เกมต่อสู้เสมือนจริง

Virtual Fighter

นายเวชอมร จรูญผลฐิติ
นายศักดิ์ชัย เตียวศิริ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2547

เกมต่อสู้เสมือนจริง

Virtual Fighter

โดย

นายเวชอมร จรูญผลฐิติ

นายศักดิ์ชัย เตียวศิริ

อาจารย์ที่ปรึกษา

ดร. สมศักดิ์ วลัยรัชต์

อ. สมเกียรติ วังศิริพิทักษ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2547

ปริญญานิพนธ์ปีการศึกษา 2547

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง เกมต่อสู้เสมือนจริง

VIRTUAL FIGHTER

ผู้จัดทำ

นายเวชอมร จรูญผลฐิติ รหัสประจำตัว 44010476

นายศักดิ์ชัย เตียวศิริ รหัสประจำตัว 44010481

_____อาจารย์ที่ปรึกษา

(ดร. สมศักดิ์ วลัยรัชต์)

_____อาจารย์ที่ปรึกษา

(อ. สมเกียรติ วังศิริพิทักษ์)

เกมต่อสู้เสมือนจริง

นายเวชอมร จรูญผลฐิติ 44010476
นายศักดิ์ชัย เตียวศิริ 44010480
ดร. สมศักดิ์ วลัยรัชต์ อาจารย์ที่ปรึกษา
อ. สมเกียรติ วังศิริพิทักษ์ อาจารย์ที่ปรึกษา

บทคัดย่อ

โครงการนี้เป็นการสร้างเกมจำลองการต่อสู้เสมือนจริงระหว่างผู้เล่นกับเครื่องคอมพิวเตอร์ ในส่วนของผู้เล่นจะใช้ ระบบการจับการเคลื่อนไหวของผู้เล่นผ่านกล้องวีดีโอ แล้วนำภาพจากกล้องวีดีโอมาวิเคราะห์หาตำแหน่งของร่างกายบนระบบพิกัด 3 มิติ(vision-based real time motion capture system) เพื่อตรวจจับการเคลื่อนไหวร่างกายส่วนบนของผู้เล่น และนำข้อมูลที่ตรวจจับได้ไปควบคุมตัวละครในเกมให้เคลื่อนไหวตามที่ต้องการ ทำให้ผู้เล่นสามารถควบคุมร่างกายส่วนบนเพื่อจู่โจมและหลบหลีกการจู่โจมจากฝ่ายตรงข้ามได้อย่างอิสระ สำหรับการเคลื่อนที่ร่างกายส่วนล่างจะใช้เซนเซอร์จับการเคลื่อนที่ที่บริเวณขาของผู้เล่น แล้วนำไปแปลงเป็นข้อมูลการเคลื่อนที่ของตัวละคร ในส่วนของคู่ต่อสู้ในเกม ซึ่งก็คือเครื่องคอมพิวเตอร์ จะได้รับการโปรแกรมให้สามารถตัดสินใจด้วยตนเองว่าควรป้องกัน หลบหลีกหรือโจมตีอย่างไร โดยในการเล่นเกมนั้นเมื่อผู้เล่นถูกโจมตีจะมีการตอบสนองไปยังตัวผู้เล่น ณ จุดที่ถูกโจมตี ซึ่งทำให้ผู้เล่นรู้สึกทันทีว่าได้ถูกโจมตีตรงจุดใด

Virtual Fighter

Vejamorn Charoonponthiti
Sakchai Teaosiri
Dr. Somsak WalaiRacht Advisor
Somkiat Wangsiripitak Advisor

Abstract

This project is the virtual fighting game that player can interact with the enemy in computer. System will capture player's motion by using video cameras. Then image processing will analyze this data to find the real position of player's body part in 3D world coordinate by real-time. These positions will send to game engine as input to control the character's posture, character is instance of player in virtual world. So player can fight with the enemy by acting his body. For movement, system will get these inputs from sensors that detect player's low body movement and send it to control movement of player in game. In part of enemy, there have Artificial Intelligence module that control enemy action, such as attack, defense or movement. During fighting, player will receive force feedback when he attack or was attacked by enemy.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงได้อย่างดี ด้วยคำแนะนำ คำปรึกษาและคอยดูแลจากหลายๆ ฝ่ายด้วยกัน โดยเฉพาะอาจารย์ที่ปรึกษาทั้งสองท่านที่ให้โอกาสให้ข้าพเจ้าได้ทำปริญญานิพนธ์ฉบับนี้ คอยให้ความเอาใจใส่ แนะนำและความช่วยเหลือเสมอมา คือ ดร. สมศักดิ์ วลัยรัชต์และ อาจารย์สมเกียรติ วังศิริพิทักษ์ ซึ่งต้องขอขอบพระคุณเป็นอย่างสูง

ขอขอบคุณภาควิชาวิศวกรรมคอมพิวเตอร์ที่ได้จัดเตรียมสิ่งอำนวยความสะดวก เพื่อให้การวิจัยและพัฒนาโปรแกรมเป็นไปได้ด้วยความสะดวกและรวดเร็ว รวมทั้งยังมีอินเทอร์เน็ตความเร็วสูงให้บริการ สำหรับค้นคว้าหาความรู้ต่างๆ ซึ่งท้ายที่สุดแล้วก็ประกอบกันเป็นส่วนหนึ่งของโครงงานนี้

และสุดท้ายต้องขอขอบคุณบุคคลที่สำคัญที่สุดในชีวิตที่ทำให้ข้าพเจ้ามีวันนี้ นั่นคือ บิดา มารดา และบุคคลในครอบครัว อันเป็นที่เคารพรัก ซึ่งได้เลี้ยงดู คอยสั่งสอนข้าพเจ้ามาเป็นอย่างดี พร้อมให้โอกาสในการศึกษาอย่างเต็มที่ และยังให้กำลังใจ ความรักเสมอมา ข้าพเจ้าขอกราบพระคุณมา ณ ที่นี้ด้วย

