดแวร์ที่ไม่สนับสนุนใช้งานด้าน 3 มิติ โดยจะทำการจำลองความสามารถบางอย่างที่ฮาร์ดแวร์ตัวนั้นไม่มี ให้สามารถใช้งานได้กับโปแกรมที่ต้องการได้ แม้จะมีข้อเสียอยู่บ้างตรงที่อาจทำให้การทำงานช้าลงบ้างก็ตามแต่ก็คุ้มค่ากับความสามารถของ DirectX ที่มีอยู่ในปัจจุบัน

2.2.1 DirectX SDK

ก่อนอื่นมาทำความเข้าใจกับ 2 ส่วนประกอบสำคัญ ดังนี้

- DirectX Runtime Library เป็นส่วนประกอบของ DirectX ที่จะทำให้สามารถเล่นเกมส์ได้ ซึ่งจะถูกติดตั้งลงในระบบปฏิบัติการวินโดวส์โดยอัตโนมัติ
- DirectX SDK (Software Development Kit) เป็นชุดคำสั่งที่ออกแบบมาใช้ร่วมกับตัวแปลภาษาหลายๆ ภาษา เช่น Visual C++ หรือ Visual Basic ซึ่งสามารถพัฒนาเกมส์คอมพิวเตอร์หรือโปรแกรมมัลติมีเดียเองได้โดยการศึกษาวิธีการเขียนและหน้าที่ของคำสั่งต่างๆ

2.2.2 ส่วนประกอบของ DirectX 8.0 SDK

DirectX 8.0 SDK ถูกพัฒนาให้มีความสามารถเพิ่มขึ้นจากเวอร์ชันก่อนๆ โดยสามารถแบ่งออกเป็นส่วนประกอบหลักๆ ได้ดังนี้

- Direct3D ใช้ในการสร้างและจัดการเกี่ยวกับภาพ 3 มิติ โดยมีจุดเด่นตรงที่การประมวลผลกราฟิกแบบขนาน และสามารถจัดการกับกราฟิกให้มีความสมจริงกว่าเวอร์ชันที่ผ่านๆ มา ทำให้ผู้พัฒนาเกมส์สามารถพัฒนาเกมส์ 3 มิติ ได้ง่ายขึ้น ซึ่งภายในโครงงานได้ใช้ส่วนนี้ในการสร้างโลกเสมือน 3 มิติ
- DirectDraw ใช้ในการสร้างและจัดการเกี่ยวภาพ 2 มิติ
- DirectInput ทำหน้าที่คอยสนับสนุนอุปกรณ์ควบคุม เช่น จอยสติก และอุปกรณ์ควบคุมเกมส์อื่นๆ ทำให้ผู้พัฒนาเกมส์สามารถใช้ประโยชน์ได้ง่ายขึ้น
- DirectPlay ช่วยให้สามารถที่จะเล่นเกมส์ได้ครั้งละหลายๆ คนผ่านระบบเครือข่าย
- DirectSound ใช้สำหรับการจัดการเกี่ยวเสียงและ effect ของเสียง เช่นการทำเสียงรอบทิศทาง
- Audio Video Playback ใช้สำหรับการจัดการเกี่ยวกับเพลงและการเล่นไฟลล์วีดิโอ

2.3 แนะนำ Direct3D

Direct3D ได้รับการออกแบบ เพื่อช่วยในการสร้างเกมส์ที่แสดงภาพแบบ 2 และ 3 มิติ บนเครื่องคอมพิวเตอร์ภายใต้ระบบปฏิบัติการวินโดวส์ ซึ่งกล่าวได้ว่า Direct3D คือซอฟท์แวร์อินเตอร์เฟสซึ่งจัดการการแสดงผลที่รองรับการเข้าถึงอุปกรณ์แสดงผล โดยยังคงรักษาความเข้ากันได้กับกราฟฟิกดีไวซ์อินเตอร์เฟส (Graphics Device Interface : GDI) และจัดการให้มีการใช้งานในแบบที่ไม่ขึ้นกับชนิดของอุปกรณ์ (Device Independent) และมีความสามารถในการใช้คุณสมบัติพิเศษที่อยู่ในฮาร์ดแวร์ได้

Direct3D นี้เป็นแอพพลิเคชั่นโปรแกรมมิ่งอินเทอร์เฟสระดับล่าง ซึ่งเป็นเครื่องมือสำหรับโปรแกรมเมอร์ที่ต้องการสร้างเกมส์ส์หรือโปรแกรมมัลติมีเดียที่แสดงผล 3 มิติ ที่มีประสิทธิภาพสูง บน

ระบบปฏิบัติการวินโดว์ ซึ่งการใช้ API Direct3D นี้ ทำให้โปรแกรมสามารถติดต่อโดยตรงกับฮาร์ดแวร์ เร่งความเร็วในระดับล่าง จึงมีความยืดหยุ่นในการใช้งานสูง

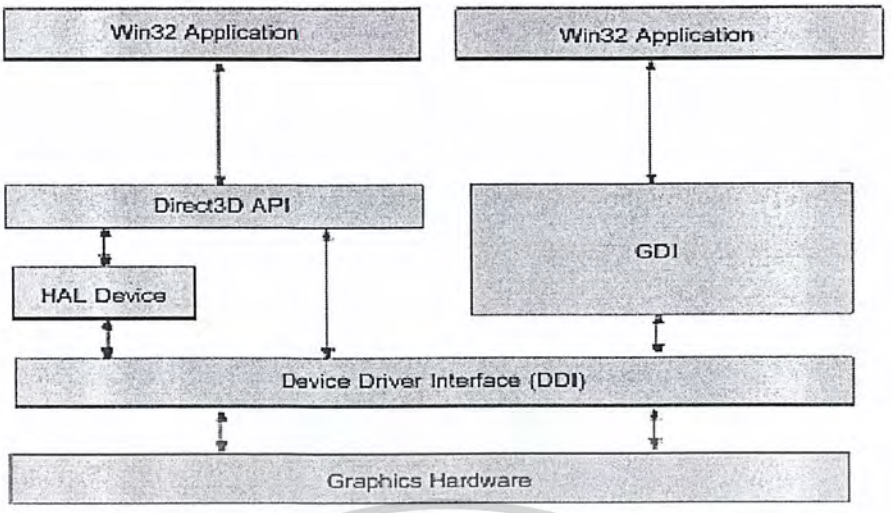
2.3.1 คุณสมบัติ Direct3D

1. สนับสนุนการใช้งาน Depth Buffer (โดยใช้ z-buffer หรือ w-buffer)
2. สามารถเรนเดอร์ภาพได้ทั้งในแบบ Flat Shading และ Gouraud Shading
3. สามารถสร้างแหล่งกำหนดแสงได้หลายชนิดและหลายแหล่ง
4. รองรับการใช้วัสดุแมตทีเรียล (Material) และพื้นผิวเท็กซ์เจอร์ (Texture) รวมไปถึงการทำมิพแมพ (Mipmap)
5. สนับสนุนซอฟต์แวร์จำลองการใช้งานของฮาร์ดแวร์ (Software Emulation Driver) ที่มีประสิทธิภาพสูง
6. มีการทำงานโดยไม่ขึ้นกับชนิดของฮาร์ดแวร์
7. รองรับการแปลงพิกัดวัตถุ (Transformation) และการขริบภาพ (Clipping)
8. สนับสนุนคำสั่งพิเศษที่มีอยู่ในซีพียู สามารถรองรับสถาปัตยกรรมแบบ Intel MMX, Intel Streaming Single-Instruction, Multiple-Data (SIMD) Extensions (SSE) และสถาปัตยกรรมทรีดีของ AMD (AMD's 3D Architecture)
9. สนับสนุน Hardware Abstraction Layer (HAL)
10. สนับสนุนการทำ Page flipping ด้วย Back buffer จำนวนมากในโปรแกรมที่แสดงผลแบบเต็มจอภาพ (Fullscreen)
11. สนับสนุนการตัด (Clipping) ในโปรแกรมประยุกต์ที่ทำงานในวินโดว์โหมด และแบบโหมดเต็มจอภาพ
12. รองรับการทำ 3D - z buffer
13. สามารถใช้งาน Image-stretching Hardware ได้
14. สามารถเข้าถึงหน่วยความจำของอุปกรณ์แสดงผล (Display-Device Memory) แบบStandard และ Enhance ได้ในขณะเดียวกัน

นอกจากนี้ยังมีคุณสมบัติอื่นๆ เช่น การใช้งานฮาร์ดแวร์แต่เพียงผู้เดียว (Exclusive Hardware Access) และการเปลี่ยนความละเอียดในการแสดงผล (Resolution Switching)

ด้วยคุณสมบัติข้างต้นเหล่านี้ ทำให้การพัฒนาโปรแกรมด้วย Direct3D มีประสิทธิภาพสูงกว่าการใช้ กราฟฟิกดีไวซ์อินเตอร์เฟส (Graphics Device Interface : GDI) ของวินโดว์และการเขียนโปรแกรมบนระบบปฏิบัติการดอส

ตามแผนผังข้างล่างนี้ แสดงถึงความสัมพันธ์ระหว่างอินเตอร์เฟส Direct3D, กราฟฟิกดีไวซ์อินเตอร์เฟส, ฮาร์ดแวร์แอ็บสแตร็กชั่นเลเยอร์ (Hardware Abstraction Layer : HAL) และ ฮาร์ดแวร์



รูปที่ 2-9 แสดงความสัมพันธ์ระหว่าง Direct3D กับระบบคอมพิวเตอร์

เมื่อนำมาเปรียบเทียบระหว่าง Direct3D กับ กราฟฟิกดีไวซ์อินเตอร์เฟส (GDI) ทั้งคู่จะมีการเข้าถึงกราฟฟิกฮาร์ดแวร์ (Graphics Hardware) ผ่านดีไวซ์ไดรเวอร์ (Device Driver Interface : DDI) สิ่งที่ต่างกันไป คือ Direct3D มีข้อดีกว่าของหน้าที่การทำงานกับฮาร์ดแวร์เมื่อมีการเลือกใช้ HAL ซึ่งดีไวซ์ HAL นี้จัดการความเร็วทางด้านฮาร์ดแวร์ บนพื้นฐานของชุดการทำงานที่สนับสนุนกราฟฟิกดีไวซ์

HAL นี้จัดการความเร็วทางด้านฮาร์ดแวร์ บนพื้นฐานของชุดการทำงานที่สนับสนุนกราฟฟิกดีไวซ์ไลน์ก็ตาม Direct3D จะใช้ซอฟต์แวร์ที่มีประสิทธิภาพสูงในการทำงานในส่วนนั้นแทน แต่การทำงานของซอฟต์แวร์ ก็ยังมีประสิทธิภาพต่ำกว่าใช้ฮาร์ดแวร์

2.4 ทฤษฎีฟิสิกส์

ในการสร้างระบบจำลองทางฟิสิกส์นั้นจำเป็นจะต้องทราบถึงพื้นฐานของคุณสมบัติทาง ฟิสิกส์ที่ใช้ ในหัวข้อนี้จะแบ่งออกเป็นสองส่วน ส่วนแรกเป็นการแสดงถึงการเคลื่อนไหว(Motion)ของวัตถุแข็ง (Rigid Body) ซึ่งในส่วนนี้จะไม่มีการทำการตรวจสอบการชน(Collisions Detection) นั่นก็คือการจำลองที่ยังไม่มีการชนเกิดขึ้นระหว่างวัตถุแข็ง ซึ่งเป็นส่วนที่แสดงถึงการเคลื่อนไหวและการเกิดการกระทำจากแรงภายนอกมายังวัตถุแข็งและส่งผลจากการชน (Response) กับการเปลี่ยนแปลงการเคลื่อนไหวอย่างไร

ในส่วนที่สองจะกล่าวถึงการทำการตรวจสอบการชน และผลจากการชน ซึ่งจะไม่ยอมให้วัตถุแข็งที่ชนกันสามารถทะลุผ่าน (Inter-penetration) ไปได้ จะต้องมีการบังคับโดยการคำนวณหาค่าที่เหมาะสมที่เกิดขึ้นการกระทบ (Contact) กันระหว่างวัตถุแข็ง ถ้ามีการคำนวณแรงกระทบได้อย่างถูกต้องก็จะทำให้ได้ผลลัพธ์ในการเคลื่อนที่ที่สมบูรณ์แบบที่สุดได้

2.4.1 พื้นฐานการจำลอง (Simulation Basics)

ในส่วนนี้จะกล่าวถึงโครงสร้างพื้นฐานสำหรับการจำลองการเคลื่อนไหวของวัตถุแข็งว่าจะต้องมีองค์ประกอบอะไรบ้างและมีแนวคิด รวมถึง สมการทางคณิตศาสตร์มาช่วยในการอธิบายร่วมด้วย.

เริ่มต้นกำหนดฟังก์ชั่น $x(t)$ คือ ตำแหน่งของวัตถุที่อยู่ใน World space ที่ช่วงเวลา t ใดๆ และฟังก์ชั่น $v(t) = x(t) = \frac{d(x(t))}{dt}$ ให้เป็นอัตราความเร็ว(Velocity) หรือกล่าวอีกในหนึ่งก็คือ อัตราความเร็ว ที่เวลา t มีค่าเท่ากับการเคลื่อนที่ (Translate) ที่เปลี่ยนแปลงไปเมื่อเทียบกับเวลา t ใดๆ สามารถนำแนวความคิดนี้มานิยามเป็น State vector Y(t) ดังนี้

$$Y(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix}$$

ในความเป็นจริงการจำลองการเคลื่อนไหวนั้นต้องทราบถึงแรง (Force) ที่กระทำกับวัตถุแข็งที่ช่วงเวลาใดๆ (t) ด้วยดังนั้นจะมีฟังก์ชั่น F(t) คือแรงที่กระทำกับวัตถุ ณ ช่วงเวลา t ใดๆ ซึ่งเป็นผลรวมของแรงทั้งหมดที่กระทำกับวัตถุ เช่นแรงโน้มถ่วง (Gravity), แรงลม (Wind), แรงดีด (Spring) เป็นต้น ถ้าวัตถุมีมวล (Mass) m ดังนั้นการเปลี่ยนแปลงของ Y ในช่วงเวลามี่เปลี่ยนไปจะได้

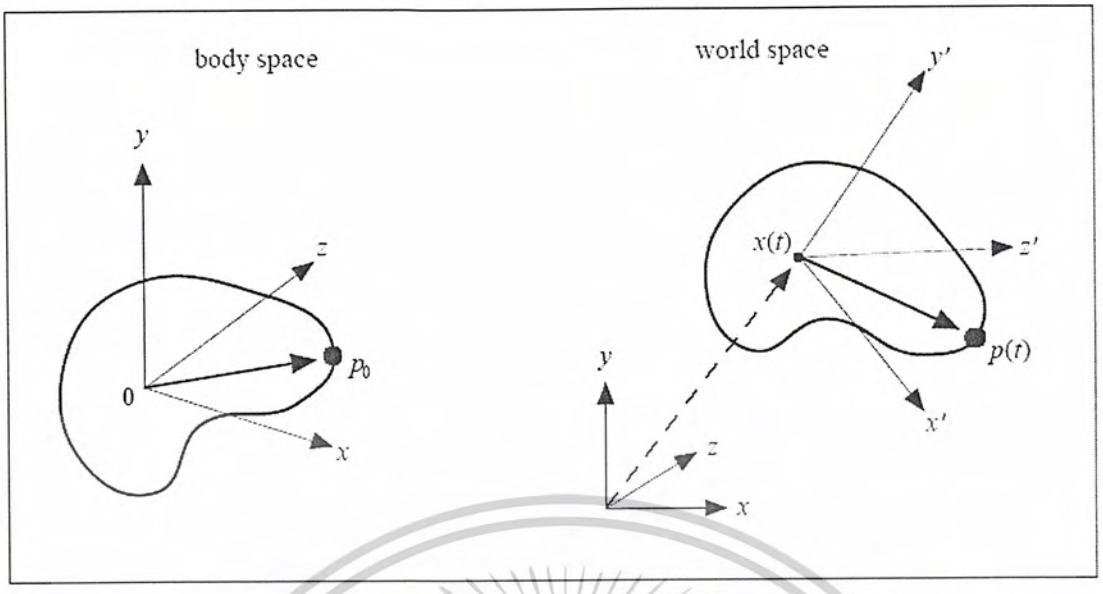
$$\frac{d}{dt}Y(t) = \frac{d}{dt}\begin{pmatrix} x(t) \\ v(t) \end{pmatrix} = \begin{pmatrix} v(t) \\ F(t)/m \end{pmatrix}$$

2.4.2 แนวคิดของวัตถุแข็ง (Rigid Body Concepts)

ในโครงงานนี้จะทำการจำลองการปฏิสัมพันธ์กับวัตถุแข็ง ซึ่งรายละเอียดของคุณสมบัติวัตถุแข็งมีดังต่อไปนี้

2.4.2.1 ตำแหน่งและการหันเห (Position and Orientation)

ตำแหน่งที่ตั้งของวัตถุใน Space ช่วงเวลา t ใดๆ สามารถอธิบายได้ด้วย vector x(t) ซึ่งถูกทำให้เคลื่อนที่มาจากจุดกำเนิด (Origin) ใช้เวคเตอร์ x(t) อธิบายการเคลื่อนที่ของวัตถุแข็ง และ R(t) ใช้ในการอธิบายการหมุนของวัตถุแข็งซึ่งจะอยู่ในรูปของเมตริกซ์ ขนาด 3x3 โดยที่รูป 2-10 จะแสดงถึงจุดศูนย์กลางของวัตถุแข็งถูกเคลื่อนที่ไปยังจุด x(t) ใน World space ที่ช่วงเวลา t ใดๆ ตำแหน่งของ x,y,z ของวัตถุใน Body Space จะถูกเคลื่อนย้ายไปยังเวคเตอร์ $x' = R(t)x, y' = R(t)y, z' = R(t)z$ จุด p_0 ใน Body Space ถูกเคลื่อนที่ ไปยังจุด $p(t) = R(t)p_0 + x(t)$



รูปที่ 2-10 จุดศูนย์กลางของวัตถุแข็งถูกเคลื่อนที่ไปยังจุด x(t) ใน world space ที่ช่วงเวลา t ใดๆ