เวชอมร จรูญผลฐิติ
ศักดิ์ชัย เตียวศิริ

สารบัญ

สารบัญตาราง

สารบัญรูปภาพ

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

การเล่นเกมต่อสู้ในปัจจุบัน ผู้เล่นจะต้องบังคับตัวละครผ่าน แป้นพิมพ์ เมาส์ หรือคันบังคับ ทำให้ผู้เล่นมีปฏิสัมพันธ์กับเกมด้วยมือและนิ้วที่ใช้บังคับและ หูที่ฟังเสียงการต่อสู้เท่านั้น โครงงานนี้จึงพยายามสร้างมิติใหม่ของการเล่นเกมต่อสู้โดยแปลงการเคลื่อนไหวของผู้เล่นไปเป็นการเคลื่อนไหวของตัวละคร ผู้เล่นสามารถออกหมัดใส่ศัตรูหรือโยกหลบหมัดของศัตรูได้ เคลื่อนที่เข้าหาหรือหลบหนีศัตรูได้ อีกทั้งยังมีแรงตอบสนองกลับไปยังผู้เล่นเมื่อคู่ต่อสู้โจมตีถูกตัวละครของผู้เล่น ซึ่งจะทำให้ผู้เล่นรู้สึกเสมือนว่าได้เข้าไปอยู่ในเกมต่อสู้จริงๆ

นอกจากนี้ความรู้ที่ได้จากการจับการเคลื่อนไหวในเกมที่พัฒนาขึ้น ยังสามารถนำไปประยุกต์ใช้กับงานอื่นๆ ได้ เช่น ข้อมูลที่ได้จากการจับการเคลื่อนไหวของมนุษย์สามารถนำไปใช้ในงานแอนิเมชันเป็นต้น

1.2 วัตถุประสงค์ของโครงงาน

- สามารถสร้างเกมที่ทำให้ผู้เล่นรู้สึกเสมือนว่าได้เข้าไปอยู่ในเกมจริงๆ
- เพื่อศึกษาการจับการเคลื่อนไหวของผู้เล่น
- เพื่อศึกษาเทคโนโลยีการสร้างเกม 3 มิติโดยใช้ DirectX
- เพื่อเรียนรู้ขั้นตอนต่างๆ ของการเขียนและพัฒนาซอฟแวร์ ตั้งแต่การวิเคราะห์ความต้องการ การออกแบบ การเขียนโปรมแกรม และการทดสอบ
- เพื่อส่งเสริมการเล่นเกมไปพร้อมๆ กับการออกกำลังกาย

1.3 ขอบเขตของในการดำเนินงาน

โครงงานนี้มีเป้าหมายเพื่อพัฒนาเกมจำลองการต่อสู้เสมือนจริง ที่ผู้เล่นสามารถบังคับการเคลื่อนไหวและเคลื่อนที่ของตัวละคร โดยใช้การเคลื่อนไหวของร่างกายผู้เล่นจริงๆ และมีการตอบสนองกลับไปยังร่างกายของผู้เล่น บนส่วนของร่างกายที่ถูกโจมตี ในที่นี้เทคโนโลยีการจับการเคลื่อนไหวที่ใช้เป็นแบบวิธีที่ใช้กล้องถ่ายภาพวิดีโอในการจับการเคลื่อนไหว (Optical Motion Capture System) ดังนั้นที่ตัวผู้เล่นจะต้องมีการติดมาร์กเกอร์ตามจุดต่างๆ เพื่อแสดงตำแหน่งของร่างกายส่วนบน และมีการติดตั้งฐานเหยียบที่พื้นเพื่อตรวจจับการเคลื่อนไหวของร่างกายส่วนล่าง นอกจากนี้ยังต้องมีอุปกรณ์กระตุ้นตามจุดต่างๆ ของร่างกายเพื่อสร้าง แรงตอบสนองที่เกิดจากการปะทะของคู่ต่อสู้ ในส่วนของคู่ต่อสู้ ได้แก่ คอมพิวเตอร์ที่ได้รับการโปรแกรมให้สามารถต่อสู้ตามสภาวะแวดล้อมที่เกิดขึ้นได้อย่างเหมาะสม

1.4 วิธีการดำเนินงาน

1.4.1 ศึกษาทฤษฎีด้านการประมวลผลภาพ(Image Processing) เพื่อนำมาวิเคราะห์หาตำแหน่งมาร์กเกอร์

1.4.2 ศึกษาภาษาที่จะนำมาใช้ในการจับการเคลื่อนไหว

1.4.3 ศึกษาโปรแกรมประยุกต์ 3 มิติ เช่น ทรีดีสตูดิโอ แม็กซ์ และ มิลค์เชฟ

1.4.4 ศึกษา DirectX9.0 SDK เพื่อนำมาใช้ในการพัฒนาเกม

1.4.5 ศึกษาฮาร์ดแวร์เพื่อนำมาตรวจจับการเคลื่อนไหวของร่างกายส่วนล่าง

1.4.6 จัดหาวัสดุ อุปกรณ์ที่จำเป็นในการพัฒนา

1.4.7 พัฒนาโปรแกรมในส่วนตรวจจับการเคลื่อนไหวของผู้เล่น

1.4.8 พัฒนาโปรแกรมในส่วนเกม

1.4.9 พัฒนาฮาร์ดแวร์เพื่อนำมาตรวจจับการเคลื่อนไหวของร่างกายส่วนล่าง

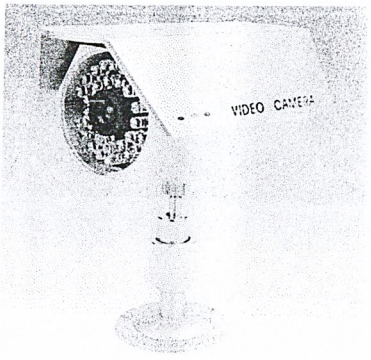
1.4.10 วิเคราะห์ระบบทั้งหมดเพื่อตรวจหาและแก้ไขข้อผิดพลาด

1.5 ส่วนประกอบของโครงงาน / อุปกรณ์ที่ใช้

1.5.1 กล้องถ่ายภาพวิดีโอชนิด CCD 2 ตัว ยี่ห้อ LYD รุ่น 805C ผลิตในประเทศจีน เป็นกล้องแบบ day-night คือถ่ายภาพได้ทั้งกลางวันและกลางคืนโดยใช้แสงอินฟราเรดจากหลอด LED ที่ติดตั้งอยู่รอบๆ ตัวเลนส์ โดยมีคุณสมบัติดังนี้

รายการ	รายละเอียด
Image Device	¼ inch Sharp CCD
Image Sensor Size	3.2 x 2.4 mm.
Focal Length	5 mm. / F2.0
Picture Element	PAL: 512x582, NTSC: 512x492
Horizontal Definition	380 lines
Scan Frequency	50 Hz (PAL) / 60 Hz (NTSC)
Lense Type	Day / Night (Infrared)
Infrared Effective Range	10 m. (CDS Auto Control)
Function	AWB, AGC, AES
Video Output	$1V_{p-p}$ / 75 Ohm, Composite
Power Consumption	12VDC, 50-60Hz

ตารางที่ 1.1 ตารางแสดงคุณลักษณะของกล้อง LYD-805C



รูปที่ 1.1 กล้องวิดีโอยี่ห้อ LYD รุ่น 805C

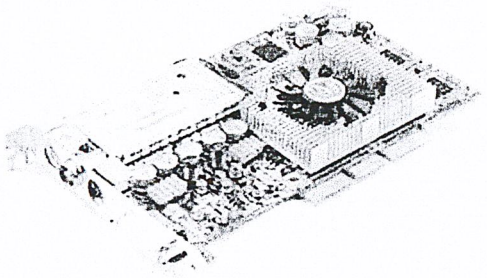
1.5.2 ขาตั้งกล้อง 2 ตัว (Tripod) ยี่ห้อ SLIK รุ่น U9000

1.5.3 เครื่องคอมพิวเตอร์ 2 ตัว และอุปกรณ์เครือข่ายสำหรับเชื่อมต่อ

1.5.4 การ์ดแสดงผลพร้อมตัวจับภาพ ยี่ห้อ ATI รุ่น Radeon All-in-wonder 9800 PRO

รายการ	รายละเอียด
GPU	RADEON 9800 PRO
Memory	128 MB 256-bit DDR SDRAM
Support DirectX	9.0
Support OpenGL	1.5
Output	DVI-I / TV-Out / Video-Out / D-Sub
Input	Video-In / TV Tuner (AV, S-Video Connector)
Bus	AGP 8X/4X/2X
Max Capture Rate	50 fps @ 320x240
Supported Video Signal	PAL, NTSC, SECAM
Image Color Output	YUV Color

ตารางที่ 1.2 ตารางแสดงคุณลักษณะของการ์ดจับภาพ



รูปที่ 1.2 การ์ดแสดงผลยี่ห้อ PowerColor รุ่น Radeon All-in-Wonder 9800 Pro

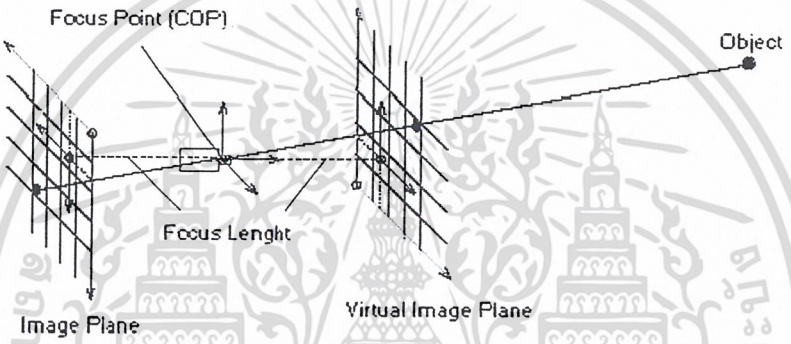
1.5.5 มาร์กเกอร์สี ซึ่งมีสีแตกต่างกัน 7 ชิ้น

บทที่ 2
การรับภาพจากกล้องถ่ายภาพวิดีโอ

2.1 การเกิดภาพบนกล้องวีดีโอ และการรับภาพจากกล้องวีดีโอ

องค์ประกอบสำคัญของการเกิดภาพบนกล้องวีดีโอ ประกอบไปด้วย

1. วัตถุ
2. ฉากรับภาพ
3. ความยาวโฟกัส



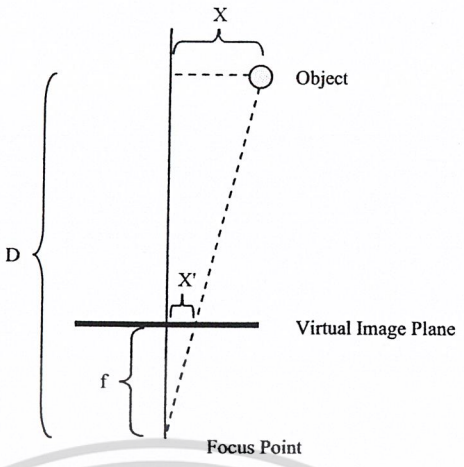
รูปที่ 2.1 ภาพจำลองการเกิดภาพบนกล้องวีดีโอ

การเกิดภาพของกล้องวีดีโอ เกิดจากแสงกระทบวัตถุ แล้วสะท้อนกลับเข้ากล้องโดยผ่านจุดโฟกัส แล้วฉายไปยังฉากรับภาพ หรือเซนเซอร์รับภาพ CCD (CCD Image Size) หลังจากนั้นกล้องวีดีโอจะทำการส่งต่อข้อมูลไปยังคอมพิวเตอร์ต่อไป