2.4.2.2 Linear Velocity

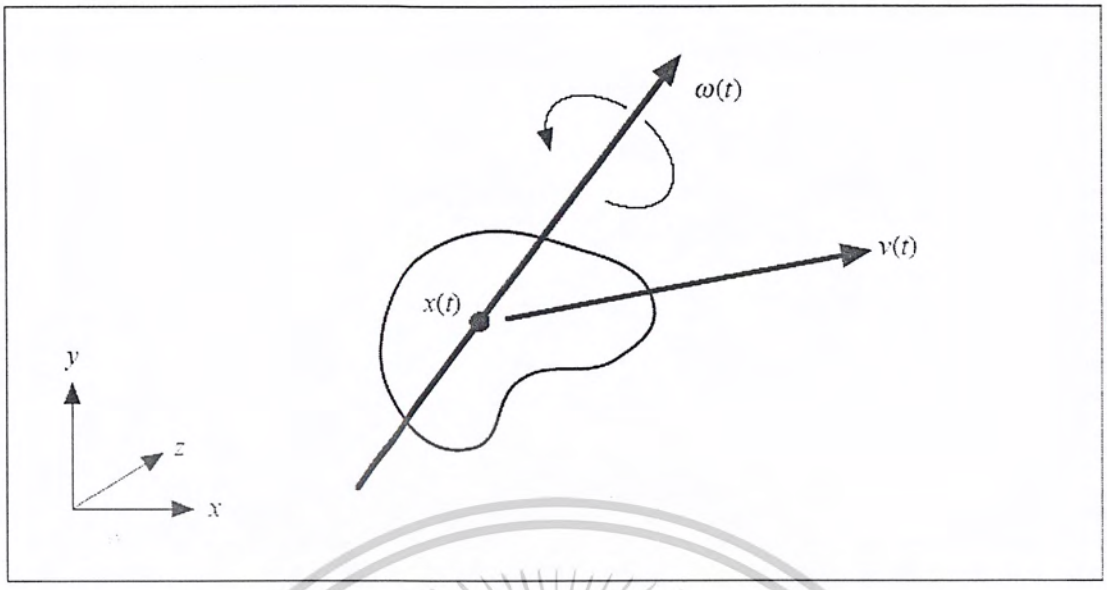
เนื่องจาก x(t) คือตำแหน่งของจุดกึ่งกลางของวัตถุแข็งใน World Space ดังนั้นการเคลื่อนที่ของจุดกึ่งกลางของวัตถุจากจุดกำเนิดเคลื่อนที่ไปยังตำแหน่ง $\dot{x}(t)$ ที่ช่วงเวลา t ใดๆ ก็คือ Linear Velocity $v(t)$ นั่นคือ

$$v(t) = \dot{x}(t)$$

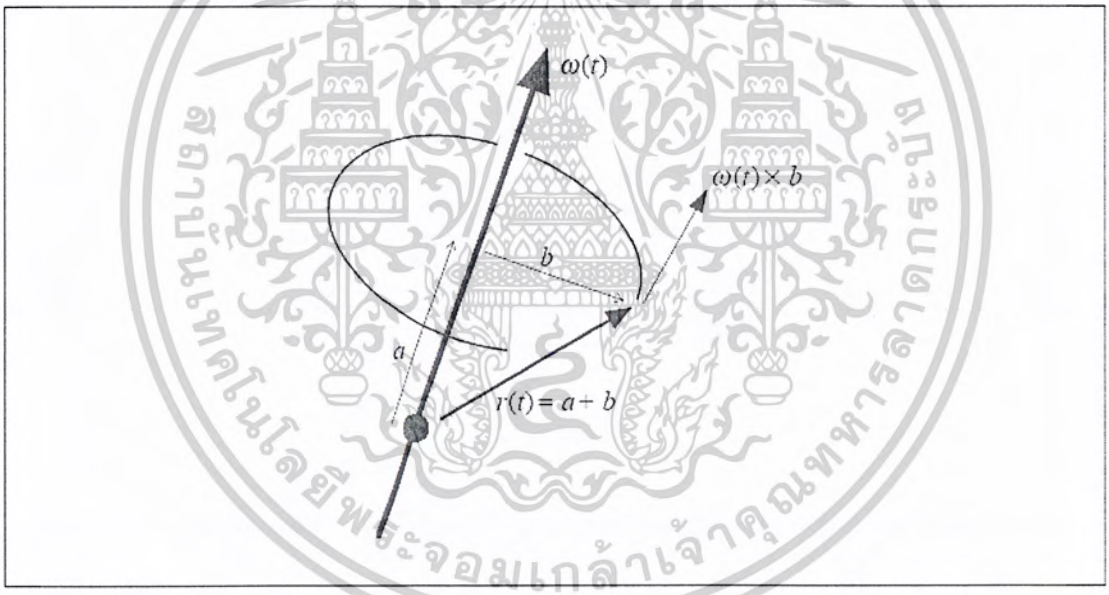
2.4.2.3 Angular Velocity

ในการเคลื่อนที่ของวัตถุซึ่งวัตถุแข็งนั้นสามารถหมุนรอบตัวเอง พร้อมๆกับการเคลื่อนที่ไปใน World space โดยกำหนดเวคเตอร์ในการหมุน นี้คือเวคเตอร์ $\omega(t)$ ซึ่งเป็นแกนของการหมุนของวัตถุ ขนาดของ $\omega(t), |\omega(t)|$ จะเป็นตัวบอกความเร็วในการหมุนของวัตถุแข็ง

โดยในรูปที่ 2-11 แสดงถึงการรวมของ Linear Velocity และ Angular Velocity ที่เกิดขึ้นในวัตถุพร้อมๆ กัน



รูปที่ 2-11 Linear velocity v(t) and angular velocity $\omega(t)$ ของ Rigid body



รูปที่ 2-12 แสดงอัตราการเปลี่ยนแปลงของการหมุน r(t) หมุนรอบแกน $\omega(t)$ ความเร็วของการหมุนของ r(t) คือ $|\omega(t)||b|$

เนื่องจาก r(t) = a + b และเวคเตอร์ a ขนานกัน $\omega(t)$ จะได้ $\omega(t)$ cross a = 0 ดังนั้น

$$\dot{r}(t) = \omega(t) \times b = \omega(t) \times b + \omega(t) \times a = \omega(t) \times (b + a)$$

ดังนั้นจึงสามารถเขียนสมการใหม่ได้ว่า

$$\dot{r}(t) = \omega(t) \times r(t)$$

สามารถเขียนสมการของการหมุนเมื่อเทียบกับเวลาได้ว่า

$$\dot{R} = \begin{pmatrix} \omega(t) \times \begin{pmatrix} r_{xx} \\ r_{xy} \\ r_{xz} \end{pmatrix} & \omega(t) \times \begin{pmatrix} r_{yx} \\ r_{yy} \\ r_{yz} \end{pmatrix} & \omega(t) \times \begin{pmatrix} r_{zx} \\ r_{zy} \\ r_{zz} \end{pmatrix} \end{pmatrix}$$

ถ้าจะทำการ cross product ระหว่าง vector a กับ vector b

$$\begin{pmatrix} a_y b_z - b_y a_z \\ -a_x b_z - b_x a_z \\ a_x b_y - b_x a_y \end{pmatrix}$$

ซึ่งจะมีค่าเท่ากับการเอา matrix a* คูณเข้าไป โดยที่ matrix a* มีค่าดังนี้

$$\begin{pmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{pmatrix}$$

ดังนั้น

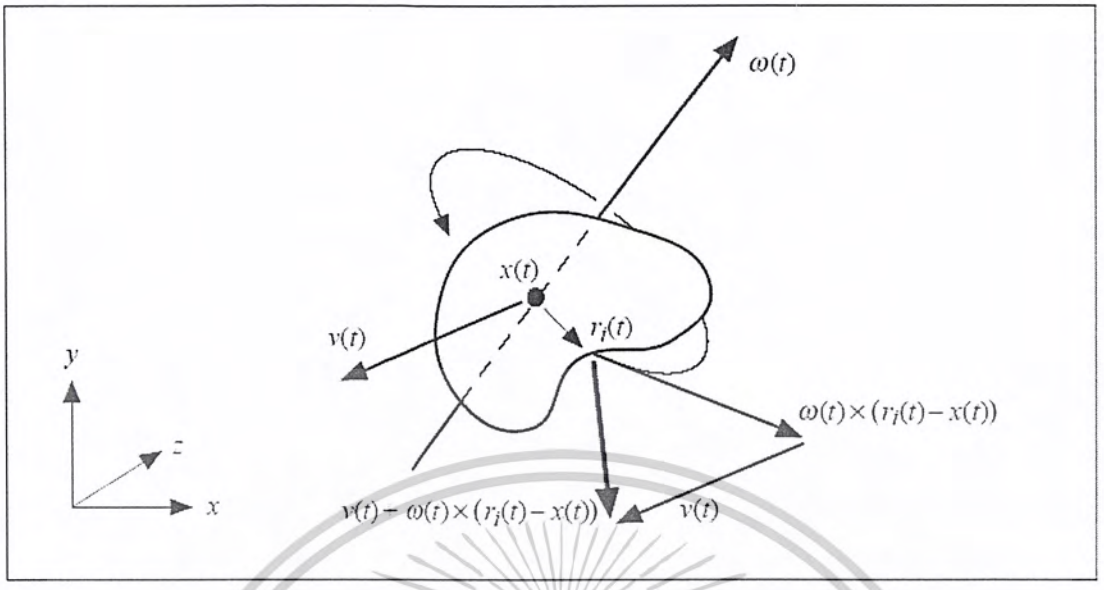
$$a^* b = \begin{pmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{pmatrix} \begin{pmatrix} b_x \\ b_y \\ b_z \end{pmatrix} = \begin{pmatrix} a_y b_z - b_y a_z \\ -a_x b_z + b_x a_z \\ a_x b_y - b_x a_y \end{pmatrix} = a \times b$$

ใช้ " * " เข้ามาเขียนสมการใหม่จะได้ว่า

$$\dot{R}(t) = \omega(t)^* \begin{pmatrix} \begin{pmatrix} r_{xx} \\ r_{xy} \\ r_{xz} \end{pmatrix} & \begin{pmatrix} r_{yx} \\ r_{yy} \\ r_{yz} \end{pmatrix} & \begin{pmatrix} r_{zx} \\ r_{zy} \\ r_{zz} \end{pmatrix} \end{pmatrix}$$

$$\dot{R}(t) = \omega(t)^* R(t)$$

ซึ่งจะได้สมการที่เหมือนกันระหว่าง $\dot{r}(t) = \omega(t) \times r(t)$ กับเมทริกซ์ $\dot{R}(t) = \omega(t)^* R(t)$

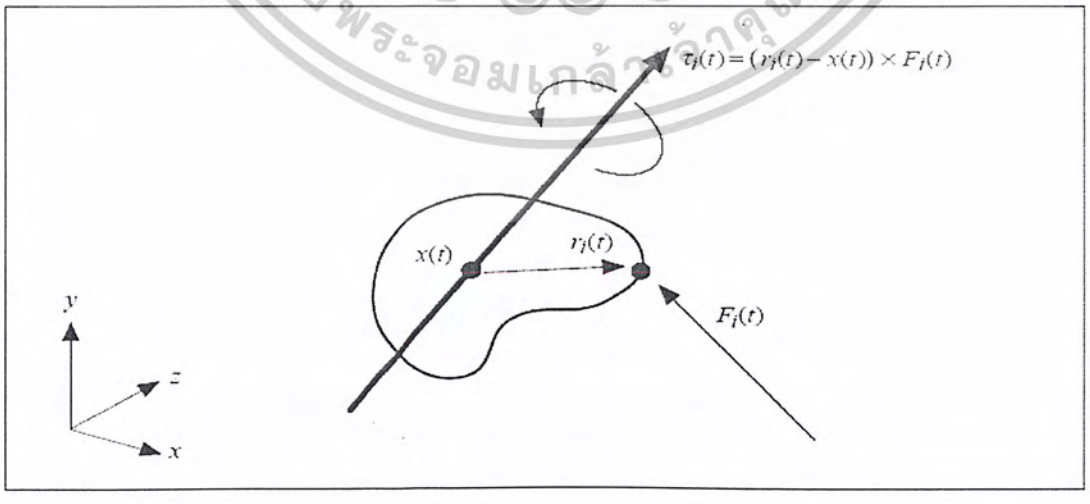


รูปที่ 2-13 Velocity ของจุดใดๆบนวัตถุแข็งใน World Space ซึ่งอัตราความเร็ว ของ $r_i(t)$ สามารถแตกออกเป็นสองส่วนคือ linear term $v(t)$ และ angular term $\omega(t)\times(r_i(t)-x(t))$

2.4.2.4 Force and Torque

เมื่อมีแรงมากระทำบนวัตถุแข็งย่อมส่งผลกระทบโดยกำหนดแรงรวม $F_i(t)$ แรงทั้งหมดจากภายนอกที่กระทำกับวัตถุแข็งที่ช่วงเวลา t ใดๆ และกำหนด Torque $\tau_i(t)$ ซึ่งกระทำบนจุดใดจุดหนึ่งบนวัตถุจะได้ว่า

$$\tau_i(t)=(r_i(t)-x(t))\times F_i(t)$$



รูปที่ 2-14 แสดงค่า Torque $\tau_i(t)$ ที่รับกับค่าแรง $F_i(t)$ ที่กระทำกับวัตถุที่จุด $r_i(t)$

แรงรวมจากภายนอก $F(t)$ที่กระทำบนวัตถุคือผลรวมของ $F_i(t)$:

$$F(t)=\sum F_i(t)$$

ในขณะที่ค่า Torque จากภายนอกทั้งหมดกำหนดได้คล้ายๆ กันคือ

$$\tau(t)=\sum \tau_i(t)=\sum (r_i(t)-x(t))\times F_i(t)$$

2.4.2.5 Linear Momentum

กำหนดให้ p คือ Linear Momentum ของวัตถุโดย มวลของวัตถุคือ m และอัตราความเร็วคือ v กำหนดได้ว่า

$$p(t)=mv(t)$$

Linear Momentum ทั้งหมด P(t) ของวัตถุแข็งคือ ผลรวมของผลคูณของมวลกับอัตราความเร็ว

$$P(t)=\sum m_i\dot{r}_i(t)$$

จากสมการ $\dot{r}_i(t)=v(t)+\omega(t)\times(r_i(t)-x(t))$ จะได้ Linear momentum ทั้งหมดของวัตถุแข็ง คือ

$$\begin{aligned}P(t) &= \sum m_i\dot{r}_i(t)\\ &= \sum (m_i v(t)+m_i\omega(t)\times(r_i(t)-x(t)))\\ &= \sum m_i v(t)+\omega(t)\times\sum m_i(r_i(t)-x(t))\end{aligned}$$

2.4.2.6 Angular Momentum

นิยาม Angular Momentum ทั้งหมด $L(t)$ ของวัตถุแข็งได้ด้วยสมการ

$$L(t)=I(t)\omega(t)$$

เมื่อ I(t) คือเมตริกซ์ขนาด 3x3 ที่เรียกว่า inertia tensor ซึ่งรายละเอียดจะได้กล่าวอยู่ในหัวข้อถัดไป ค่า Inertia Tensor นี้จะใช้อธิบายว่ามวลของวัตถุแข็งที่กระจายอยู่มีความสัมพันธ์อย่างไรกับศูนย์กลางของมวลวัตถุแข็ง $L(t)$ เป็นอิสระจากผลของการเคลื่อนที่ใดๆ ในขณะที่ $P(t)$ เป็นอิสระจากผลของการหมุน

2.4.2.7 The Inertia Tensor

The Inertia Tensor $I(t)$ คือ Scaling Factor ระหว่างค่า Angular Momentum $L(t)$ และค่า Angular Velocity $\omega(t)$ ณ ที่ช่วงเวลา t ใดๆ กำหนดให้ r_i' แทนค่าที่ ith ของ x(t) โดยกำหนด $r_i' = r_i(t) - x(t)$, Tensor I(t) ซึ่งอยู่ในรูปของ r_i' ดังนี้

$$I(t) = \sum \begin{pmatrix} m_i\left(r_{iy}'^2 + r_{iz}'^2\right) & -m_i r_{ix}' r_{iy}' & -m_i r_{ix}' r_{iz}' \\ -m_i r_{iy}' r_{ix}' & m_i\left(r_{ix}'^2 + r_{iz}'^2\right) & -m_i r_{iy}' r_{iz}' \\ -m_i r_{iz}' r_{ix}' & -m_i r_{iz}' r_{iy}' & m_i\left(r_{ix}'^2 + r_{iy}'^2\right) \end{pmatrix}$$

แทนค่า $r_i'^T r_i' = r_{ix}'^2 + r_{iy}'^2 + r_{iz}'^2$ แล้วเขียนรูปสมการใหม่ได้ว่า

$$I(t) = \sum m_i r_i'^T r_i' \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} - \begin{pmatrix} m_i r_{ix}'^2 & m_i r_{ix}' r_{iy}' & m_i r_{ix}' r_{iz}' \\ m_i r_{iy}' r_{ix}' & m_i r_{iy}'^2 & m_i r_{iy}' r_{iz}' \\ m_i r_{iz}' r_{ix}' & m_i r_{iz}' r_{iy}' & m_i r_{iz}'^2 \end{pmatrix}$$

โดยที่

$$r_i'\ r_i'^T = \begin{pmatrix} r_{ix}' \\ r_{iy}' \\ r_{iz}' \end{pmatrix} \begin{pmatrix} r_{ix}' & r_{iy}' & r_{iz}' \end{pmatrix} = \begin{pmatrix} r_{ix}'^2 & r_{ix}' r_{iy}' & r_{ix}' r_{iz}' \\ r_{iy}' r_{ix}' & r_{iy}'^2 & r_{iy}' r_{iz}' \\ r_{iz}' r_{ix}' & r_{iz}' r_{iy}' & r_{iz}'^2 \end{pmatrix}$$

กำหนดให้ I มีค่าเท่ากับ

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

นำ I แทน 3 x3 matrix

$$I(t) = \sum m_i \left(\left(r_i'^T r_i' \right) 1 - r_i'\ r_i'^T \right)$$

แต่เนื่องจาก $r_i'(t) = R(t) r_{0i} + x(t)$ และค่า r_i' คือค่าคงที่โดยที่ $r_i' = R(t) r_{0i}$ และค่า $R(t)R(t)^T = 1$

$$\begin{aligned} I(t) &= \sum m_i \left(\left(r_i'^T r_i' \right) 1 - r_i' r_i'^T \right) \\ &= \sum m_i \left(\left((R(t) r_{0i})^T (R(t) r_{0i}) \right) 1 - (R(t) r_{0i})(R(t) r_{0i})^T \right) \\ &= \sum m_i \left(r_{0i}^T R(t)^T R(t) r_{0i} 1 - R(t) r_{0i} r_{0i}^T R(t)^T \right) \\ &= \sum m_i \left(\left(r_{0i}^T r_{0i} \right) 1 - R(t) r_{0i} r_{0i}^T R(t)^T \right) \end{aligned}$$

เนื่องจาก ${r_{0i}}^T r_{0i}$ เป็นค่าคงที่จึงจัดสมการใหม่ได้ว่า

$$\begin{aligned} I(t) &= \sum m_i \left(\left({r_{0i}}^T r_{0i} \right) 1 - R(t) r_{0i} {r_{0i}}^T R(t)^T \right) \\ &= \sum m_i \left(R(t) \left({r_{0i}}^T r_{0i} \right) R(t)^T 1 - R(t) r_{0i} {r_{0i}}^T R(t)^T \right) \\ &= R(t) \left(\sum m_i \left(\left({r_{0i}}^T r_{0i} \right) 1 - r_{0i} {r_{0i}}^T \right) \right) R(t)^T \end{aligned}$$

ถ้ากำหนดให้ I_{body} เป็นเมตริกซ์

$$I_{body} = \sum m_i \left(\left({r_{0i}}^T r_{0i} \right) 1 - r_{0i} {r_{0i}}^T \right)$$

แทน I_{body} ลงในสมการก่อนหน้าจะได้

$$I(t) = R(t) I_{body} R(t)^T$$

เนื่องจาก I_{body} เป็นค่าคงที่ตลอดการจำลอง ดังนั้นจึงทำการคำนวณค่านี้ไว้ก่อนล่วงหน้าได้เลย สามารถคำนวณ $I(t)$ ได้จาก I_{body} และค่าการหันเห $R(t)$

2.4.2.8 Rigid Body Equations of Motion

สุดท้ายเอา สมการทั้งหมดที่ได้มากำหนดเป็น State vector Y(t) สำหรับวัตถุแข็ง กำหนดว่า

$$\mathrm{Y}(t) = \begin{pmatrix} x(t) \\ R(t) \\ P(t) \\ L(t) \end{pmatrix}$$

สถานะของวัตถุแข็ง คือ ตำแหน่งและการหันเหของตัวมันเอง (อธิบายได้โดย Spatial Information) และ Linear กับ Angular Momentum (อธิบายโดย Velocity Information) มวลของวัตถุแข็ง และค่า Inertia tensor I_{body} เป็นค่าคงที่ที่คำนวณไว้แล้วก่อนเริ่มทำการจำลอง ที่ช่วงเวลาหนึ่งค่าที่ช่วยใช้ในการคำนวณ $I(t)$, $\omega(t)$ และ $v(t)$ คำนวณโดย

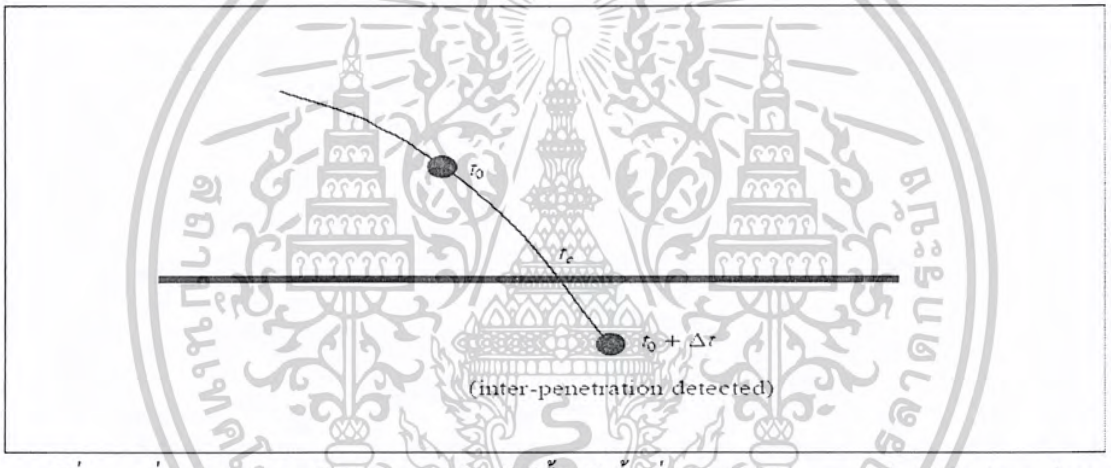
$$v(t) = \frac{P(t)}{M}, \quad I(t) = R(t) I_{body} R(t)^T \text{ และ } \omega(t) = I(t)^{-1} L(t)$$

เขียนใหม่อยู่ในรูปของ $\frac{d}{dt}Y(t)$

$$\frac{d}{dt}Y(t)=\frac{d}{dt}\begin{pmatrix}x(t)\\R(t)\\P(t)\\L(t)\end{pmatrix}=\begin{pmatrix}v(t)\\\omega(t)*R(t)\\F(t)\\\tau(t)\end{pmatrix}$$

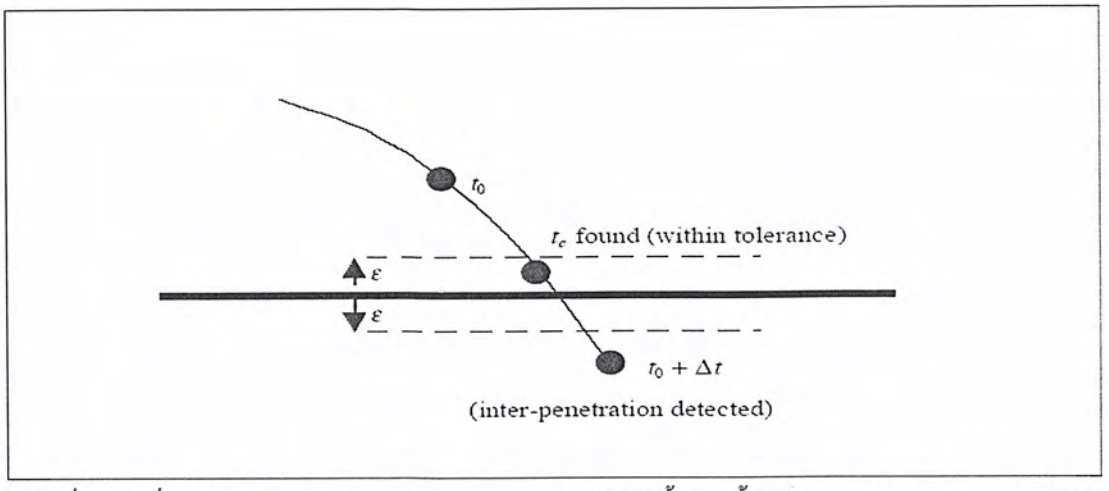
2.5 การตรวจสอบการชน

ขณะที่ทำการจำลอง การคำนวณตำแหน่งของวัตถุที่มันตกลงสู่พื้นในค่าเวลาค่าหนึ่ง(ดังรูปที่ 2-15) พิจารณาวัตถุที่เวลา $t_0, t_0+\Delta t, t_0+2\Delta t$ ไปเรื่อยๆ และสมมติว่าที่เวลาของการชน t_c ที่ซึ่งวัตถุกระทบพื้นจริงๆอยู่ระหว่าง t_0 และ $t_0+\Delta t$ ดังนั้นที่เวลา t_0 จะพบว่าวัตถุอยู่เหนือพื้นแต่ที่ช่วงเวลาต่อไป $t_0+\Delta t$ จะพบว่าวัตถุอยู่ใต้พื้น



รูปที่ 2-15 ที่เวลา $t_0+\Delta t$ วัตถุจะถูกพบว่าอยู่ใต้พื้น ดังนั้นที่เวลาชนจริง t_c จะอยู่ระหว่างตำแหน่ง t_0 กับ $t_0+\Delta t$

ถ้าทำการหยุดและเริ่มใหม่ของการจำลองที่เวลา t_c จำเป็นจะต้องคำนวณ t_c ทั้งหมดนี้รู้ได้ว่า t_c นั้นอยู่ระหว่าง t_0 และ $t_0+\Delta t$ วิธีที่ง่ายของการตัดสินจะใช้ Numerical Method เรียกว่า Bisection คือ ถ้าที่เวลา $t_0+\Delta t$ พบว่าทะลุผ่าน จะบอกให้การคำนวณว่าต้องการให้กลับไปยังที่เวลา t_0 ใหม่ และจำลองการเดินหน้าไปยังเวลา $t_0+\Delta t/2$ ถ้าการจำลองไปถึง $t_0+\Delta t/2$ แล้วไม่เกิดการทะลุผ่าน จะรู้ว่าเวลาที่จะชน t_c จะอยู่ระหว่าง $t_0+\Delta t/2$ กับ $t_0+\Delta t$ แต่ถ้าเกิดการทะลุผ่าน ก็แสดงว่าจะอยู่น้อยกว่า $t_0+\Delta t/2$ แล้วทำการจำลองจาก t_0 ไปยัง $t_0+\Delta t/4$ ความแน่นอนในการพบ t_c ขึ้นอยู่กับ Collision Detection Routines ซึ่งมีบางตัวคือ ε ตัดสินว่าการคำนวณ t_c นั้นดีพอ เมื่อวัตถุทะลุผ่านพื้นโดยไม่เกินค่า ε และอยู่น้อยกว่า ε คือเหนือพื้น ดังรูปที่ 2-16



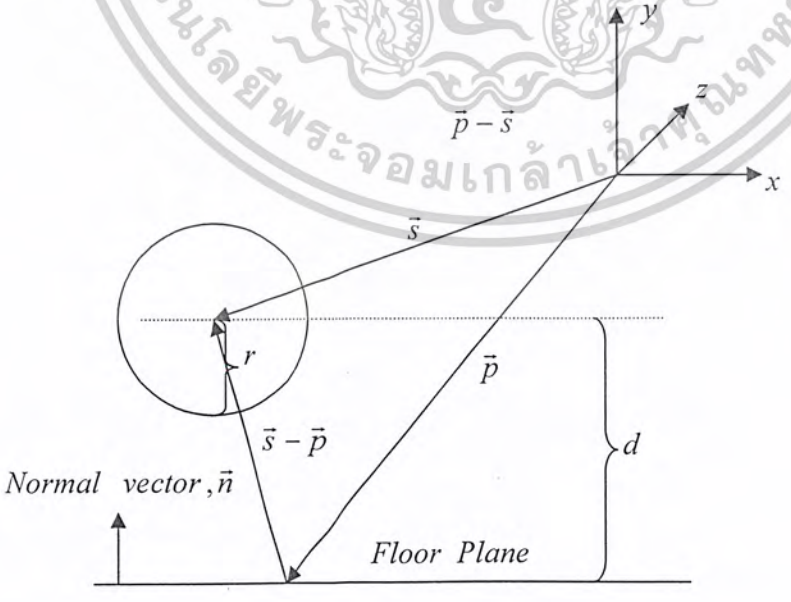
รูปที่ 2-16 เมื่อวัตถุถูกพบว่าอยู่ในช่วงของ ε ของการสัมผัสพื้น ดังนั้นก็จะทำการคำนวณ t_c จะทำให้พิจารณาได้อย่างมีประสิทธิภาพและแม่นยำ

2.5.1 การตรวจสอบการชนในแต่ละกรณี **(Colliding Detection)**

อัลกอริทึ่มตรวจสอบการชนเริ่มต้นด้วยขั้นแรกใส่ bounding box ครอบลงในแต่ละวัตถุแข็ง (กล่องนั้นมีด้านขนานกับแกน) กำหนดให้ n เป็นจำนวน bounding boxes ทั้งหมด จะต้องการที่จะตัดสินใจอย่างรวดเร็วว่าคู่ของ bounding boxes นั้นเหลื่อมกัน คู่ bounding box ของวัตถุแข็งใดที่ไม่เหลื่อมกันก็ไม่จำเป็นต้องนำมาพิจารณา คู่ bounding box ของวัตถุแข็งใดที่เหลื่อมทับกันก็ต้องการนำมาพิจารณา

ในการตรวจสอบการชนนั้นเราได้ใช้วัตถุอยู่สองชนิดคือวัตถุทรงกลมและวัตถุทรงสี่เหลี่ยม ซึ่งเมื่อเกิดการชนจะแบ่งชนิดของการชนออกเป็นกรณีดังต่อไปนี้

- **การชนระหว่างทรงกลมกับระนาบ**



รูปที่ 2-17 แสดงวิธีการทดสอบการชน

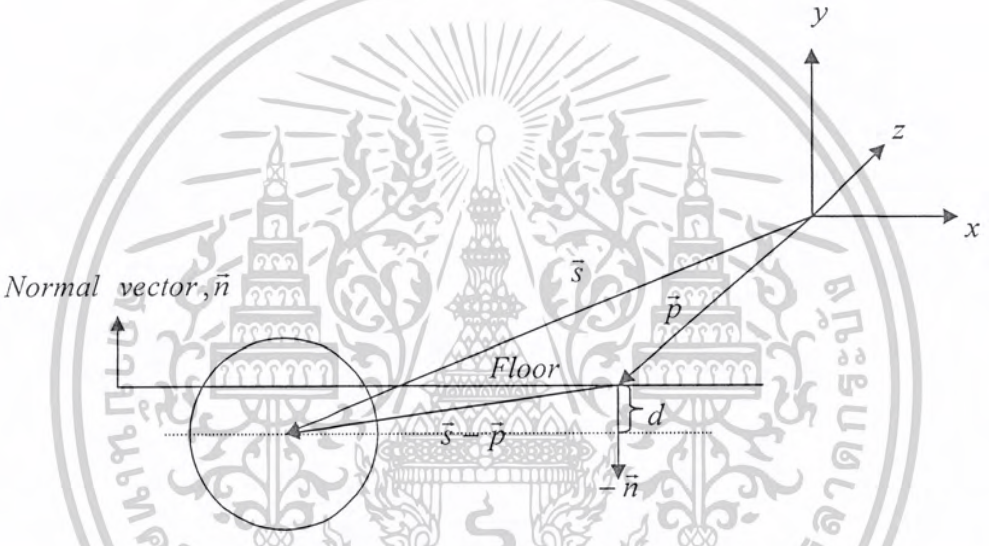
การทดสอบการชนในรูปแบบนี้คือ ขั้นแรกให้เราหาเวคเตอร์ที่ชี้จากระนาบไปยังขอบของทรงกลมโดยหาได้จาก

$$\vec{s} - \vec{p} = (\text{center\_sphere} - \text{radius}) - \text{floor\_position}$$

หลังจากเราได้เวคเตอร์ $\vec{s} - \vec{p}$ แล้วเราก็ทำการ dot product ระหว่างเวคเตอร์ $\vec{s} - \vec{p}$ กับ normal vector ของระนาบจะได้

$$\text{dist} = a * \_\text{vector\_floor}$$

จากนั้นเราก็จะทำการทดสอบว่าถ้าค่าของ dist มีค่ามากกว่า 0 แสดงว่าทรงกลมนั้นยังอยู่เหนือระนาบ แต่ถ้าทรงกลมนั้นสัมผัสกับระนาบหรือทะลุผ่านระนาบไปแล้วค่าของ dist จะมีค่าเท่ากับหรือน้อยกว่า 0

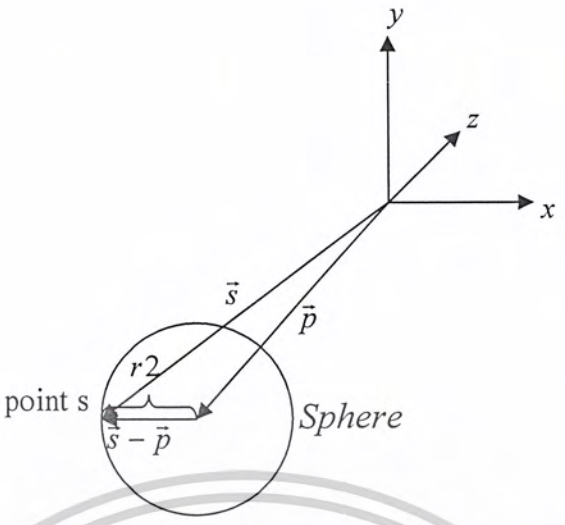


รูปที่ 2-18 แสดงเมื่อทรงกลมสัมผัสกับระนาบ

จากรูปที่ 2-18 เมื่อวัตถุเกิดการชนจะสามารถคำนวณหาระยะ d ได้จากการนำเวคเตอร์ $\vec{S}$ ลบด้วยเวคเตอร์ $\vec{P}$ จะได้เวคเตอร์ที่ชี้จากจุด $\vec{S}$ มายังจุด $\vec{P}$ จากนั้นเราจะหาระยะทางที่โปรเจคชั่นอยู่บนแนวของระนาบ – normal vector ($-\vec{n}$) ก็โดยอาศัยวิธีการ dot product

- การชนระหว่างทรงกลมกับทรงกลม

ในการตรวจสอบการชนในกรณีนี้ เราจะใช้วิธีของการหาระยะห่างระหว่างทรงกลมแล้วตัดค่ารัศมีของทรงกลมทั้งสองออกก็จะทำให้ทราบถึงการสัมผัสของทรงกลมทั้งสอง



รูปที่ 2-19 แสดงการทดสอบการชนระหว่างทรงกลมกับทรงกลม

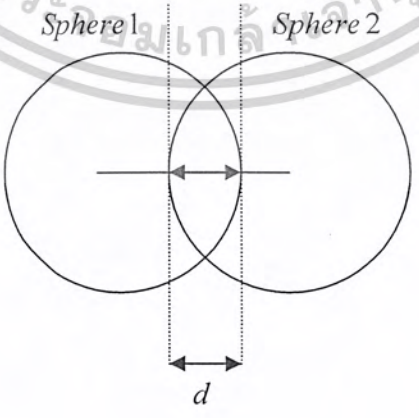
ก่อนอื่นเราจะต้องทำการหาเวคเตอร์ $\vec{s} - \vec{p}$ โดยการนำจุดกึ่งกลางของทรงกลมที่สองลบด้วยจุดกึ่งกลางของทรงกลมที่หนึ่งจะได้เวคเตอร์

$$\vec{s} - \vec{p} \quad = \quad \vec{s} - \vec{p}$$

เมื่อนำเวคเตอร์ $\vec{s} - \vec{p}$ มาหาขนาดโดยการ normalize vector ก็จะทำให้ได้ขนาดของเวคเตอร์ $\vec{s} - \vec{p}$ ออกมาซึ่งนั่นก็คือความยาวของเวคเตอร์ $\vec{s} - \vec{p}$ หลังจากนั้นเราจะต้องคิดโดยคำนึงถึงรัศมีทั้งสองของทรงกลม

$$rr = r1 + r2$$

แล้วทำการเปรียบเทียบค่าของ $\vec{s} - \vec{p}$ ที่ได้กลับรัศมี rr ว่ามีการเหลื่อมกันหรือยังโดยตรวจสอบว่าระยะห่างของทรงกลมทั้งสองมีค่าน้อยกว่ารัศมีทั้งสองอันรวมกันหรือไม่

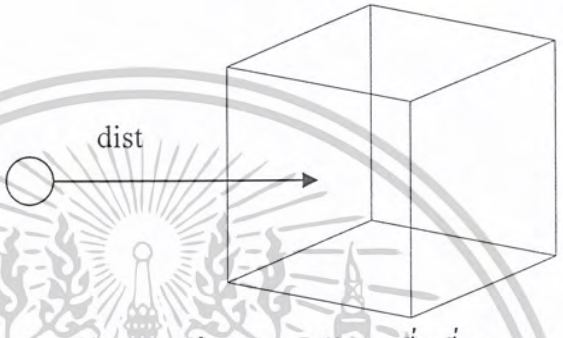


รูปที่ 2-20 แสดงเมื่อทรงกลมทั้ง 2 สัมผัสกัน

จากรูปที่ 2-20 เมื่อเกิดการชนกันกิดขึ้น สามารถหาระยะที่ชนกันเข้าไปว่าถึงขนาดเท่าใดโดยการนำรัศมีของทรงกลมทั้งสองมาคิดเพื่อจะคำนวณเป็นแรงในการกระทบของวัตถุต่อไป

- **การชนระหว่างทรงกลมกับทรงสี่เหลี่ยม**

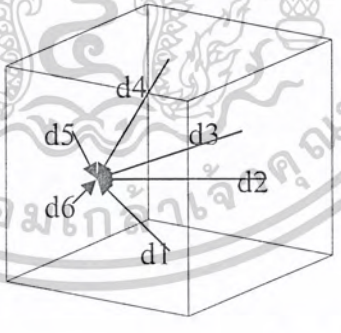
ในการตรวจสอบการชนกรณีนี้ เราจะใช้แนวความคิดที่ว่าถ้าจุดใดบนผิวของทรงกลมชนกับทรงสี่เหลี่ยม นั่นหมายความว่าเราจะปล่อยให้ผิวของทรงกลมผ่านผิวของทรงสี่เหลี่ยมเข้าไปนิดหน่อยแล้วคิดว่าถ้ามันทะลุเข้าไปแล้วค่าของระยะห่างจะต้องมีค่าติดลบกับทุกด้านของทรงสี่เหลี่ยม ไม่เช่นนั้นถ้ายังไม่ทะลุจะมีด้านใดด้านหนึ่งหรือมากกว่ามีค่าเป็นบวกอยู่



รูปที่ 2-21 แสดงการทดสอบการชนระหว่างทรงกลมกับทรงสี่เหลี่ยม

การหาระยะห่างระหว่างทรงกลมกับแต่ละด้านของทรงสี่เหลี่ยมเราจะใช้วิธีเดียวกับการหาการชนระหว่างทรงกลมกับระนาบ

distance = (sphPos – face[k].pos) dot face[k].normal

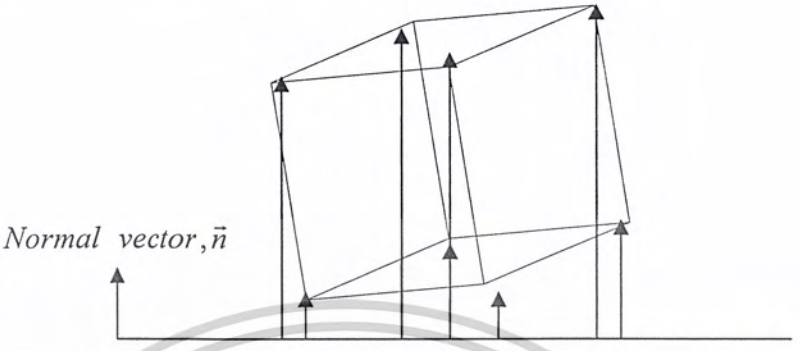


รูปที่ 2-22 แสดงเมื่อทรงกลมเกิดการชนกับทรงสี่เหลี่ยม

จากรูปที่ 2-22 เราจะเห็นได้ว่าเมื่อวัตถุชนเข้าไปแล้วเวคเตอร์ทั้งหมดของทุกหน้าจะตรวจสอบระยะห่างแล้วมีค่าติดลบทั้งหมด

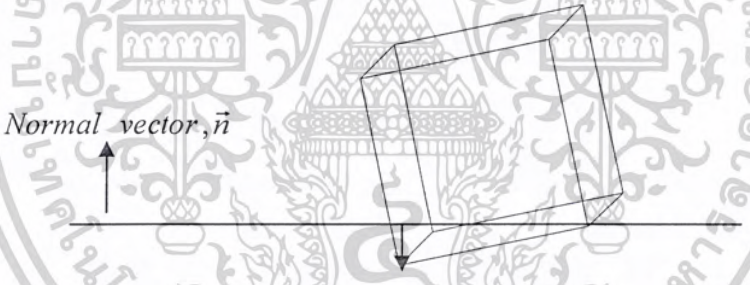
- **การตรวจสอบการชนระหว่างทรงสี่เหลี่ยมกับระนาบ**

การตรวจสอบกรณีนี้เพียงแต่เรานำแต่ละจุดยอด (vertex) ของทรงสี่เหลี่ยมมา dot product กับระนาบเพื่อที่จะหาว่ามีจุดใดจุดหนึ่งที่ทะลุผ่านระนาบไปแล้วหรือยัง



รูปที่ 2-23 แสดงการตรวจสอบการชนระหว่างทรงสี่เหลี่ยมกับระนาบ

รูปที่ 2-24 ได้แสดงว่าเมื่อวัตถุเคลื่อนที่ทะลุผ่านระนาบไปแล้วเวคเตอร์ที่ชี้ไปยังจุดยอดนั้นจะมีทิศตรงข้ามกับเวคเตอร์ของระนาบเมื่อนำมาดอทโปรดักซ์ก็จะทำให้ค่าที่ได้มีค่าติดลบนั่นเอง การหาระยะทางที่ทะลุผ่านเข้าไปก็ทำเช่นเดียวกับวิธีก่อนๆ หน้านี้เช่นเดียวกัน