สำหรับกล้องที่ส่งสัญญาณแบบอนาล็อกไม่ว่าจะเป็นแบบใดก็ตาม (PAL, NTSC) นั้น การรับภาพเข้ามาในคอมพิวเตอร์เพื่อประมวลผล จำเป็นต้องใช้อุปกรณ์จับภาพ (Video Capture Device) ในการแปลงสัญญาณอนาล็อกไปเป็นสัญญาณดิจิตอล เพื่อนำไปใช้งานต่อไป

2.2 ความสัมพันธ์ของภาพและวัตถุ

ในการพิจารณาถึงความสัมพันธ์ระหว่างวัตถุและภาพที่เกิดบนฉากรับภาพ เราสามารถพิจารณาโดยใช้หลักการของกล้องรูเข็มแทนได้ ซึ่งเป็นการคิดความสัมพันธ์ในรูปแบบของสามเหลี่ยมคล้าย ทำให้เราสามารถคำนวณหาตำแหน่งจริงของวัตถุหรือตำแหน่งของวัตถุบนภาพได้ง่ายขึ้น แต่การนำความสัมพันธ์นี้ไปหาค่าจำเป็นที่จะต้องรู้ค่าความยาวโฟกัส และขนาดของฉากรับภาพหรือเซ็นเซอร์รับภาพในแนวตั้งและแนวนอน เพื่อหาอัตราส่วนในการคำนวณ



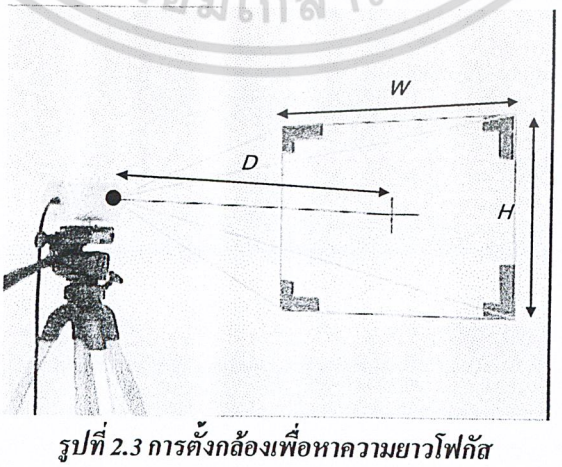
รูปที่ 2.2 การหาระยะบนภาพ

สมการความสัมพันธ์ของรูปด้านบนแสดงด้วยสมการด้านล่าง

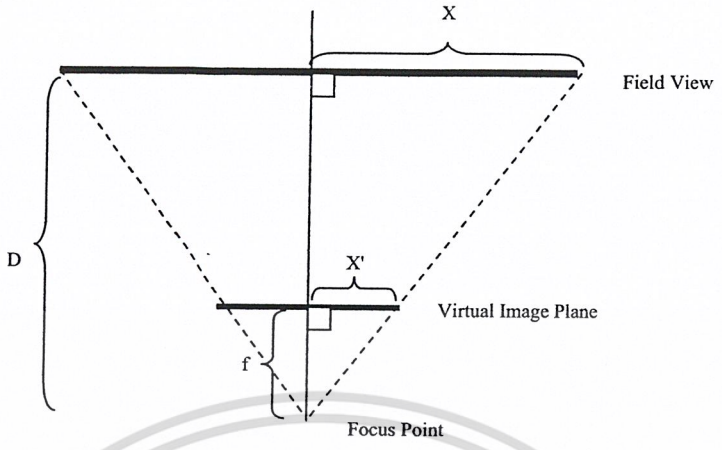
$$\frac{X'}{X} = \frac{f}{D} \qquad (2.1)$$

2.3 การคำนวณหาอัตราส่วนการเกิดภาพของกล้องถ่ายภาพ

การหาค่าอัตราส่วนการเกิดภาพอย่างง่ายที่สุด ทำได้โดยการตั้งกล้องให้ถ่ายผนัง โดยต้องตั้งกล้องให้เส้นสมมาตรของกล้องตั้งฉากกับผนังทั้ง 2 แกน ณ ระยะห่างหนึ่งที่วัดค่าได้ (D) จากนั้นให้กำหนดค่าโฟกัส (f) หรือขนาดฉากรับภาพ (X') ขึ้นมาค่าใดค่าหนึ่ง ก็จะหาอีกค่าหนึ่งได้ แต่ค่าที่ได้ยังไม่ใช่ค่าที่แท้จริง จะเป็นอัตราส่วนระหว่างระยะโฟกัสกับขนาดฉากรับภาพของกล้องเท่านั้น ซึ่งสามารถนำไปใช้ในการคำนวณได้ แต่ถ้าต้องการทราบค่าโฟกัสจริงๆ จะไม่สามารถใช้หลักการของกล้องรูเข็มได้ ต้องนำขนาดของเลนส์เข้ามาคำนวณด้วย



รูปที่ 2.3 การตั้งกล้องเพื่อหาความยาวโฟกัส



รูปที่ 2.4 การหาความยาวโฟกัส

สมการความสัมพันธ์ของรูปด้านบนแสดงด้วยสมการที่ (2.1)

2.4 วีดีโอโอซีเอ็กซ์ (VideoOCX)

วีดีโอโอซีเอ็กซ์ เป็นส่วนประกอบชนิด แอกทีฟเอ็กซ์ คอนโทรล(ActiveX Control) สำหรับระบบปฏิบัติการวินโดว์ ซึ่งมีหน้าที่รับภาพมาจากกล้องวีดีโอหรือ รับภาพจากไฟล์วีดีโอ และยังมีฟังก์ชันการใช้งานสำหรับการแสดงผล การบันทึกภาพวีดีโอได้ทั้งไฟล์ภาพ และไฟล์วีดีโอ การเข้าถึงข้อมูลในแต่ละพิกเซล (Pixel) ซึ่งฟังก์ชันการใช้งานเหล่านี้มีความเหมาะสมและง่ายต่อการใช้งาน