รูปที่ 2-24 แสดงการชนกันระหว่างทรงสี่เหลี่ยมกับระนาบ

บทที่ 3
การพัฒนาด้านฮาร์ดแวร์

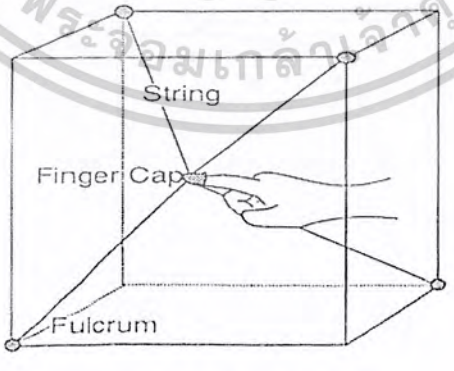
โครงงานนี้ใช้อุปกรณ์ SPIDAR เป็นส่วนเชื่อมต่อระหว่างผู้ใช้กับสภาวะแวดล้อมเสมือนเพื่อปฏิสัมพันธ์กับวัตถุจำลองที่สร้างขึ้น โดยอุปกรณ์นี้จะประกอบไปด้วย เชือก (String), ลูกรอก (Pulley) และ มอเตอร์ ตำแหน่งของนิ้วสามารถทราบด้วยการคำนวณความยาวเชือกที่ถูกขึงมายังที่ครอบนิ้ว (Finger cap) เชือกและมอเตอร์เป็นตัวสร้างแรงกระทำกับนิ้ว สภาพแวดล้อมเสมือนที่สร้างขึ้นในลักษณะคอมพิวเตอร์กราฟิก 3 มิติ สามารถให้รู้สึกเสมือนจริงในการสัมผัส

3.1 SPIDAR

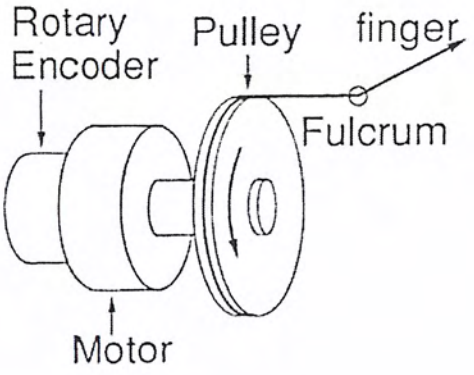
ในการที่เราจะจัดการเกี่ยวกับวัตถุ 3 มิติในโลกเสมือนจริง เราต้องการอุปกรณ์ ที่สามารถจะแสดงการเคลื่อนที่ได้เสมือนจริงและจะต้องมีการตอบโต้กลับด้วย ซึ่งการบังคับด้วยมือโดยตรงนั้นถือได้ว่าเป็นการควบคุมที่ดีที่สุดในการเคลื่อนที่วัตถุ

ซึ่งเป็นเหตุผลที่ทำให้ Prof. Makoto Sato และ Dr. Masahiro Ishii คิดค้นสร้าง อุปกรณ์ในการอ่านตำแหน่งของนิ้ว โดยใช้ เชือก ลูกรอก และมอเตอร์ ขึ้นมาโดยมีชื่อว่า SPIDAR (SPace Interface Device for Artificial Reality) การอ่านตำแหน่งของนิ้วจะอาศัยการคำนวณความยาวของเชือกที่นิ้วผู้ใช้และใช้มอเตอร์จะเป็นตัวคอยดึงเชือกเพื่อให้เกิดแรงต้านกลับ และพวกเขาได้สร้างสภาพแวดล้อมจำลองและใช้ SPIDAR นี้เป็นตัวควบคุมวัตถุภายในนั้นได้เสมือนกับได้สัมผัสจริง

ในส่วนนี้จะอธิบายถึงโครงสร้างของ SPIDAR และ การนำไปใช้ รูปที่ 3-1 แสดงถึงรูปร่างของ SPIDAR การทำงานโดยการสวมนิ้วเข้ากับที่ครอบนิ้วซึ่งมีเชือกสี่เส้นขึงอยู่ แต่ละเส้นเชือกขึงไปยังแต่ละมุมของโครงโลหะ (Frame) เชือกนั้นจะขึงไปพันรอบๆ ในลูกรอก ซึ่งลูกรอกนั้นจะติดอยู่ที่ตัวมอเตอร์อีกทีหนึ่ง ดังรูปที่ 3-2



รูปที่ 3-1 SPIDAR



รูปที่ 3-2 มอเตอร์และตัวถอดรหัส ที่ติดอยู่ตามมุมของโครง

3.1.1 การวัดตำแหน่งของนิ้ว

ให้ตำแหน่งของที่ครอบนิ้วคือ $P(x,y,z)$ และความยาวของเชือกเส้นที่ i^{th} เป็น $l_i (i=0-3)$ และให้ตัวมอเตอร์ติดอยู่กับมุมของโครงที่ตำแหน่ง $(-a,-a,-a)$, $(-a,a,a)$, $(a,-a,a)$ และ $(a,a,-a)$ ดังนั้นจะสร้างสมการได้สี่สมการดังนี้

$$(x+a)^2+(y+a)^2+(z+a)^2 = {l_0}^2$$
$$(x+a)^2+(y-a)^2+(z-a)^2 = {l_1}^2$$
$$(x-a)^2+(y+a)^2+(z-a)^2 = {l_2}^2$$
$$(x-a)^2+(y-a)^2+(z+a)^2 = {l_3}^2$$

แก้สมการทั้งสี่จะได้

$$4a(y+z) = {l_0}^2-{l_1}^2$$
$$4a(z+x) = {l_0}^2-{l_2}^2$$
$$4a(x+y) = {l_0}^2-{l_3}^2$$

สามารถบอกตำแหน่งของที่ครอบนิ้วได้ดังนี้

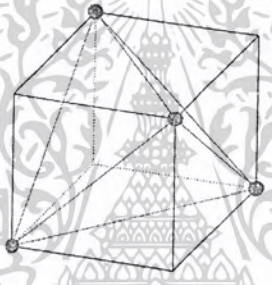
$$x=({l_0}^2+{l_1}^2-{l_2}^2-{l_3}^2)/8a$$
$$y=({l_0}^2-{l_1}^2+{l_2}^2-{l_3}^2)/8a$$
$$z=({l_0}^2-{l_1}^2-{l_2}^2+{l_3}^2)/8a$$

3.1.2 การเกิดแรงที่กระทำกับนิ้ว

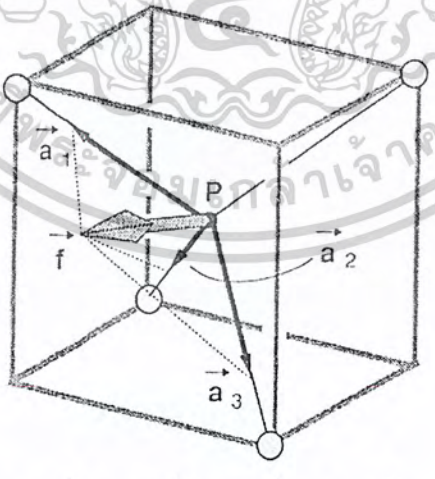
SPIDAR ใช้แรงลัพธ์ของความตึงเชือกเพื่อให้รู้สึกสัมผัสถึงแรงดึง ซึ่งจะมีลักษณะพีระมิดมีมิติที่จะใช้ในการออกแรงดึง เมื่อให้เชือกทั้งสี่มุมตึงโดยการดึงจากมอเตอร์แรงที่เกิดขึ้นที่ตำแหน่งของที่ครอบนิ้วเป็นเวคเตอร์แรง f และเวคเตอร์หน่วยของแรงตึงเชือกเป็น $\vec{u}_i (i = 0,1,2,3)$ ดังนั้นแรงทั้งหมดที่กระทำกับที่ครอบนิ้ว (ดังรูปที่ 3-3) คือ

$$\vec{f} = \sum a_i \vec{u}_i \qquad (a_i >= 0)$$

เมื่อ a_i คือค่าของแรงตึงเชือกแต่ละเส้น สามารถรวมแรงลัพธ์ของทุกๆเส้นในทุกๆทิศทางโดยการบังคับของ a_i ทั้งหมดดังรูปที่ 3-4 ซึ่งในรูปนั้นจะเห็นได้ว่าเกิดการเคลื่อนนิ้วไปด้านหลังทำให้เชือกเส้นที่อยู่มุมขวาบนจะไม่ได้ใช้ดังนั้นแรงที่เกิดขึ้นมีทิศตรงข้ามกับแรงที่เรากระทำจากนั้นเราจะทำการแตกแรงให้ตกลงไปอยู่บนเส้นเชือกที่เหลืออีกทั้งสามเส้นเพื่อที่จะได้ทราบว่าเชือกจะต้องออกแรงตึงเท่าใด



รูปที่ 3-3 ลักษณะของการตึงเชือก



รูปที่ 3-4 แรงที่เกิดรวมที่จุดครอบนิ้ว

3.2 SPIDAR II

ซึ่งเป็นเวอร์ชั่นที่สองของตัว SPIDAR ซึ่งมีการพัฒนาให้มีจุดการควบคุมเพิ่มขึ้นอีกหนึ่งจุดแต่จะอาศัยหลักการขึงเชือกเหมือนในเวอร์ชั่นแรกแต่จะใช้มุมของเฟรมที่เหลืออีกสี่มุมในการติดยึดมอเตอร์ ในเวอร์ชั่นที่สองนี้จะสามารถทำงานเพิ่มขึ้นได้อีกอย่างหนึ่งคือการหยิบยกวัตถุซึ่งจะประกอบไปด้วย

- การเคลื่อนย้ายจากตำแหน่งปัจจุบันหรือการหันเหของวัตถุนั้นๆไปยังอีกจุดหนึ่ง
- การยึดจับวัตถุ

งานเหล่านี้จะต้องใช้อย่างน้อยสองนิ้ว ในหัวข้อนี้จะอธิบาย SPIDAR II สำหรับสองนิ้ว ดังรูปที่ 3-5 SPIDAR II สามารถวัดการหันเหของวัตถุได้นอกเหนือจากการวัดตำแหน่งของนิ้วซึ่งต้องอยู่ภายใต้สมมติฐานที่ว่าวัตถุอยู่ขนานกับนิ้วเสมอ สิ่งหนึ่งที่สำคัญคือการพิจารณาถึงการสิ่งที่รบกวนของเชือกทั้งหมดซึ่งจะทำให้ผลที่ได้ผิดพลาดไป

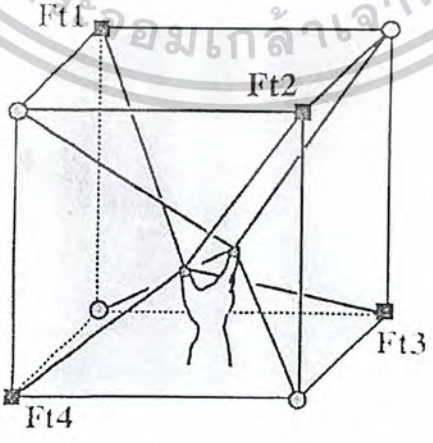
3.2.1 การหยิบและวาง

ระหว่างการหยิบและวางจะมีแรงอยู่สามชนิดเกิดขึ้นในการจำลองตามนี้

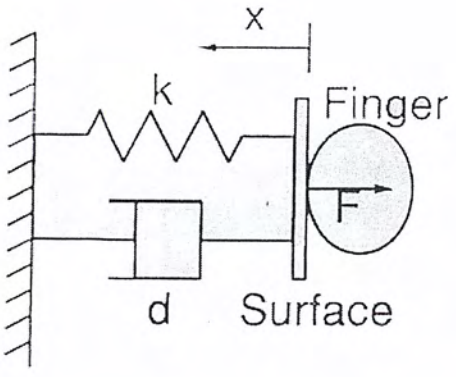
- N แรงกระทำโต้ตอบ (Reaction) ระหว่าง วัตถุกับนิ้ว
- R แรงกระทำโต้ตอบ ระหว่างวัตถุกับวัตถุแวดล้อมอื่นหรือแม้แต่ตัวของวัตถุเอง
- W น้ำหนักของวัตถุ

วัตถุที่จำลองขึ้นนั้นจะมีรูปแบบคล้ายกับโลกความเป็นจริง เมื่อวัตถุสองอันชนกันในความเป็นจริงจะต้องไม่เกิดการทะลุผ่านกันไปได้ อย่างไรก็ตามในโลกเสมือนอาจจะเกิดขึ้นได้เล็กน้อยเพราะขึ้นอยู่กับขอบเขตของรอบเวลาของระบบ(Cycle time of the system) ในการจัดการกับวัตถุแข็งในระบบจำลองนั้นจะมีลักษณะของแรงสปริง (Spring Damper) ดังแสดงในรูปที่ 3-6 ให้ค่าคงที่สปริงเป็น k และ Damper เป็น d และตำแหน่งของผิวของวัตถุเป็นด้าน x ฉะนั้นแรงที่กระทำกับนิ้วคือ

$$F = kx + dx$$



รูปที่ 3-5 SPIDAR II

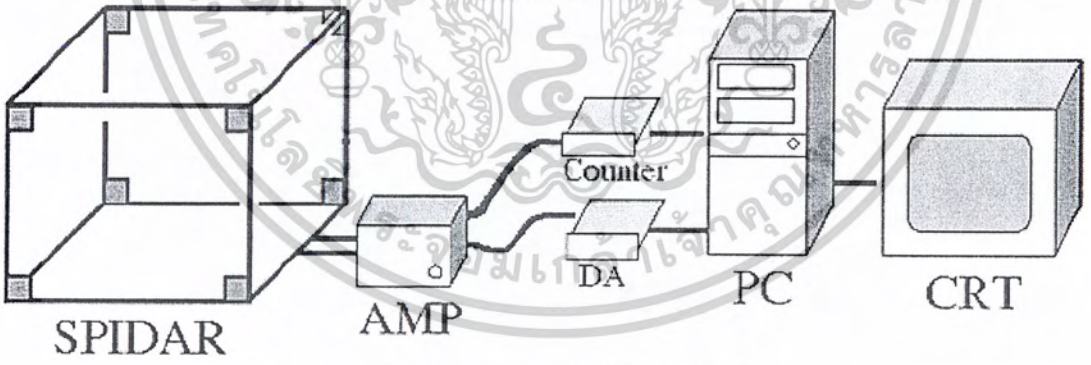


รูปที่ 3-6 ลักษณะแรงที่กระทำระหว่างนิ้วกับวัตถุ

3.3 ส่วนประกอบของฮาร์ดแวร์ทั้งหมด

ในการสร้างโครงงานนี้ส่วนฮาร์ดแวร์นี้จะประกอบด้วยอุปกรณ์อยู่สามส่วนหลักๆด้วยกันนั่นก็คือ

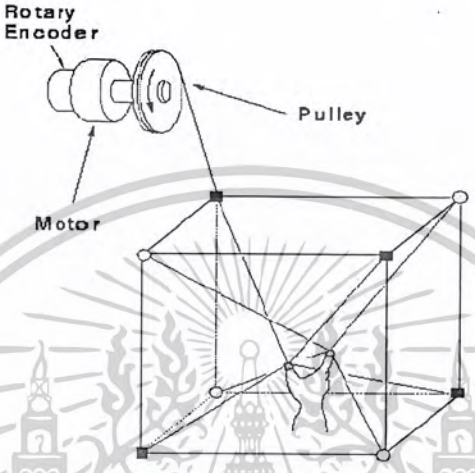
- SPIDAR II
- หน่วยขยายแรงดันกระแส (Amplifier) สำหรับมอเตอร์
- การ์ดอินเตอร์เฟส D/A และ Counter ชนิด ISA-BUS
- เครื่องคอมพิวเตอร์สำหรับการประมวลผลและจอภาพสำหับการแสดงผลภาพ



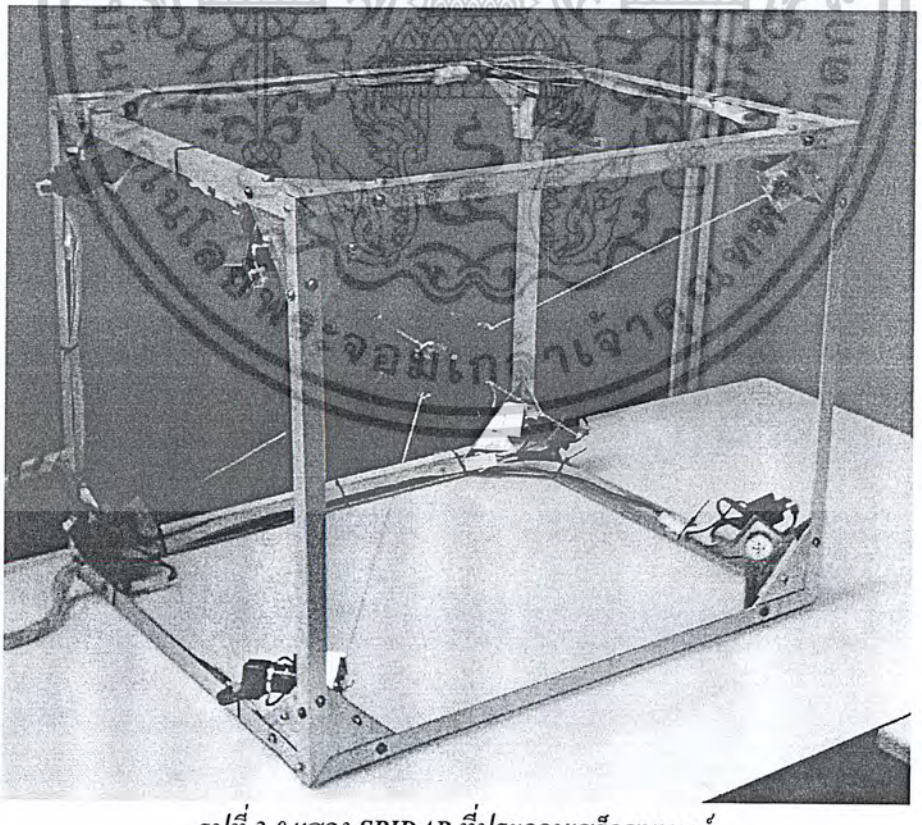
รูปที่ 3-7 ส่วนประกอบฮาร์ดแวร์ทั้งหมดในโครงงาน

3.3.1 การพัฒนา SPIDAR ในโครงงานนี้

เป็นโครงเหล็กสี่เหลี่ยมลูกบาศก์ขนาด 80 x 60 x 60 cm. โดยที่มุมแต่ละมุมของโครงเหล็กจะมี DC motor ติดอยู่ทั้ง 8 มุมซึ่งแต่ละตัวจะร้อยเชือกอยู่ในลูกรอกซึ่งติดอยู่ที่ปลายแกนของ DC motor และปลายเส้น 4 เส้นจะผูกรวมกันที่ปลายนิ้วของผู้ใช้ เมื่อเคลื่อนที่นิ้วมือของผู้ใช้ไปสัมผัสกับวัตถุเสมือน โปรแกรมจะควบคุมมอเตอร์เพื่อให้เกิดแรงดึงนิ้วมือผู้ใช้ ทำให้เกิดแรงดึงที่ให้ความรู้สึกในลักษณะเช่นเดียวกับนิ้วมือสัมผัสวัตถุจริง



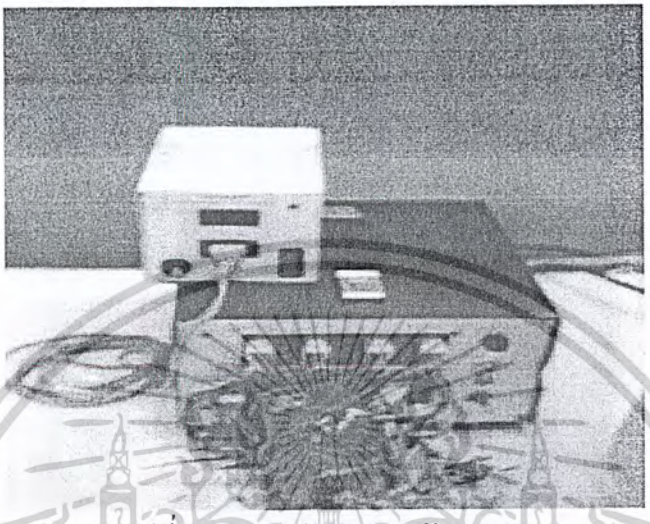
รูปที่3-8 แสดงโครงสร้างและการผูกเชือกของ SPIDAR



รูปที่ 3-9 แสดง SPIDAR ที่ประกอบเสร็จสมบูรณ์

3.3.2 หน่วยขยายแรงดันกระแส (Amplifier)