ความสามารถของวีดีโอโอซีเอ็กซ์ มีดังนี้

- สามารถจับภาพสีได้ที่ความเร็วสูงสุด (เช่น 25 fps ที่ขนาดภาพ 768x576 พิกเซล)
- บันทึกภาพที่จับได้ในรูปของไฟล์วีดีโอ (.AVI) ในรูปแบบการเข้ารหัสได้หลายชนิด
- บันทึกเสียงลงพร้อมกับไฟล์วีดีโอได้
- บันทึกภาพที่จับได้ลงบนดิสก์ (.bmp หรือ .jpg)
- แปลงภาพสีเป็นขาวดำได้โดยทันที
- สามารถเข้าถึงข้อมูลภาพสี และ ภาพขาวดำได้
- รองรับอุปกรณ์จับสัญญาณภาพแบบ PAL/NTSC ได้เป็นส่วนใหญ่
- สามารถจับภาพจากกล้องแบบ USB ได้ (Webcam)
- รองรับการเล่นไฟล์วีดีโอ สำหรับการเปิดไฟล์วิดีโอขึ้นมาประมวลผล
- สามารถดักจับการทำงานของเม้าส์บนวีดีโอโอซีเอ็กซ์ได้
- สามารถจัดการกับข้อผิดพลาดในตัววีดีโอโอซีเอ็กซ์ได้เอง

- สามารถนำไปใช้พัฒนาด้วยเครื่องมือการพัฒนาโปรแกรมหลากหลาย เช่น Visual C++, Visual Basic, Delphi เป็นต้น

นอกจากนี้วีดีโอโอซีเอ็กซ์ ยังมาพร้อมกับวีดีโอโอซีเอ็กซ์ทูล (VideoOCXTools) เป็นส่วนประกอบชนิด แอกทีฟเอ็กซ์ คอนโทรล ที่มีฟังก์ชัน 2 กลุ่ม คือ ฟังก์ชันในการประมวลผลภาพ ซึ่งฟังก์ชันการประมวลผลภาพเหล่านี้สามารถทำงานบนภาพขาวดำได้เท่านั้น ไม่สามารถทำงานกับภาพสีได้ และฟังก์ชันอีกกลุ่ม คือ ฟังก์ชั่นการวาดรูปภาพบนภาพวีดีโอก่อนนำมาแสดงด้วย

บทที่ 3
การประมวลผลภาพ

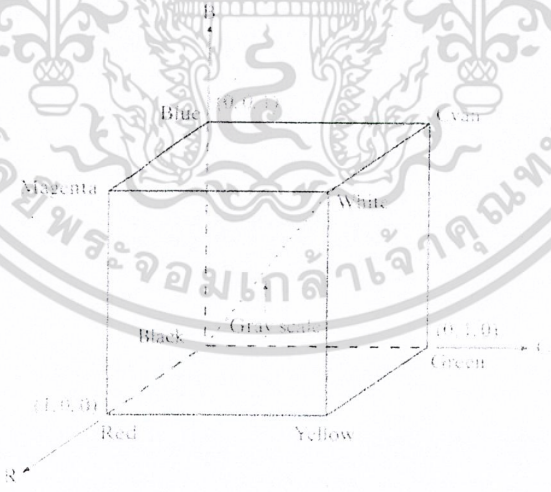
หน้าที่พื้นฐานของการประมวลผลภาพ (Image Processing) ได้แก่ การสร้างภาพใหม่โดยแยกแยะส่วนที่เราต้องการ หรือสนใจกับสิ่งรบกวนออกจากกัน (Noise elimination), การแบ่งส่วนภาพ(Image Segmentation), การหาขอบภาพ (Edge enhancement), การกรอง(Filtering), การปรับปรุงและเปลี่ยนแปลงค่าระดับเกรย์ (Gray scale modification) ซึ่งในโครงงานนี้นำหลักการการแบ่งส่วนภาพมาใช้ในการหาตำแหน่งมาร์กเกอร์

3.1 พื้นฐานและระบบของสี

ระบบสีพื้นฐานที่ใช้ในการประมวลผลภาพทั่วไปมี 2 ระบบ คือ ระบบสี RGB และระบบสี HSI

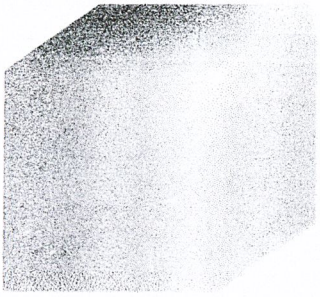
3.1.1 ระบบสี RGB

ระบบสีนี้เป็นระบบที่คอมพิวเตอร์ใช้อยู่ในปัจุบัน โดยมีสีพื้นฐาน 3 สี คือ red, green และ blue ระบบสีนี้มีลักษณะเป็นสี่เหลี่ยมลูกบาศก์ ดังรูปที่ 2.6 โดยที่สีพื้นฐาน 3 สี จะอยู่ที่มุม 3 มุม และสีที่เกิดจากการผสมกันระหว่างสีแต่ละคู่ (cyan, magenta และ yellow) จะอยู่ที่มุมที่เหลืออีก 3 มุม สีดำจะอยู่ที่จุดกำเนิด สีขาวจะอยู่ห่างจากจุดกำเนิดมากที่สุด และเส้นที่ลากเชื่อมระหว่างสีขาวและสีดำ จะเรียกว่าระดับสีเทา ค่าสี คือจุดที่อยู่บนผิว หรือ ภายในลูกบาศก์



รูปที่ 3.1 แสดงระบบสี RGB

ปัจจุบันเครื่องคอมพิวเตอร์จะกำหนดสีพื้นฐานให้มีขนาด 8 บิต ซึ่งการนำค่าเหล่านี้มาผสมกันทำให้ได้สีที่แตกต่างกัน เพราะฉะนั้นในปัจจุบันนี้คอมพิวเตอร์สามารถให้สีที่แตกต่างกันมากถึง 16 ล้านสี (2^{24})