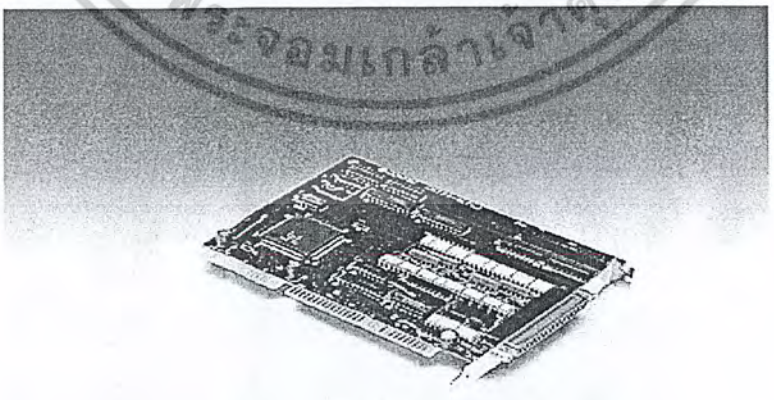
มีส่วนประกอบภายในอยู่ด้วยกันสองส่วนคือ หน่วยขยายแรงดันกระแสซึ่งจะใช้ในการขยายแรงดันกระแสก่อนที่จะส่งไปยังมอเตอร์ ส่วนที่สองคือส่วนของบัฟเฟอร์ที่รับมาจาก Counter ที่อ่านได้จาก Rotary encoder ที่ตัวมอเตอร์ก่อนที่จะส่งไปยังตัวการ์ดอินเตอร์เฟสในเครื่องคอมพิวเตอร์



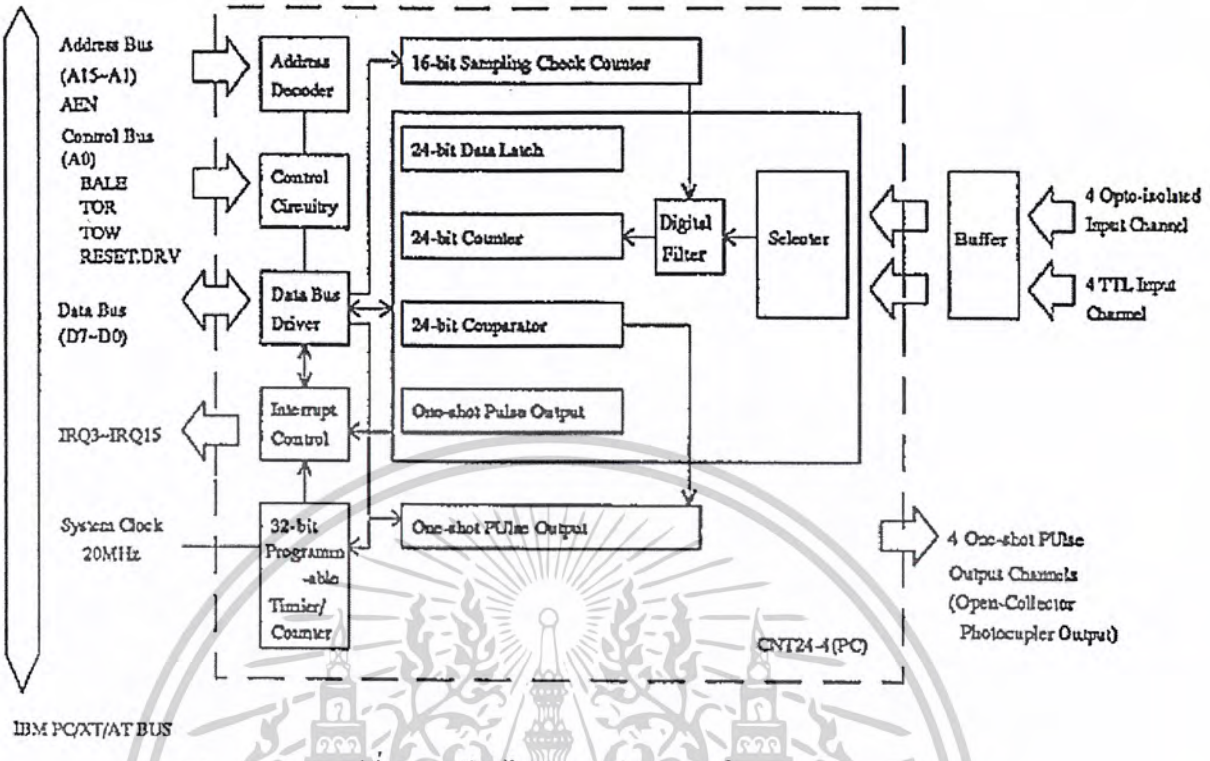
รูปที่3-10 แสดงตัวขยายแรงดันกระแส

3.3.3 แผงวงจรนับการหมุนของมอเตอร์ (Counter)

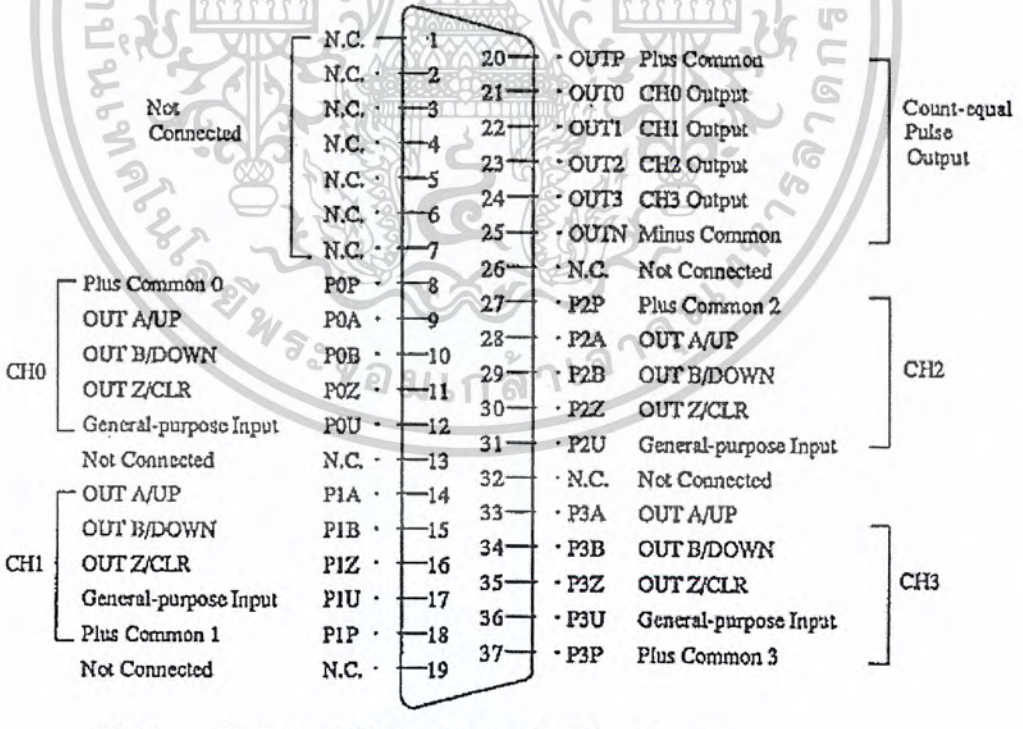
จะทำหน้าที่นับการหมุนของมอเตอร์ซึ่งจะอ่านค่าจากตัวถอดรหัส ที่ติดอยู่กับด้านท้ายของมอเตอร์แล้วส่งข้อมูลให้คอมพิวเตอร์ โดยผ่านทางช่องเสียบ ISA ต่อไป ซึ่งตัวเคาเตอร์นี้ต่อหนึ่งการ์ดมีเพียง 4 ไชแนลเราจึงจำเป็นที่จะต้องใช้ทั้งหมดสองการ์ดด้วยกันในการเชื่อมต่อเพราะจะต้องใช้กับมอเตอร์หนึ่งตัวต่อหนึ่งไชแนล



รูปที่ 3-11 แผงวงจรนับการหมุนของมอเตอร์รุ่น CNT24-4 (PC)

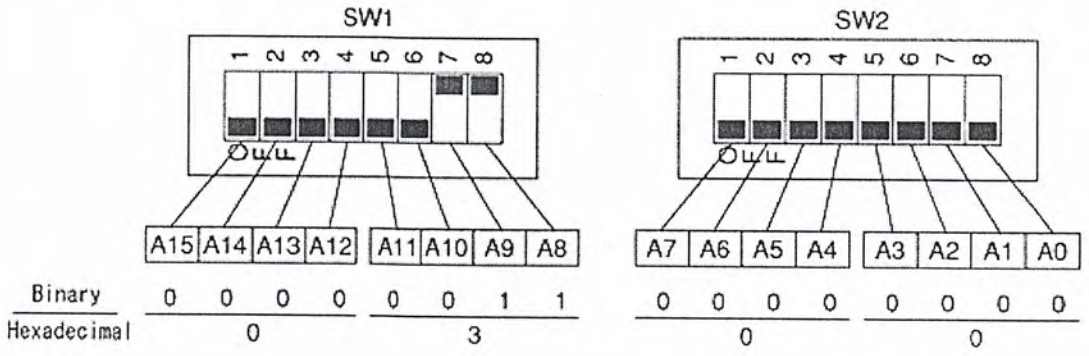


รูปที่ 3-12 บล็อคไดอะแกรมวงจรภายใน



Note! ***Each channel has an independent Plus Common.***

รูปที่ 3-13 ตำแหน่งของขาในการเชื่อมต่อ

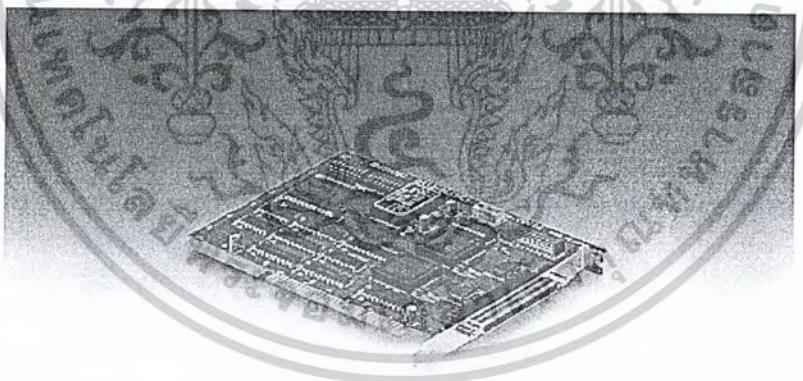


รูปที่ 3-14 การติดตั้งดิ๊ปสวิชต์เพื่อการกำหนดแอดเดรส

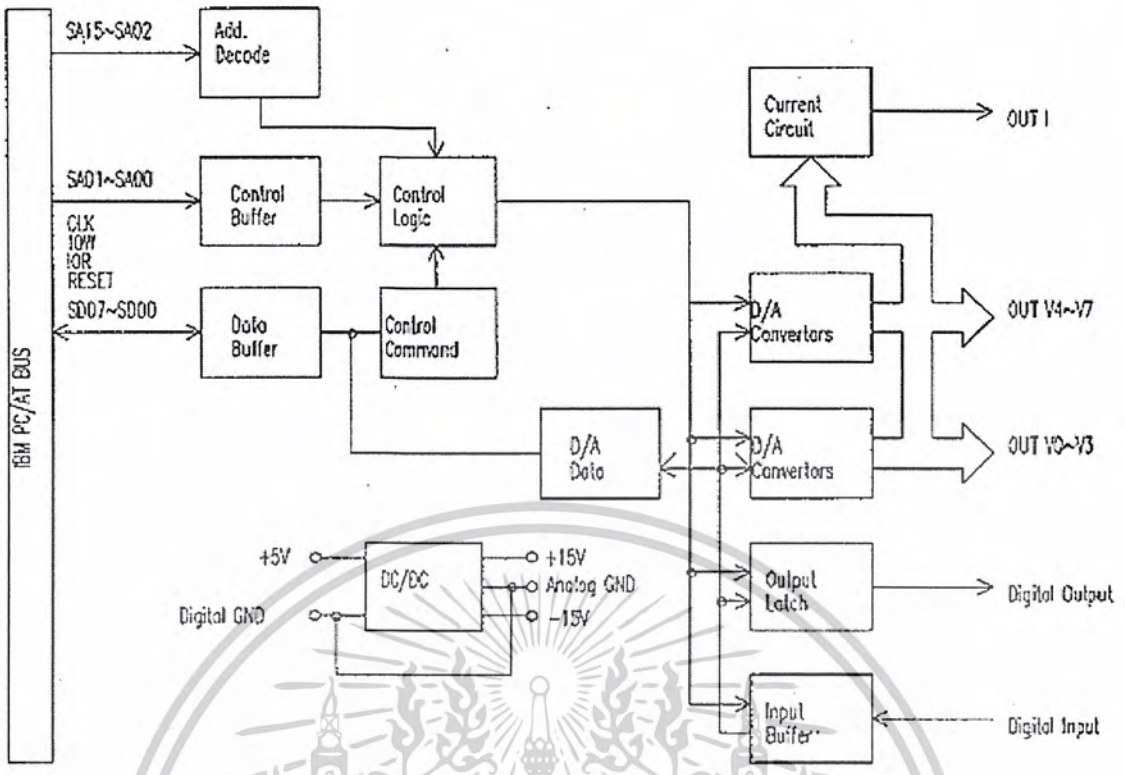
จากรูปที่ 3-14 การติดตั้งดิ๊ปสวิชต์นี้เพื่อใช้ในการกำหนดช่องทางการเชื่อมต่อเพื่อใช้ในการเชื่อมต่อจากการเขียนโปรแกรมของโปรเจ็ค

3.3.4 แผงวงจรแปลงสัญญาณดิจิตอลเป็นแอนนะลอก (Digital to Analog Converter) รุ่น DA12-8L(PC)

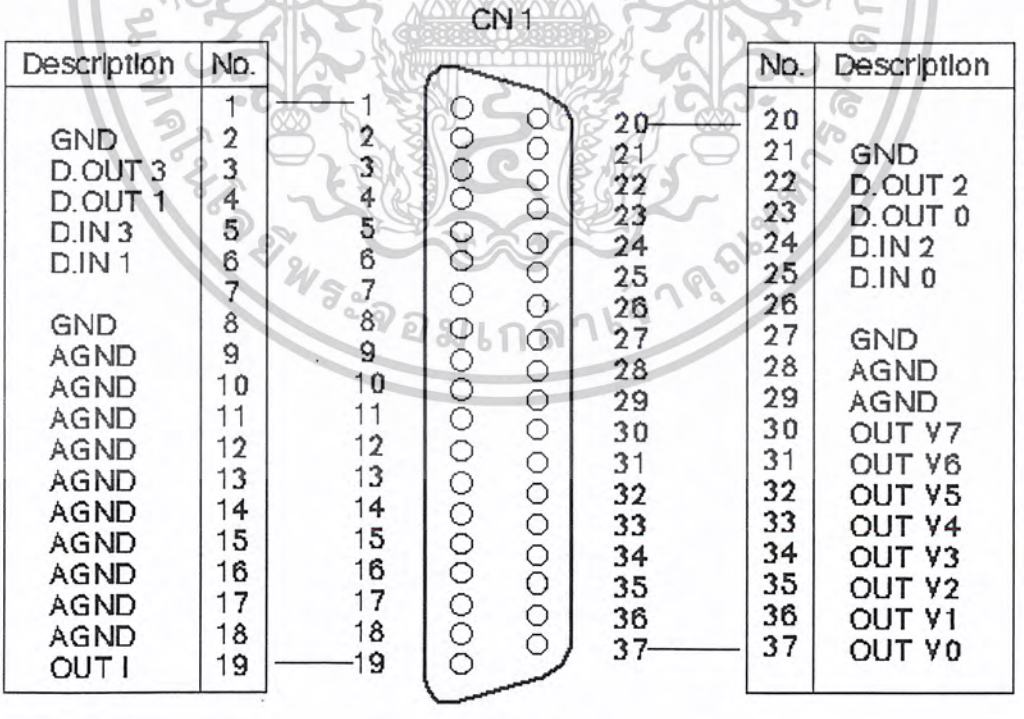
ทำหน้าที่แปลงสัญญาณที่ได้จากคอมพิวเตอร์ซึ่งเป็นดิจิตอลให้กลายเป็น ไปเป็นสัญญาณ แอนนะลอกเพื่อใช้ในการขับเคลื่อน มอเตอร์ โดยผ่านสัญญาณไปยังแอมปริไฟล์ก่อนเพื่อขยายแรงดันกระแส การ์ดนี้มี 8 ไชแนลอยู่แล้วจึงใช้เพียงการ์ดเดียวในการเชื่อมต่อ



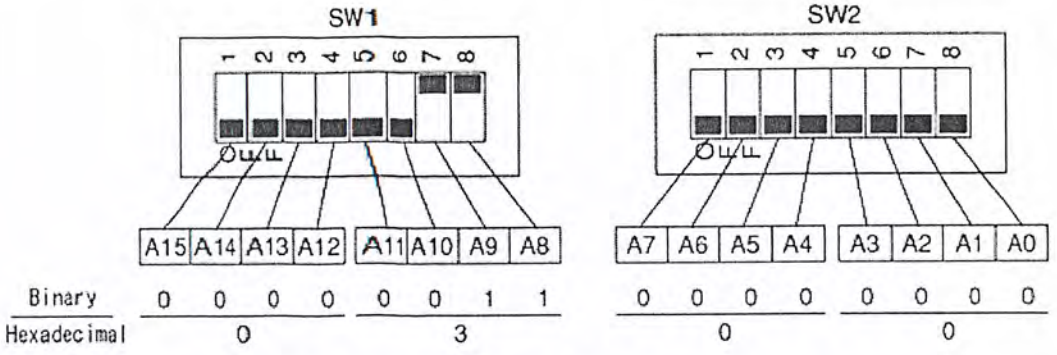
รูปที่ 3-15 แผงวงจรแปลงสัญญาณดิจิตอลเป็นแอนนะลอกรุ่น DA12-8L (PC)



รูปที่ 3-16 บล็อกไดอะแกรมของวงจรแปลงสัญญาณดิจิตอลเป็นแอนาลอก



รูปที่ 3-17 ตำแหน่งของขาในการเชื่อมต่อ



รูปที่ 3-18 การติดตั้งดิ๊ปสวิชต์เพื่อการกำหนดแอดเดรส

3.3.5 อุปกรณ์ที่ใช้ในการประมวลผลและแสดงผลภาพ

เครื่องคอมพิวเตอร์ส่วนบุคคล (Personal Computer)

หน่วยประมวลผลกลาง Intel Pentium II 400Mhz

Ram 128MB

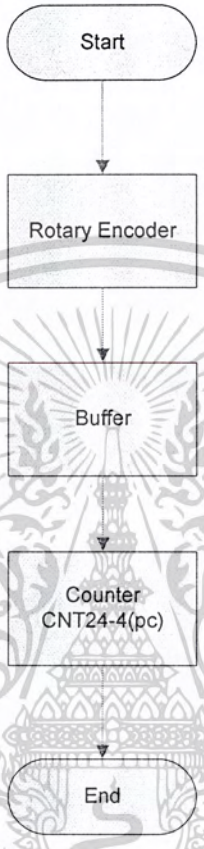
Display Card ATI 3D Range Pro(4MB)

ระบบปฏิบัติการ Microsoft Windows 98

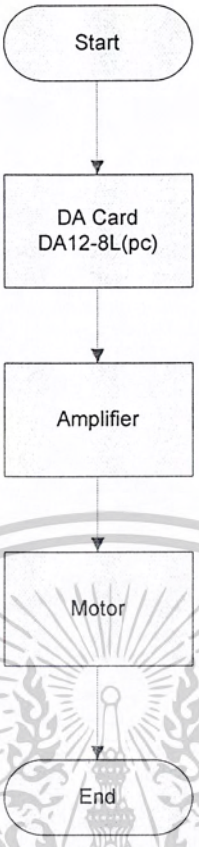
3.4 หลักการทำงาน

จากการที่เราเคลื่อนนิ้วที่สวมอยู่กับแหวนจะทำให้มอเตอร์นั้นเกิดการหมุน แล้วตัวดีโคดเดอร์ที่ติดอยู่กับมอเตอร์อ่านค่าเป็นจำนวนพัลซ์ (pulse) ซึ่งตัวมอเตอร์เองมีสเป็คเท่ากับ 500 พัลซ์ จากนั้นจะส่งไปยังบัฟเฟอร์ที่ตัวแอมปริไฟล์ก่อนที่จะส่งไปยังการ์ดเคาเตอร์ที่เครื่องคอมพิวเตอร์จากนั้นโปรแกรมก็จะทำการคำนวณได้จากการที่เราทราบ ว่าหนึ่งพัลซ์นั้นจะดึงเชือกไปยาวเท่ากับ 2.513 x 10^{-5} เมตร ซึ่งได้มาจากการคำนวณโดยที่ทราบค่ารัศมีของ ลูกลอกมีค่าเท่ากับ 0.008 เมตร ดังนั้นจึงสามารถหาความยาวของเชือกต่อหนึ่งพัลซ์ได้ 2*PI*0.008[m] / 500[pulse]*4 = 2.513 x 10^{-5} เมตร ซึ่งนำค่าที่ได้นี้ไปคำนวณกับค่าพัลซ์ที่อ่านเข้ามาได้จริงก็จะทำให้ได้ความยาวเชือก จากนั้นก็จะนำไปหาตำแหน่งที่ต้องนำความยาวของเชือกของมอเตอร์ทั้ง 4 มาคิดว่าตอนนี้จริงๆ แล้วนิ้วอยู่ที่ตำแหน่งใดในเฟรมแล้วนำไปสู่กรบวนการตรวจสอบการชนกับวัตถุต่างๆ ในโลกเสมือน แต่ก่อนอื่นใดในการเริ่มต้นของโปรแกรมก็จะมีการกำหนดค่าเบื้องต้นทั้งหมดให้กับสภาพแวดล้อมรวมถึงวัตถุ และค่าเริ่มต้นของตัวอุปกรณ์ที่ใช้ในการติดต่อให้พร้อมที่จะใช้งาน เมื่อเราสามารถอ่านค่าตำแหน่งได้แล้วระบบการจำลองก็จะทำการตรวจสอบตลอดเวลาตามการเคลื่อนไหวของเฟรมต่อวินาทีซึ่งมีหน่วยอยู่ที่ประมาณ 20-30 Hz แต่การส่งค่าพัลซ์จากมาเตอร์มานั้นจะมีความเร็วอยู่ที่ประมาณ 1 kHz เมื่อมีการตรวจสอบแล้วว่าเกิดการชนเกิดขึ้นก็จะส่งแรงที่กระทำกับตำแหน่งของนิ้วที่อยู่ในโลกเสมือนออกไป เราจะต้องทำการคำนวณว่าแรงเท่านี้จะต้องมีแรงดันออกไปมากกี่โวลท์โดยกำหนดค่าแรงดันต่อนิวตันไว้เป็นค่าคงที่ซึ่งมีค่าเท่ากับ 0.57 V และกำหนดให้ค่าดิจิตอลไปยังแอนนาลอกต่อแรงดันโวลท์มีค่าเท่ากับ 409.6 เมื่อทราบสองค่านี้เราก็จะสามารถคำนวณ

ค่าแรงที่เกิดจากการชนออกไปในรูปแบบของแรงดันเพื่อส่งไปยัง Amplifier โดยดิจิตอลไปยังแอนนาลอก ใช้แรงดันโวลท์ต่อนิวตันจะมีค่าเท่ากับการนำค่า 0.57 x 409.5 โดยที่แรงดันนี้จะเกิดที่ตัวมอเตอร์แต่ละตัว อาจไม่เท่ากัน ขึ้นอยู่กับว่าหลักการที่จะหาว่าแรงที่เกิดขึ้นนั้นเมื่อแตกแรงหลักออกเป็นแรงย่อยแล้วให้อยู่ในทิศทางของเชือกแต่ละเส้นที่ถูกต้องนั้นมีค่าเป็นเท่าใด



รูปที่ 3-19 ขั้นตอนการทำงานของการอ่านค่าจากมอเตอร์



รูปที่ 3-20 ขั้นตอนการควบคุมการหมุนของมอเตอร์

บทที่ 4
การพัฒนาด้านซอฟแวร์

หลังจากที่ได้กล่าวถึงการพัฒนาทางด้านฮาร์ดแวร์มาแล้วในบทที่ 3 ในบทนี้จะขอกล่าวถึงการพัฒนาด้านซอฟแวร์ที่ใช้ในระบบการจำลองซึ่งแบ่งออกเป็นส่วนย่อยๆดังนี้

4.1 การสร้างสภาพแวดล้อมเสมือน (Virtual World)

เริ่มตั้งการสร้างสภาพแวดล้อมที่ว่างเปล่า เพียงแต่เรากำหนดคุณสมบัติในการแสดงผลไว้โดยการตั้งค่าตำแหน่งของกล้องคุณสมบัติของแสงลักษณะของแสง จากนั้นก็จะเริ่มทำการกำหนดคุณสมบัติของวัตถุต่างๆมากมายดังนี้

- กล้อง
  มุมมองการเห็นภาพเราได้กำหนดตำแหน่งของกล้องไว้ที่ x = 0, y = 12, z = -50 ซม. แล้วให้กล้องส่องไปยังตำแหน่ง (0,0,0)
- แสง
  ตำแหน่งของแสงอยู่ที่ x = 60, y = 250 , z = -200 ซม. ให้คุณสมบัติของแสงเป็นแบบจุดแสงและให้การสะท้อนของแสงเป็นแบบ ambient
- ระนาบ และ กำแพง
  กำหนดให้พื้นมีขนาดกว้างxยาว 60x80 ซม. ที่ตำแหน่ง y = -12 ซม. ขนาดของกำแพงตามแนวแกน x สองด้านให้มีขนาด 25x60 ซม. ที่ตำแหน่ง x = -40 ซม. กับ x = 40 ซม. กำแพงด้านหลังซึ่งตามแนวแกน z กำหนดให้มีขนาด 25x80 ซม. อยู่ที่ตำแหน่ง z = 30 ซม.
- วัตถุทรงกลม
  เริ่มต้นกำหนดให้ทรงกลมมีรัศมี 5 ซม. มีมวล = 0.25 นิวตัน แรงเสียดทานที่พื้นผิวมีค่าเท่ากับ 0.98 $kg*m^2/sec$
- วัตถุทรงสี่เหลี่ยม
  กำหนดให้มีขนาด กว้าง x ยาว x สูง = 8 x 8 x 8 ซม. มีมวล = 0.3 นิวตัน มีแรงเสียดทานที่ผิวเท่ากับ 0.8 $kg*m^2/sec$

4.2 การสร้างการจำลอง (Simulation)

ในการสร้างการจำลองนั้นเราได้ทำการสร้างวัตถุทรงกลมขึ้นมา 2 ลูก และอีกหนึ่งกล่องลูกบาศก์ โดยมีทรงกลมขนาดเล็กสองลูกเพื่อใช้ในการบอกตำแหน่งของนิ้วที่อยู่ในโลกเสมือนเมื่อเริ่มการจำลองจะทำให้เกิดกรณีขึ้นดังนี้

- ทรงกลมปะทะกับทรงกลม

  อัลกอริทึมที่ใช้ในการค้นหาการปะทะนี้ จะใช้วิธีการตรวจสอบว่าเมื่อทรงกลมทั้งสองนั้นเข้ามาปะทะกัน ก็แสดงว่าระยะห่างของวัตถุทรงกลมทั้งสองมีค่าน้อยกว่ารัศมีของทรงกลมทั้งสองรวมกัน

```
distance = sphere1.pos - sphere2.pos
```

  จากนั้นเมื่อจะทำการตรวจสอบก็เพียงนำค่า distance นี้มาเปรียบเทียบกับ (r1+r2) ซึ่งถ้ามีค่าน้อยกว่าก็แสดงว่าเกิดการชนกันเกิดขึ้นแล้ว หลังจากเราทราบว่ามีการชนเกิดขึ้นเราก็ทำการคำนวณหาแรงที่เกิดจากการชนโดยนำส่วนที่เหลื่อมจากการชนนั้นมา dot product กับ normal vector แล้วคูณด้วยค่าแรงสปริงเข้าไปจะทำให้ได้แรงที่ได้จากการชนแล้วนำแรงนี้รวมเข้าไปกับแรงที่มีอยู่แล้วของวัตถุทรงกลม

- ทรงกลมปะทะกับกล่อง

  ใช้หลักการดังที่ได้กล่าวไว้ในบทที่สอง ในเรื่องการตรวจสอบการชนระหว่างทรงกลมกับทรงสี่เหลี่ยม โดยเริ่มแรกจะต้องหาตำแหน่งของด้านแต่ละด้านของกล่องพร้อมทั้งหา normal vector ของด้านแต่ละด้าน หลังจากนั้นเราจะทำการหาระยะห่างจากทุกๆด้านของกล่องมายังทรงกลม โดยถ้าทรงกลมนั้นอยู่นอกกล่องคือยังไม่ได้ปะทะกับกล่อง ก็จะทำให้ผลจากการหาค่าระยะห่างจากทุกๆด้านจะมีด้านใดด้านหนึ่งหรือมากกว่าหนึ่งด้านที่มีค่าเป็นบวก แต่เมื่อใดก็ตามที่ผลของการหาระยะห่างจากด้านทุกๆด้านของกล่องมายังทรงกลมมีค่าเป็นลบทุกด้าน แสดงว่าทรงกลมนั้นได้เข้าไปปะทะกับกล่องแล้วโดยตัวอย่างการหาซึ่งนำมาจากโปรแกรม

```
for(int k=1; k<=6; k++)
{
      dist[k] = (sphPos - face[k].pos) * face[k].normal;
}
```

  หลังจากที่ทราบแล้วว่ามีการปะทะกันเกิดขึ้น ต่อไปเราก็จะทำการหาว่าทรงกลมนั้นได้ปะทะที่ด้านไหนของกล่อง โดยแนวคิดที่ว่าทรงกลมที่ปะทะด้านใดอยู่นั้นค่าระยะห่างที่ได้ติดลบนั้นจะมีค่าน้อยที่สุด เมื่อได้ด้านที่เราชนแล้วต่อไปอีกเราจะต้องหาตำแหน่งที่ชนโดยเอาด้านที่ชนมาพิจารณาเป็นกรณีๆ ไปแล้วหาแรงที่เกิดจากการชนระหว่างสองวัตถุเพื่อนำไปรวมกับแรงที่มีอยู่ในตัววัตถุเดิมอยู่ก่อนแล้วก่อนที่จะนำไปอัพเดทตัววัตถุต่อไป

- ทรงกลมปะทะกับระนาบ

  ในการทดสอบกรณีนี้จะใช้วิธีเดียวกันกับทรงกลมปะทะกับกล่อง แต่ระนาบมีเพียงด้านเดียว ดังนั้นจึงไม่ต้องมีการทำลูปเพื่อหาด้านหลายด้าน ส่วนการหาแรงนั้นก็สามารถหาโดยใช้วิธีเดียวกันได้เช่นกัน

- กล่องปะทะกับระนาบ

  การทดสอบนี้ใช้วิธีนำเอาจุดยอดทั้งแปดจุดของกล่องมา dot product กับระนาบเพื่อหาว่ามีจุดยอดใดไปปะทะกับระนาบบ้าง แล้วหาว่าทะลุเข้าไปเท่าใดเพื่อนำไปหาแรงที่เกิดจากการปะทะ โดยหาได้จากนำตำแหน่งของระนาบไปลบกับค่าของจุดยอดที่ปะทะ ค่าที่ได้มาจะเป็นเวคเตอร์ที่ชี้จากระนาบไปยังจุดยอดที่ทะลุอยู่ใต้ระนาบหลังจากนั้นนำค่าที่ได้นี้ไปทำการ dot product กับ normal vector ในทิศทางตรงข้ามจากปกติคือ – normal vector นั่นเป็นเพราะตอนนี้เราต้องการหาระยะทางซึ่งจุดยอดนั้นทะลุเข้ามาในทิศทางตรงข้ามกับ normal vector แล้วเราจึงต้องทำให้มันเป็นค่าลบเสียก่อนเมื่อ dot product แล้วผลลัพธ์ก็คือความยาวจากระนาบจนถึงจุดยอดที่ทะลุผ่านซึ่งอยู่ในทิศทาง – normal vector เรียบร้อยแล้ว ตัวอย่างจากโปรแกรมเป็นดังนี้

```
for(int i   = 1; i <= 8; i++)
{
        dist[i] = (vertex[j] - floorPos) * -floorNormal;
}
```

4.3 ส่วนติดต่อกับฮาร์ดแวร์

ได้ทำการศึกษาการติดต่อกับการ์ดอินเตอร์เฟส จากไลบารี่ซึ่งเป็นไลบารี่ที่สามารถติดต่อกับตัวการ์ดอินเตอร์เฟสได้อยู่แล้วจึงนำคลาสที่ใช้ติดต่อกับฮาร์ดแวร์ทั้งคลาส CHapticDevice, CSpidar, CSpidarMotor, CSpidarEncoder ซึ่งเพียงทำการเรียกเมทอดในการกำหนดค่าเริ่มต้นให้กับตัวมอเตอร์และตัวดีโคดเดอร์ ส่วนการกำหนดการใช้ใช้งานก็เพียงเรียกเมทอดเซท เพื่อใส่แรงที่ได้จากการชนกันเข้าไปเมทอดก็จะทำการคำนวณค่าที่จะส่งไปยังการ์ดให้เรียบร้อยทันที การอ่านค่าจากดีโคดเดอร์ก็เช่นกันสามารถใช้เมทอดจากคลาส CSpidarEncoder ในการอ่านค่าได้เป็นความยาวเชือก

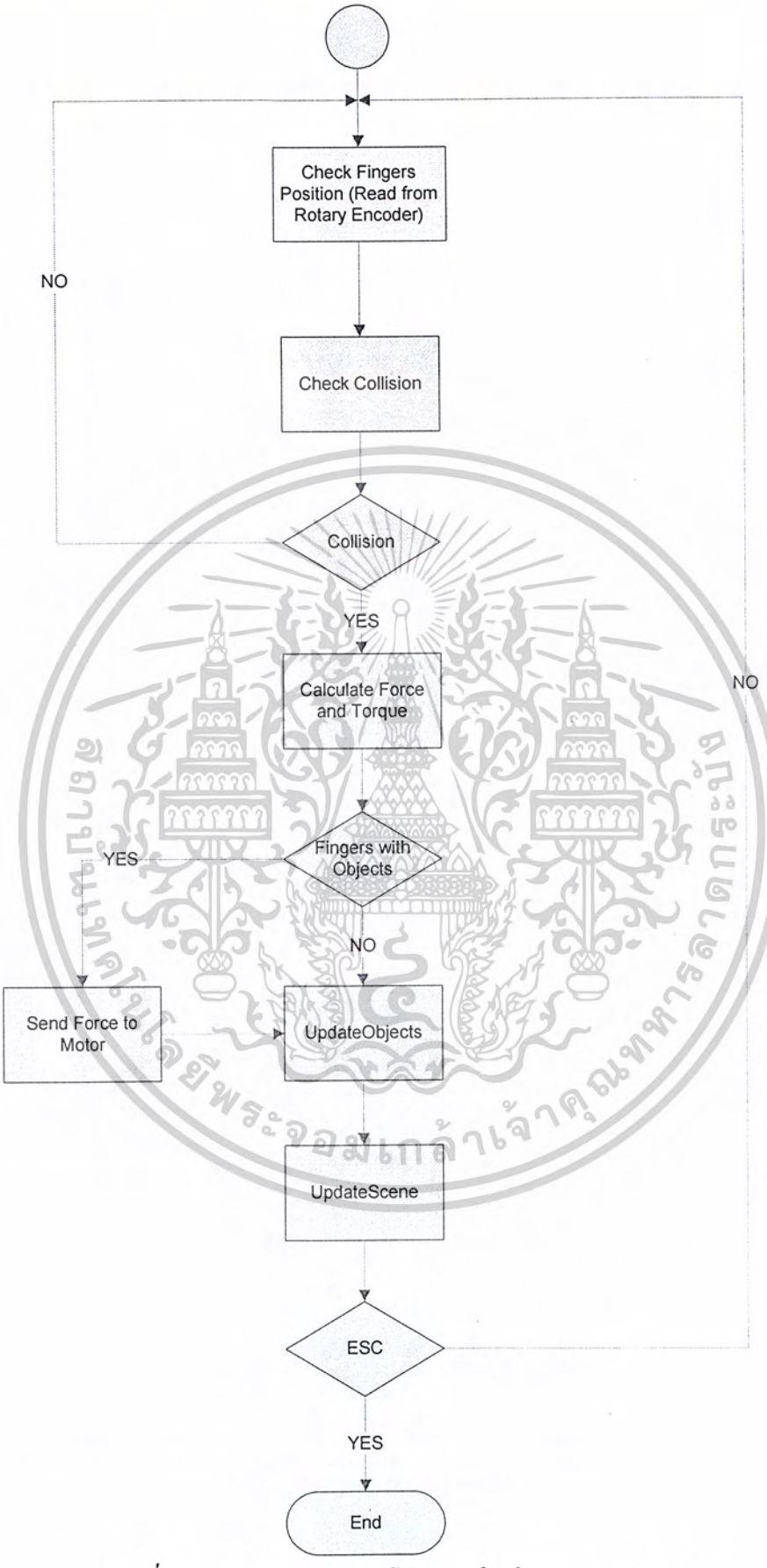
4.4 หลักการทำงานของระบบ

เมื่อเริ่มการทำงานของโปรแกรม เริ่มแรกโปรแกรมจะทำการเริ่มต้นระบบโดยการจัดตำแหน่งของแหวนโดยจะดึงเชือกให้ตึงทุกเส้น และผู้ใช้จะต้องคอยจัดให้วงแหวนทั้งสองวงซ้อนกันและให้อยู่ในตำแหน่งจุดศูนย์กลางของ SPIDAR โดยประมาณ หลังจากนั้นหน้าจอจะแสดงผลเพื่อถามการยืนยันจากผู้ใช้ว่าได้จัดตำแหน่งของแหวนเรียบร้อยหรือยัง เมื่อผ่านส่วนนี้ขั้นต่อไปโปรแกรมจำทำการสร้างส่วนประกอบต่างๆ ของโลกเสมือนตามหัวข้อที่ 4.1 โดยการทำงานแสดงในรูปที่ 4-1

ขั้นตอนต่อมาในส่วนของการจำลอง โปรแกรมจะทำการตรวจสอบตำแหน่งนิ้วมือของผู้ใช้พร้อมกับแสดงผลการเคลื่อนไหวของนิ้วมือให้เห็นด้วยตำแหน่งลูกบอล 2 ลูก หากตำแหน่งอ้างอิงดังกล่าวสัมผัสกับวัตถุเสมือน โปรแกรมจะทำการคำนวณแรงและแรงบิดที่เกิดขึ้นกับวัตถุ และทำการตรวจสอบว่าวัตถุเสมือนนั้นได้กระทำกับนิ้วเสมือนหรือเปล่า หากใช่ให้ทำการส่งแรงที่คำนวณได้ไปยังมอเตอร์เพื่อสร้างความรู้สึกให้ผู้ใช้ หากเป็นการชนกันของวัตถุเสมือนกันเอง ให้ทำการเคลื่อนที่ตำแหน่ง

ของวัตถุเสมือนตามหลักการที่ได้กล่าวมา แล้วทำการแสดงผลภาพการเคลื่อนที่ ที่เกิดขึ้นบนหน้าจอ และจะทำงานเช่นนี้ไปเรื่อยๆจนกว่าจะสั่งปิดโปรแกรม ดังรูปที่ 4-2

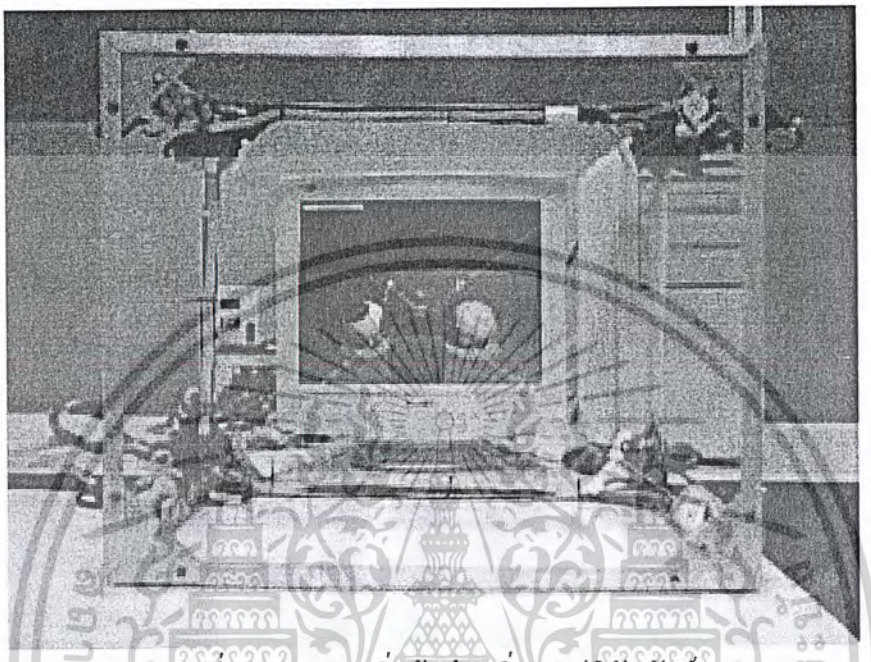
รูปที่ 4-1 แสดงการจัดค่าเริ่มต้นของโปรแกรม