รูปที่ 3.2 แสดงระบบสี RGB แบบ 24 บิต

3.1.2 ระบบสี HSI (หรือ HSV)

ระบบสีนี้มีค่าพื้นฐาน 3 ค่า คือ ค่าบ่งบอกว่าเป็นสีใด(Hue) , ค่าอิ่มตัวของสี (Saturation) ซึ่งบอกว่าสีเข้มหรือขาวขนาดไหน และค่าความเข้มของแสง(Intensity) ซึ่งบอกว่าสีสว่าง หรือ ดำขนาดไหน ระบบสีนี้มีลักษณะเป็นกรวย ดังรูปที่ 2.6 โดยที่ค่า Hue บอกด้วยองศาการหมุนในกรวย ค่า Saturation บอกด้วยระยะห่างจากตรงกลางกรวย และค่า Intensity บอกด้วยความสูงในกรวย

รูปที่ 3.3 แสดงระบบสี HSI

ระบบสีนี้นำมาใช้ในการทำการประมวลผลภาพได้ดีกว่า เพราะค่า Hue ค่าเดียวสามารถบอกได้ถึงความแตกต่างของค่าสีได้แล้ว

3.1.3 การแปลงค่าสีจากระบบสี RGB เป็นระบบสี HSI

ถ้าทราบค่า RGB สามารถแปลงไปเป็น HSI ได้ดังนี้

$$H = \begin{cases} \theta & if \quad B \le G \\ 360 - \theta & if \quad B > G \end{cases} \tag{3.1}$$

โดย

$$\theta = \cos^{-1}\left\{ \frac{\frac{1}{2}\left[(R-G)+(R-B)\right]}{\left[(R-G)^2+(R-B)(G-B)\right]^{1/2}} \right\} \tag{3.2}$$

$$S = 1 - \frac{3}{(R+G+B)}\left[\min(R,G,B)\right] \tag{3.3}$$

$$I = \frac{1}{3}(R+G+B) \tag{3.4}$$

3.2 การแบ่งส่วนของภาพ (Image Segmentation)

การแบ่งส่วนของภาพ เป็นกระบวนการแยกส่วนที่สนใจออกจากส่วนอื่นๆ ของภาพซึ่งสามารถแบ่งส่วนของภาพได้หลายวิธี แต่ในโครงงานมีวิธีแบ่งส่วนอยู่ 2 ขั้นตอนดังนี้

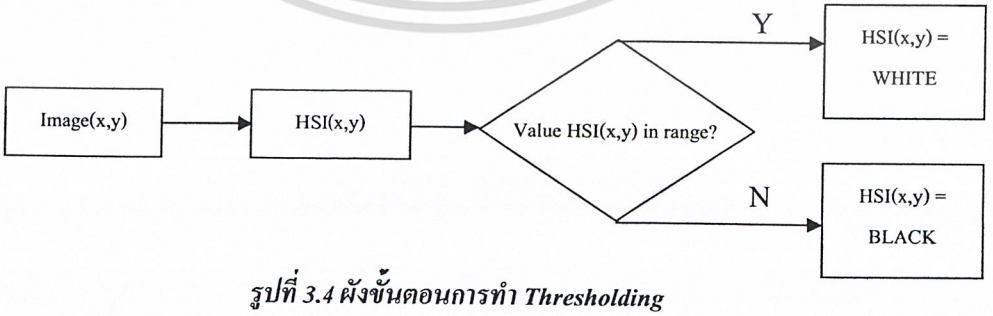
3.2.1 Thresholding

เป็นการเปลี่ยนสีของพิกเซล (Pixel) ของภาพให้เป็นสีขาวหากค่าสีอยู่ในช่วงที่สนใจ (Threshold Range) หากไม่อยู่ในช่วงสีที่สนใจให้เปลี่ยนเป็นสีดำ การทำ Thresholding นี้ จะช่วยให้สามารถแยกช่วงสีที่สนใจออกมาจากส่วนอื่นของภาพได้ โดยสีเหล่านี้จะอยู่เป็นกลุ่มซึ่งแสดงด้วยจุดสีขาว เรียกว่า *Segment*

ในโครงงานนี้ใช้ระบบสี HSI ในการทำ Thresholding โดยถ้าสีที่ต้องการอยู่ในช่วง HSI ที่กำหนด ก็จะให้พิกเซลนั้นเป็นสีขาว ถ้าไม่ใช่ให้เป็นสีดำ

$$\begin{aligned} &(H_{\min} \le H \le H_{\max}) and \\ if \quad &(S_{\min} \le S \le S_{\max}) \ and \quad then \ \ pixel = white \ \ else \ \ pixel = black \\ &(I_{\min} \le I \le I_{\max}) \end{aligned} \tag{3.5}$$

ซึ่งมีแผนผังการทำงาน ดังนี้



รูปที่ 3.4 ผังขั้นตอนการทำ Thresholding

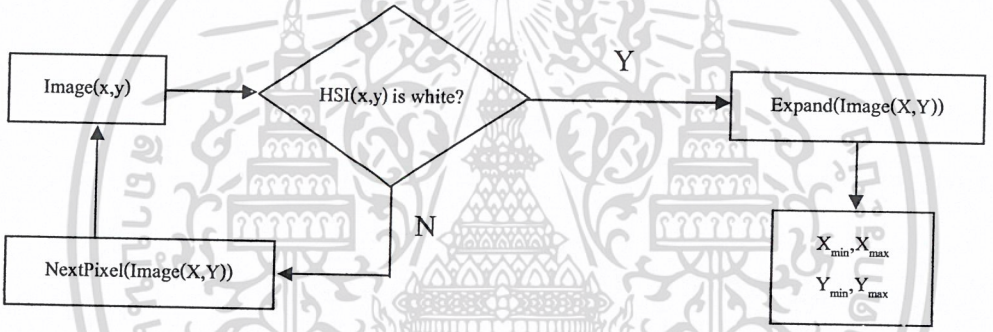
(ก) (ข)