รูปที่ 4-2 แสดงการทำงานของโปรแกรมในส่วนของการจำลอง

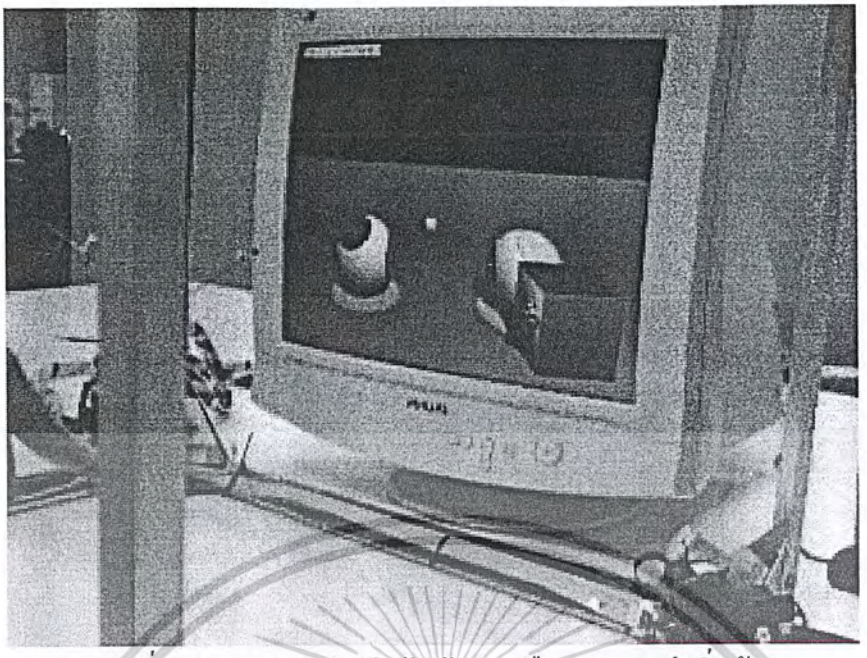
4.5 ผลการทำงานของระบบ

เมื่อเราได้พัฒนาระบบทั้งในด้านฮาร์ดแวร์และซอฟต์แวร์เสร็จเรียบร้อยแล้ว เราได้นำสองนั้นมาประกอบร่วมกันทำให้ได้ ระบบการปฏิสัมพันธ์ของวัตถุเสมือนในสภาพแวดล้อมเสมือนจริง ที่เสร็จสมบูรณ์ดังรูปที่ 4-3



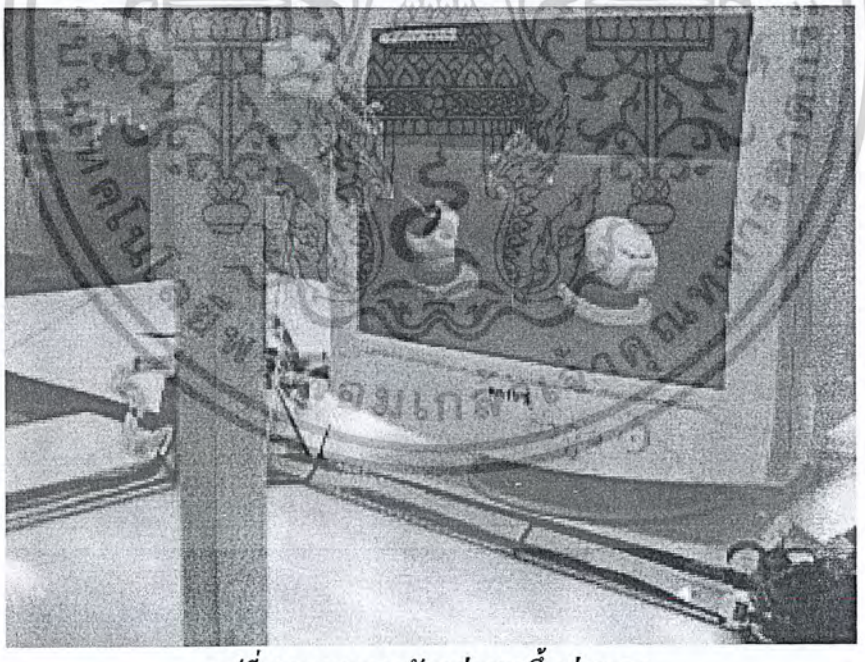
รูปที่ 4-3 แสดงการเริ่มต้นก่อนเริ่มการปฏิสัมพันธ์

ก่อนจะเริ่มทดสอบผลการดำเนินงานของระบบจะเห็นได้ว่าจะมีการสร้างวัตถุดังนี้คือ พื้นระนาบ กำแพงด้านหลัง กำแพงด้านข้างสองด้านซึ่งมองไม่เห็น ลูกบอลทรงกลมสองลูก กล่องทรงสี่เหลี่ยม ลูกบาศก์และสุดท้ายคือลูกบอลเล็กสองลูกซึ่งใช้ในการบอกตำแหน่งของนิ้วทั้งสอง ต่อไปจะเริ่มทำการปฏิสัมพันธ์โดยเริ่มจากการย้ายตำแหน่งของกล่อง โดยการใช้ผลักจากตำแหน่งปัจจุบันไปด้านขวามือดังแสดงในรูปที่ 4-4 ในรูปนั้นได้แสดงถึงการที่ทรงกลมซึ่งบอกตำแหน่งของนิ้วในโลกเสมือนไปสัมผัสกับกล่องซึ่งแรงในการผลักนั้นเกิดจากการเคลื่อนที่ด้วยความเร็วค่าหนึ่งแล้วไปปะทะกับกล่องจากนั้นก็ทำการหาแรงที่เกิดจากการปะทะโดยหาว่าทรงกลมนั้นปะทะเข้าไปยังเนื้อของกล่องเป็นระยะทางเท่าใด จากนั้นก็จะทำการส่งแรงกลับไปยังมอเตอร์

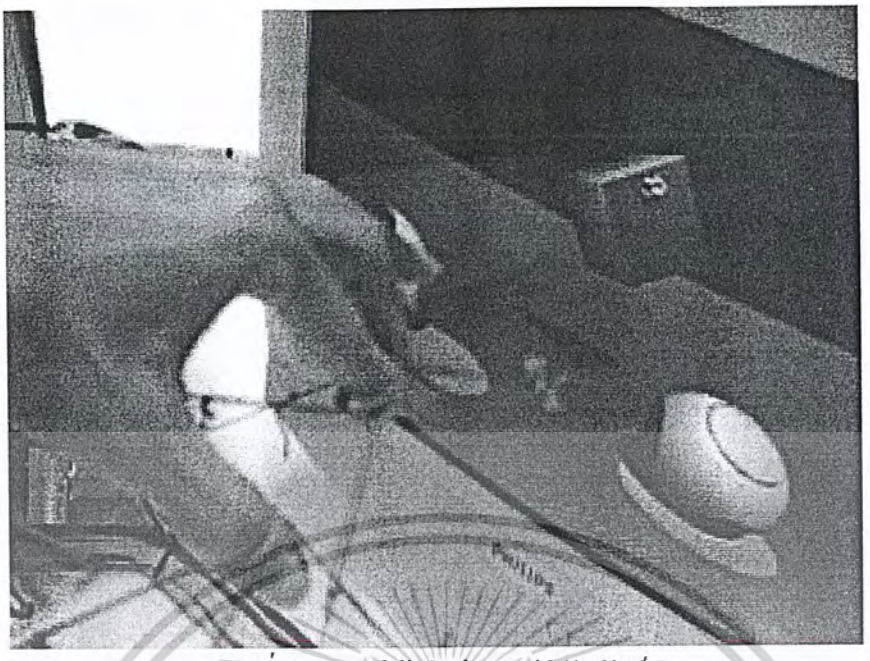


รูปที่ 4-4 แสดงการผลักกล่องไปด้านขวามือจากตำแหน่งเริ่มต้น

ต่อไปจะเริ่มทำการจับเพื่อยกกล่องขึ้นพร้อมๆกับการบิดกล่องเพื่อแสดงให้เห็นถึงการเปลี่ยนสถานะของกล่องดังแสดงในรูปที่ 4-5 และรูปที่ 4-6



รูปที่ 4-5 แสดงการจับกล่องยกขึ้นสู่อากาศ



รูปที่ 4-6 แสดงให้เห็นถึงการปฏิสัมพันธ์

ต่อไปจะทำการทดสอบการปะทะระหว่างกล่องกับทรงกลม และ กล่องปะทะกับระนาบพื้นจะเห็นได้ถึงการเคลื่อนไหวที่เสมือนจริงหลังจากปล่อยกล่องให้ตกกระทบกับทรงกลมทำให้เกิดแรงปะทะขึ้นทั้งที่กล่องและลูกบอลทำให้เกิดแรงผลักดันเกิดขึ้นและด้วยเพราะมีแรงโน้มถ่วงทำให้กล่องนั้นยังคงตกลงกระทบกับพื้นระนาบอีกด้วยดังแสดงในรูปที่ 4-7

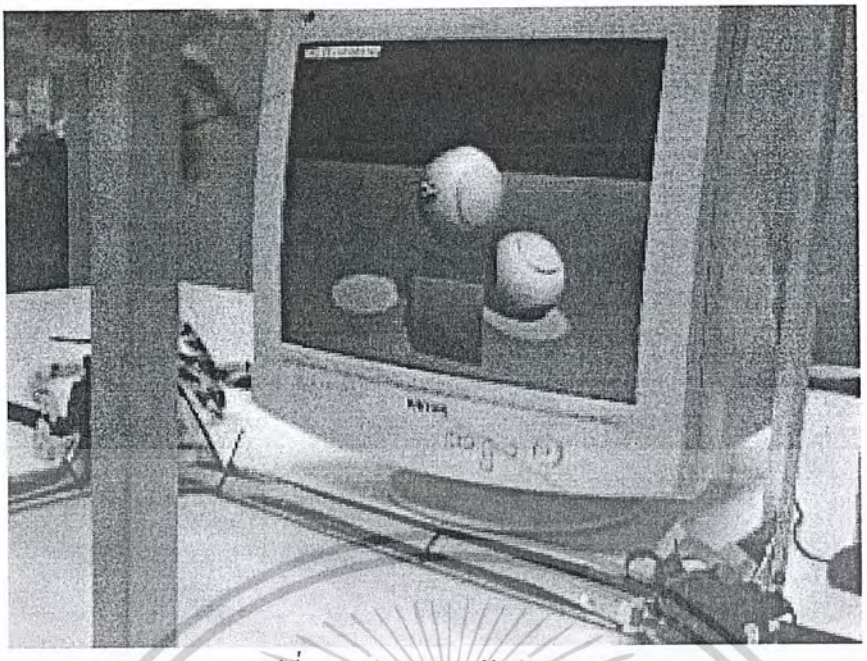
รูปที่ 4-7 เมื่อเราปล่อยกล่องให้ตกปะทะกับทรงกลม

หลังจากนั้นจะเริ่มการทดสอบต่อไป โดยจะเริ่มทำการทดสอบกับวัตถุทรงกลมโดยเริ่มจากการผลักจากตำแหน่งที่ลูกบอลอยู่เดิมคืออยู่ในพื้นที่วงกลมสีเทาที่พื้น โดยการผลักวัตถุทรงกลมไปด้านหลังดังแสดงในภาพที่ 4-8

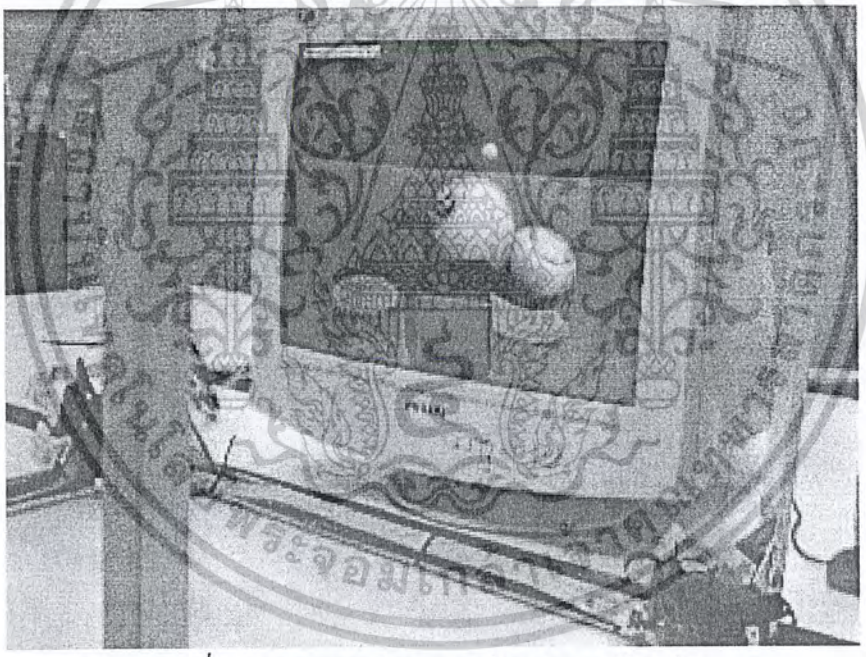
รูปที่ 4-8 แสดงการปฏิสัมพันธ์กับทรงกลมโดยการผลักดัน

เมื่อได้ทำการทดสอบการผลักเสร็จเรียบร้อยแล้ว จะทำการจับยกวัตถุทรงกลมโดยเริ่มจากการจัดตำแหน่งสภาพแวดล้อมใหม่ให้วัตถุอยู่ในตำแหน่งเริ่มต้นใหม่ทั้งหมด เพื่อที่จะได้เห็นการเปลี่ยนแปลงเมื่อเทียบกับภาพแรกในการทดลอง และเพื่อจะได้เป็นการง่ายในการมองเห็นภาพเปรียบเทียบในขั้นตอนต่อๆไป ดังแสดงในภาพที่ 4-9 จะเห็นได้ว่าจะทำการทดสอบโดยยกวัตถุทรงกลมแล้วเคลื่อนย้ายพร้อมกับการบิดทรงกลมไปพร้อมๆกันในอากาศ เพื่อทดสอบสถานะของวัตถุทรงกลมแล้วย้ายตำแหน่งของวัตถุทรงกลมมายังตำแหน่งเหนือกล่องเพื่อที่จะใช้ในการทดสอบขั้นต่อไป

ขั้นต่อมาจะทำการปล่อยให้ทรงกลมตกลงกระทบกับกล่อง เพื่อจะทำการทดสอบการปะทะระหว่างทรงกลมกับกล่องและทรงกลมกับระนาบพื้นดังแสดงในรูปที่ 4-10 จะสังเกตได้ว่าจะเห็นถึงแรงที่กระทำระหว่างสองวัตถุทำให้ทรงกลมผลักกล่องมาด้านหน้า แล้ววัตถุทรงกลมตกลงไปด้านหลังของกล่องแล้วจะตกลงมากระทบกับพื้นอีกครั้งหนึ่ง ซึ่งแสดงถึงการสามารถตรวจสอบการปะทะได้ทั้งสองกรณี



รูปที่ 4-9 แสดงการยกวัตถุทรงกลม



รูปที่ 4-10 แสดงการปะทะระหว่างทรงกลมกับกล่อง

บทที่ 5
สรุป

5.1 สรุปการทำงาน

การสร้างฮาร์ดแวร์ SPIDAR นั้นได้เริ่มจากการสร้างโครงอลูมิเนียมโดยให้มีขนาด 80 x 60 x 60 ซม. ซึ่งต้องให้อลูมิเนียมตัดเป็นพีระมิดไปยึดที่มุมทั้งแปดเพื่อใช้ในการติดมอเตอร์เข้าไป จากนั้นก็ทำการบัคกรีสายสัญญาณและสายไฟที่ใช้เชื่อมต่อระหว่างตัว SPIDAR กับแอมปริไฟล์ และระหว่าง แอมปริไฟล์ไปยังการ์ดอินเตอร์เฟสที่เสียบอยู่กับเครื่องคอมพิวเตอร์ จากนั้นก็ได้ทำการขึงเชือกโดยใช้ท่อพีวีซีตัดเพื่อใช้เป็นแหวนแล้วเจอรูทั้งหมด แปดรู จำนวนสองวง จากนั้นก็ทำการยึดแหวนกับลูกรอกของมอเตอร์แต่ละตัวตามทฤษฎีที่กล่าวไว้ในเบื้องต้นเมื่อสำเร็จแล้วเราก็ได้ทำการทดสอบการทำงาน โดยทำการติดตั้งไดรเวอร์ของการ์ดอินเตอร์เฟสทั้งสามการ์ดคือเคาเตอร์สองการ์ดและดิจิตอลไปยังแอนนาลอกอีกหนึ่งการ์ดโดยให้การ์ดเคาเตอร์ตัวที่หนึ่งกำหนดอยู่ที่แอดเดรสที่ 0x304 และตัวที่สองอยู่ที่ 0x308 และติดตั้งการ์ดดิจิตอลไปยังแอนนาลอกไว้ที่แอดเดรส 0x300 จากนั้นได้ใช้โปรแกรมที่ติดมากับไดรเวอร์เพื่อใช้ในการตรวจสอบการทำงานของการ์ดว่าสามารถอ่านค่าหรือส่งค่าได้ตามต้องการหรือไม่ เมื่อทดสอบแล้วว่าสามารถทำงานได้ถูกต้องจึงได้ไปพัฒนาส่วนที่เป็นด้านซอฟต์แวร์ต่อไป

การสร้างส่วนของซอฟแวร์นั้นเริ่มจากการได้ศึกษาการสร้างวัตถุสามมิติ โดยใช้ไดเรคเอ็กซ์ หลังจากนั้นได้ทำการศึกษาทฤษฎีการสร้างวัตถุที่มีคุณสมบัติทางฟิสิกส์ พร้อมๆกับการศึกษาโค้ดโปรแกรมที่ได้จากไลบารี่ที่มีการสร้างวัตถุที่มีคุณสมบัติทางฟิสิกส์ จากนั้นก็ได้ทำการศึกษาทฤษฎีการทดสอบการชนตั้งแต่วิธีที่ง่ายๆ จนถึงวิธีที่ยากขึ้นแล้วเปรียบเทียบความเหมาะสมในการนำมาใช้ในการทำโปรเจคนี้จากนั้นก็ได้ศึกษาผลกระทบจากการชน (Collision Response) ซึ่งความจริงแล้วเป็นส่วนหนึ่งของคุณสมบัติของวัตถุทางฟิสิกส์ซึ่งจะคำนวณหาแรงและค่าอื่นๆในการเคลื่อนที่ของวัตถุ เมื่อสามารถทำความเข้าใจได้บ้างแล้วจึงหันมาศึกษาการติดต่อกับการ์ดโดยอาศัยไลบารี่ที่มีอยู่ พร้อมกับการสร้างระบบจำลองจากไลบารี่โดยสร้างเป็นคลาสครอบลงไปอีกทีหนึ่งรวมถึงคลาสที่ใช้ในการตรวจสอบการชนเราได้อาศัยไลบารี่ในการสร้างตัวแปรบางชนิดขึ้นมา แต่วิธีการตรวจสอบนั้นสามารถสร้างขึ้นได้เองโดยอาศัยหลักการง่ายๆเบื้องต้นก็สามารถใช้ในการตรวจสอบได้แล้วจึงสร้างระบบทั้งหมดที่เหลือ เมื่อด้านซอฟต์แวร์สำเร็จสมบูรณ์จึงเริ่มทำการตรวจสอบการติดต่อกับอุปกรณ์ SPIDAR เพื่อทดสอบการทำงานทั้งอีกครั้งหนึ่งเพื่อหาข้อผิดพลาดและแก้ไขต่อจนสำเร็จ

5.2 ปัญหาและอุปสรรค

- หากเรากำหนดให้โลกจำลองมีความซับซ้อนมากเท่าไหร่จะยิ่งมีผลทำให้การตอบสนองในส่วนของฮาร์ดแวร์ทำงานช้าลงตามเนื่องจากส่วนฮาร์ดแวร์อินเตอร์เฟสนั้นได้ใช้การ์ดแบบ ISA ทำให้ต้องใช้ PC รุ่นที่ค่อนข้างเก่าเพราะใน PC รุ่นใหม่ๆจะไม่มีสล๊อต ISA แล้ว

- ไม่สามารถรองรับระบบปฏิบัติการ Windows 2000 หรือ XP ได้เนื่องจากระบบปฏิบัติการดังกล่าวไม่อนุญาตให้ผู้ใช้ทำการอ้างตำแหน่งของ I/O ได้โดยตรงจะต้องทำผ่าน ดีไวซ์ไดรเวอร์เท่านั้น จึงจำเป็นที่จะต้องกลับมาทำที่ระบบ Windows 98 ที่ยังอนุญาตให้ใช้ I/O ได้โดยตรง
- ความสัมพันธ์ของการแสดงผลกับการอ่านตำแหน่งของนิ้วมือจาก SPIDAR ไม่สอดคล้องกันโดยที่อัตราการแสดงผลจะอยู่ที่ 20-30 Hz แต่อัตราการอ่านข้อมูลของ SPIDAR จะอยู่ที่ 1 kHz ส่งผลให้ส่วนแสดงผลตามไม่ทันการอ่านข้อมูล ทำให้การอ้างอิงตำแหน่งระหว่างนิ้วเสมือนกับนิ้วจริงของผู้ใช้ผิดไป เช่น หากมีการเคลื่อนที่ของนิ้วมือที่ไวมากๆ จะทำให้นิ้วเสมือนเคลื่อนที่ตามไม่ทันส่งผลให้การจับสัมผัสวัตถุผิดไปด้วย
- แรงต้านกลับที่ฮาร์ดแวร์มีการกระตุกบ้างในบางจังหวะอันเนื่องมาจากการที่อัลกอริทึ่มที่ทำการหาอยู่ว่าตอนนี้ต้องใช้ force volume ของพีระมิดอันใดใน SPIDAR ไม่สอดคล้องกับการที่ตัว SPIDAR เองนั้นทำงานที่เร็วกว่าจึงทำให้ในขณะที่มีการประมวลผลแล้วส่งค่ากลับไปให้ยังการแสดงผลนั้นในรอบการทำงานใหม่ที่ตัว SPIDAR ก็จะมีการทำงานใหม่อีกรอบทำให้การแสดงผลแล้วส่งค่าแรงกลับมาที่ตัว SPIDAR เกิดความไม่สอดคล้องกันทำให้เกิดการกระตุกที่เส้นเชือกที่มีแรงกระทำที่เส้นเชือกไม่ถูกต้องในบ้างครั้ง วิธีแก้ไขในเบื้องต้นคือการลดแรงสูงสุดและต่ำสุดที่จะเกิดกับเส้นเชือกให้มีค่าน้อยลง

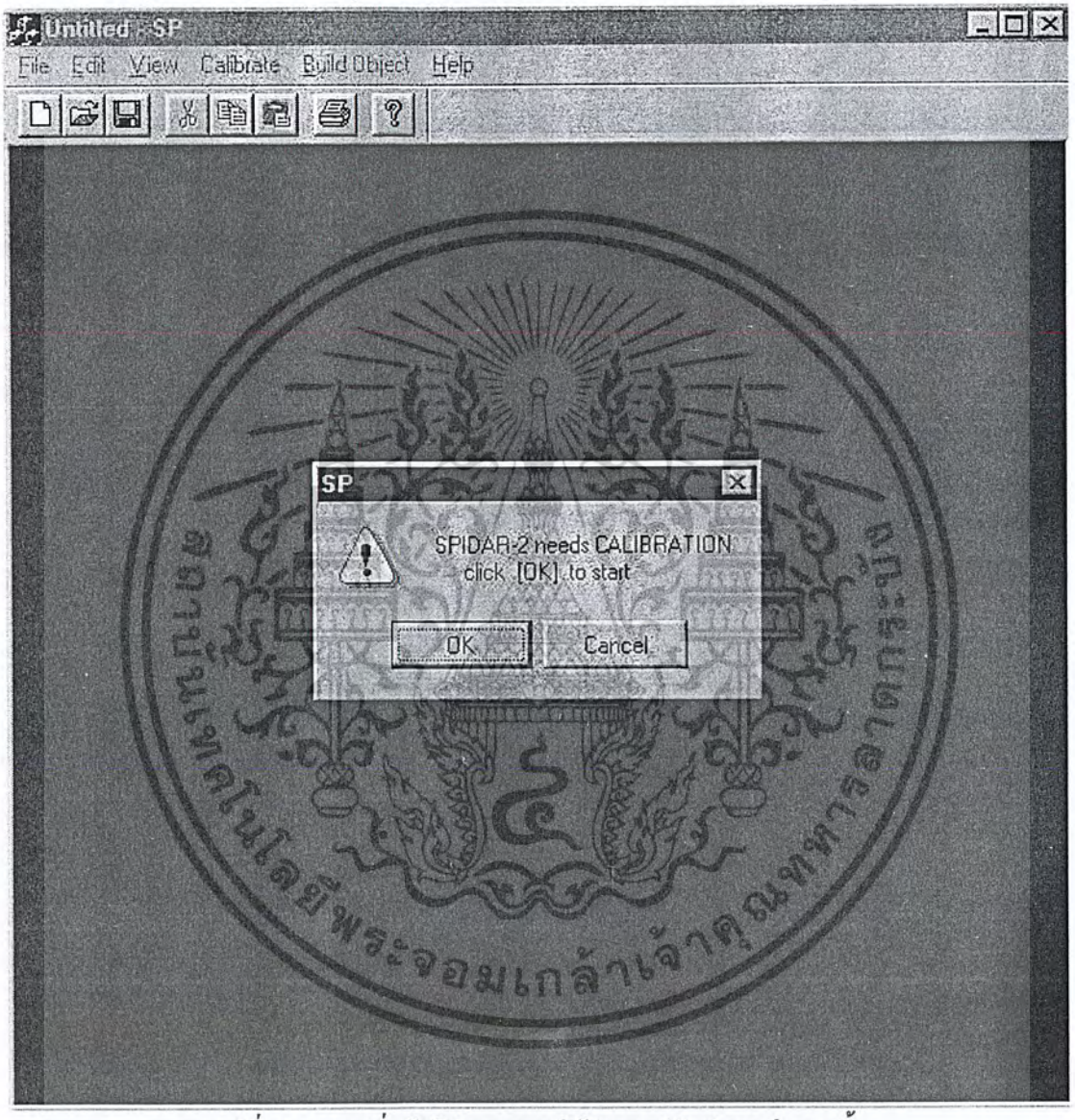
5.3 แนวทางในการพัฒนาต่อ

- พัฒนาส่วนแสดงผล 3 มิติให้วัตถุมีความหลากหลายมากขึ้น
- พัฒนาในส่วนการแสดงผลทางเสียงเมื่อเกิดการชนกันหรือการเคลื่อนที่ของวัตถุ
- พัฒนาตัวการ์ดอินเตอร์เฟสให้เป็นแบบ PCI หรือออกแบบตัวการ์ดใหม่แล้วติดต่อผ่านพอร์ต USB v2.0 พร้อมทั้งพัฒนาดีไวซ์ไดรเวอร์เพื่อที่จะสามารถใช้ใน PC รุ่นใหม่ๆ บนระบบปฏิบัติการที่ทันสมัยมากยิ่งขึ้นได้จะส่งผลให้การทำงานของการจำลองระบบดีขึ้น

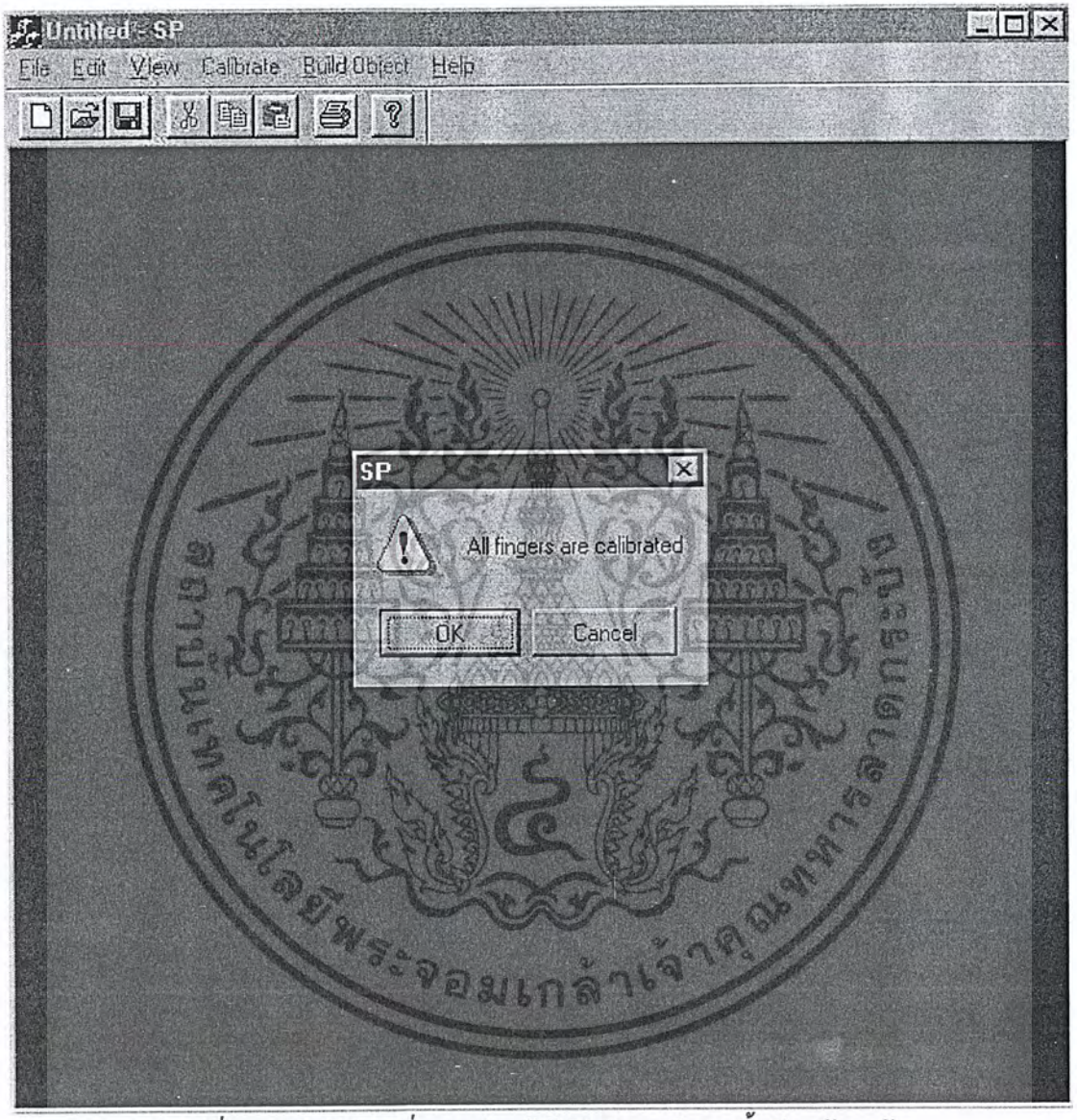
ภาคผนวก ก.

ขั้นตอนการใช้โปรแกรม

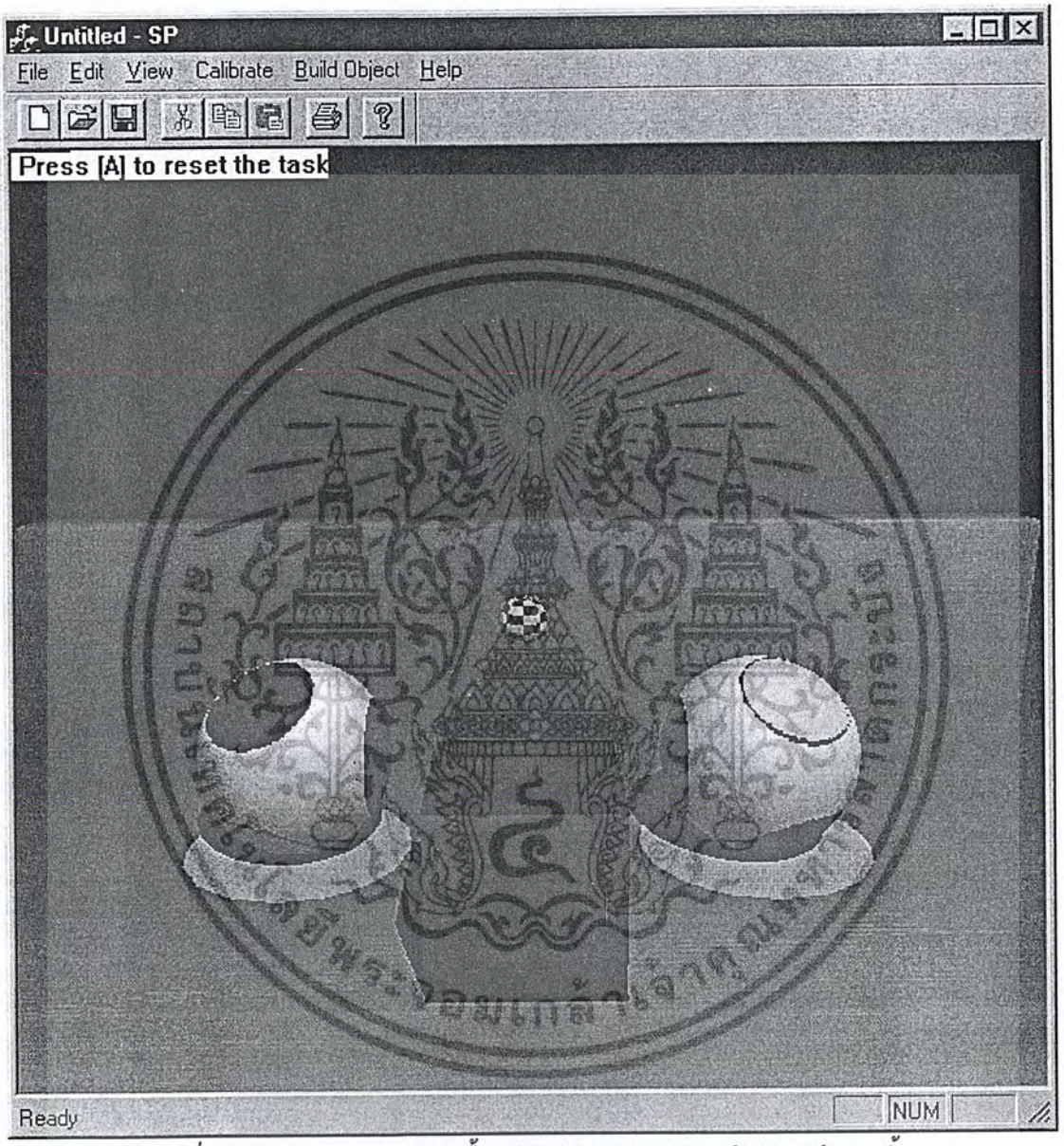
1. เริ่มต้นจะต้องมีการปรับแต่งตำแหน่งของนิ้วบนอุปกรณ์ SPIDAR โดยจะต้องเคลื่อนตำแหน่งของนิ้วทั้งสองมาที่จุดกึ่งกลางเสียก่อนดังรูปที่ ก-1 ถึง ก-4



รูปที่ ก-1 แสดงเริ่มต้นโปรแกรมจะให้ทำการปรับตำแหน่งของนิ้ว

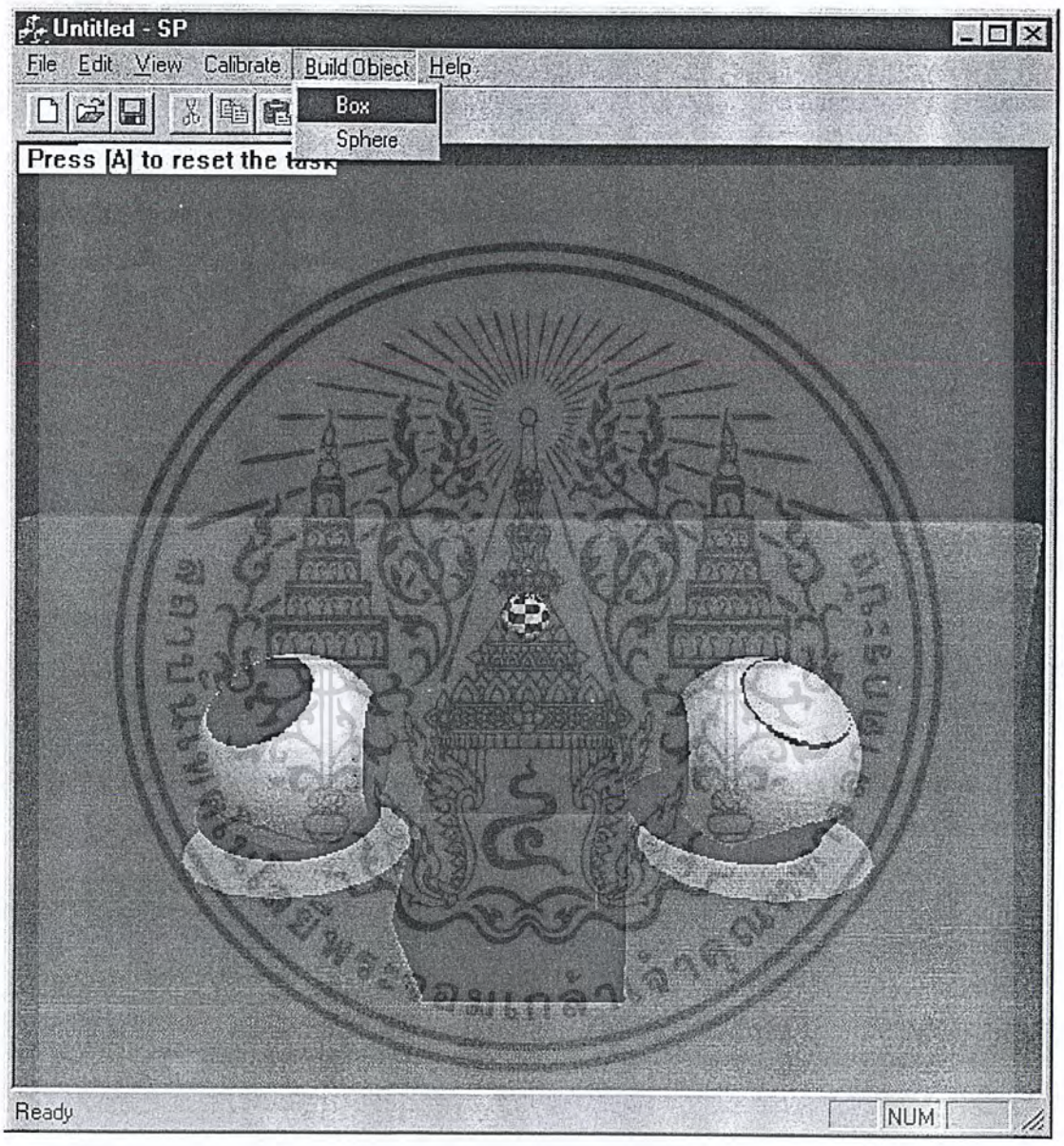


รูปที่ ก-2 กล่องโต้ตอบที่แสดงว่าเราได้ปรับตำแหน่งของนิ้วเรียบร้อยแล้ว



รูปที่ ก-3 แสดงสภาพแวดล้อมเบื้องต้นหลังจากการปรับแต่งตำแหน่งของนิ้ว

2. เมื่อปรับแต่งตำแหน่งของนิ้วเรียบร้อยต่อไปจะอธิบายการใช้งานเมนู ในโปรแกรมจะมีเมนูเพียงสามเมนูคือการสร้างวัตถุทั้งสองแบบ และเมนูการปรับแต่งตำแหน่งนิ้วในขณะที่ดำเนินโปรแกรมอยู่ ดังรูปที่ ก-6



รูปที่ ก-4 แสดงเมนูที่มีในโปรแกรม

บรรณานุกรม

[1] David Baraff. An Introduction to Physically Based Modeling:Rigid Body Simulation I Unconstrained Rigid Body Dynamics, chapter Rigid Body Dynamic. SIGGRAPH, 1997.

[2] นิรุธ อำนวยศิลป์: "เขียนเกมส์อย่างมืออาชีพด้วย Visual C++ และ DirectX ฉบับสมบูรณ์",อินโฟเพรส. พิมพ์ครั้งที่ 1 ปี 2545

[3] ชัยดำรงค์ อุทิรัมย์: "Direct3D พลังพัฒนาแห่งเกมส์สามมิติ", สามย่าน.COM พิมพ์ครั้งที่ 1 ปี 2544

[4] Ishii M., And Sato M.,: "A 3D Interface Device with Force Feedback: A Virtual Work Space for Pick-and-Place Task", Proceeding of the 1993 IEEE VRAIS, PP. 331-335.