รูปที่ 3.5 (ก) รูปก่อนทำ Thresholding

(ข) รูปหลังทำ Thresholding

3.2.2 Flood Fill

เป็นกระบวนการหาจุดศูนย์กลางของ segment โดยวิธีการของ Flood Fill คือ ทำการหาจุดสีขาวก่อน จากนั้นจะเริ่มขยายตัวไปยังจุดสีขาวรอบด้าน โดยดูว่าถ้ารอบข้างเป็นสีขาวก็ให้จุดสีขาวด้านข้างนั้นเป็นจุดเริ่มในการขยายตัวต่อไป จนครบทั้ง Segment เพื่อหาขอบเขตต่ำสุดและสูงสุดของ segment



รูปที่ 3.6 ผังขั้นตอนการทำ Flood Fill

สามารถหาจุดศูนย์กลางของ X และ Y ดังสมการด้านล่าง

$$X_{center} = \frac{X_{min} + X_{max}}{2} \tag{3.6}$$

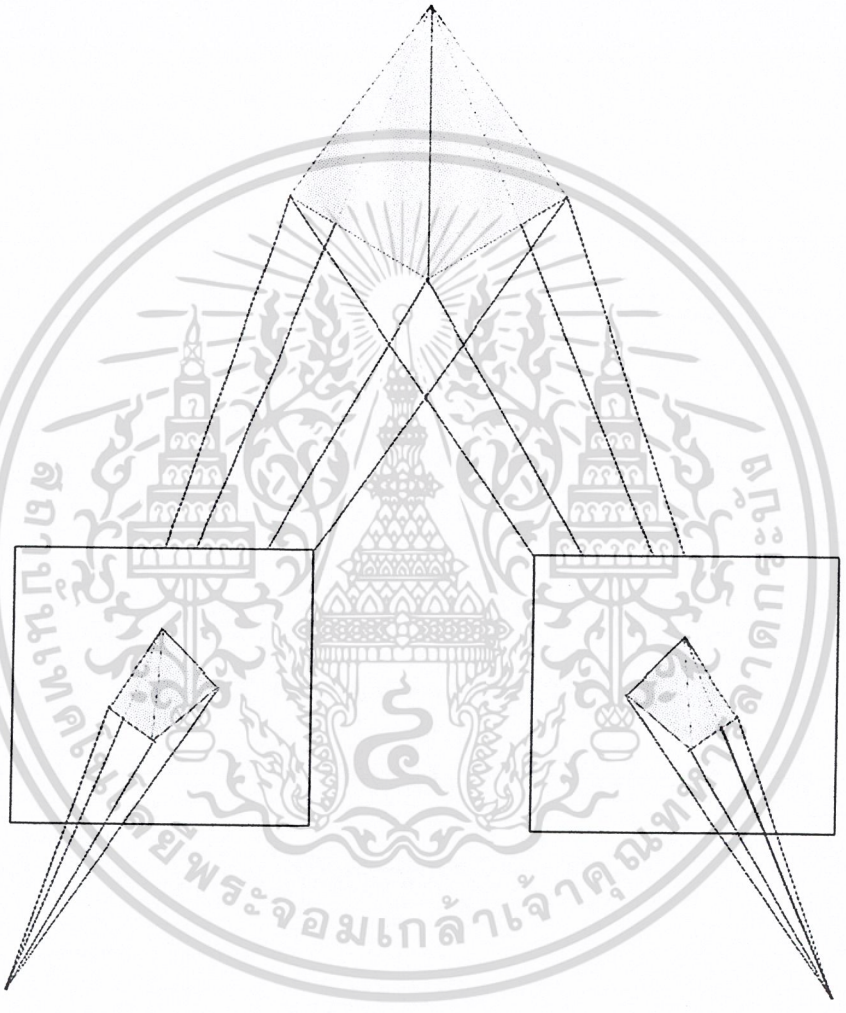
$$Y_{center} = \frac{Y_{min} + Y_{max}}{2} \tag{3.7}$$

โดย $X_{min}, X_{max}, Y_{min}, Y_{max}$ คือ ค่าขอบเขตทั้ง 4 ด้านของ Segment

บทที่ 4

Stereopsis

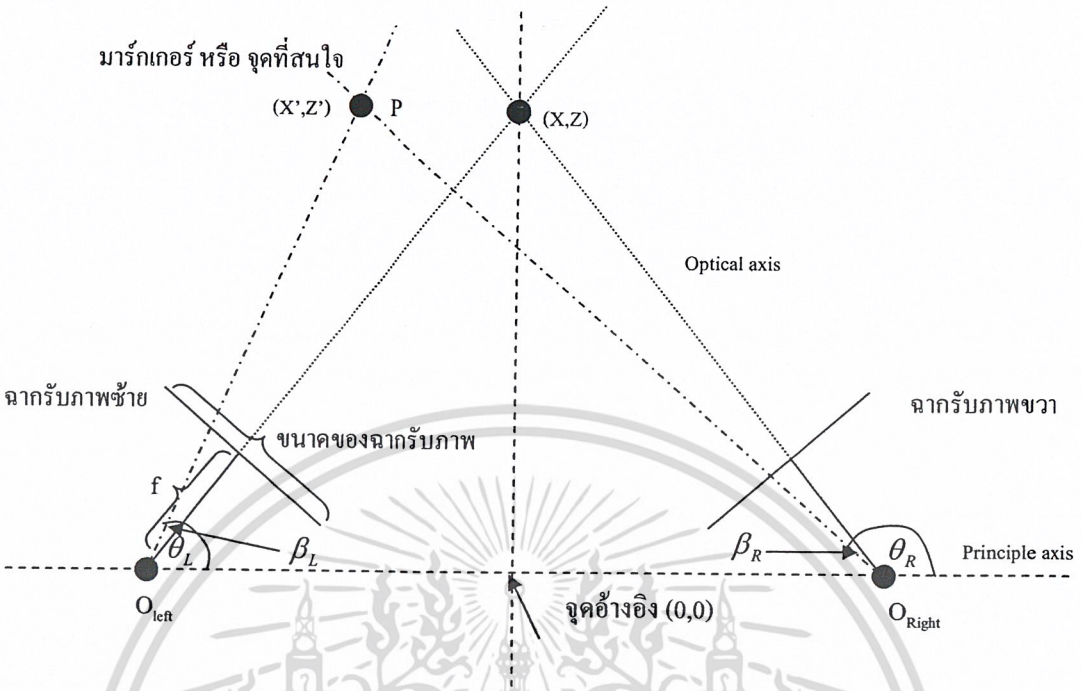
Stereopsis เป็นกระบวนการที่ทำให้เราสามารถอ้างอิงตำแหน่งของวัตถุในรูปแบบ 3 มิติได้โดยพิจารณาจากภาพอย่างน้อย 2 ภาพ โดยภาพเหล่านั้นถ่ายจากมุมที่ต่างกัน ในโครงงานนี้ใช้หลักการนี้เพื่อหาตำแหน่งจริงของมาร์กเกอร์บน World Coordinate



รูปที่ 4.1 แสดงมุมมองที่ได้จากกล้อง 2 ตัว

การหาตำแหน่งบน World coordinate จำเป็นต้องทราบค่า คือ ความยาวโฟกัส และขนาดของฉากรับภาพหรือเซ็นเซอร์รับภาพ CCD

4.1 การคำนวณหาตำแหน่ง 3 มิติ



รูปที่ 4.2 แสดงการจำลองระบบ stereopsis อย่างง่าย

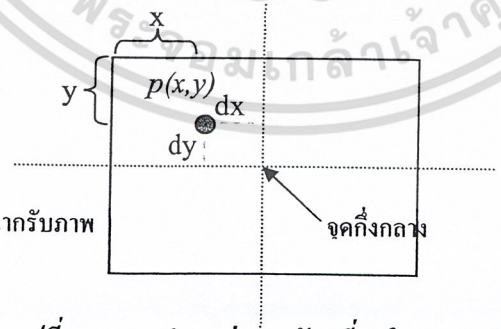
มุม θ_L คือ มุมที่กล้องซ้ายทำมุมกับแกนพื้นฐาน (Principle axis)

มุม θ_R คือ มุมที่กล้องขวาทำมุมกับแกนพื้นฐาน

มุม β_L คือ มุมระหว่าง Ray tracing ของวัตถุ บนฉากรับภาพซ้าย กับ Optical Axis ของกล้องตัวซ้ายบนระนาบ XZ

มุม β_R คือ มุมระหว่าง Ray tracing ของวัตถุ บนฉากรับภาพขวา กับ Optical Axis ของกล้องตัวขวาบนระนาบ XZ

กำหนดให้ความยาวโฟกัสของกล้อง = f



รูปที่ 4.3 แสดงตำแหน่งของวัตถุที่สนใจบนฉากรับภาพ

จากรูปที่ 4.2 และรูปที่ 4.3 จะได้

$$\beta_L = \tan^{-1}\left(\frac{dx_L}{f}\right) \qquad (6.1)$$

$$\beta_R = \tan^{-1}\left(\frac{dx_R}{f}\right) \tag{6.2}$$

โดยที่ dx และ dy เป็นระยะห่างจากจุดกึ่งกลางที่เทียบกับขนาดของฉากรับภาพจริง

การหาตำแหน่งบน world coordinate ของจุด P บนระนาบ XZ ทำได้โดยการหาสมการเส้นตรงของเส้น 2 เส้น คือ เส้นตรง $O_{left}P$ และเส้นตรง $O_{right}P$

สมการเส้นตรงของเส้นตรง $O_{left}P$ คือ

$$Z = m_1 X + c_1 \tag{6.3}$$

โดย

$$m_1 = \begin{array}{ll} \tan(\theta_L + \beta_L) & if \quad x_L \le x_{\max}/2 \\ \tan(\theta_L - \beta_L) & if \quad x_L > x_{\max}/2 \end{array} \tag{6.4}$$

สมการเส้นตรงของเส้นตรง $O_{right}P$ คือ

$$Z = m_2 X + c_2 \tag{6.5}$$

โดย

$$m_2 = \begin{array}{ll} \tan(\theta_R + \beta_R) & if \quad x_R \le x_{\max}/2 \\ \tan(\theta_R - \beta_R) & if \quad x_R > x_{\max}/2 \end{array} \tag{6.6}$$

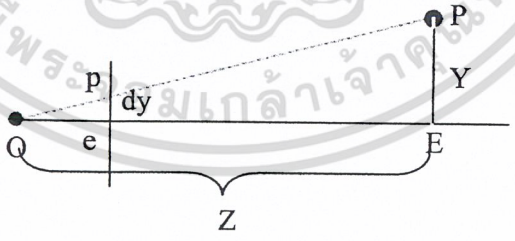
แทนค่าสมการได้

$$X = \frac{c_2 - c_1}{m_1 - m_2} \tag{6.7}$$

$$Z = m_1\left(\frac{c_2 - c_1}{m_1 - m_2}\right) + c_1 \tag{6.8}$$

นำสมการทั้งสองเส้นมาหาจุดตัดจะได้ตำแหน่ง X,Z

การหาตำแหน่งบนจุด P บนแกน Y ทำได้ โดยใช้หลักการของสามเหลี่ยมคล้าย



รูปที่ 4.4 แสดงการหาความสูงวัตถุที่สนใจ

ดังสมการ

$$Y = Z\frac{dy}{f} \tag{6.9}$$

ซึ่งค่า PE ที่ได้เป็นความสูงจากแกนอ้างอิง ซึ่งต้องนำไปบวกกับความสูงของกล้องจึงได้ความสูงที่แท้จริงบน world coordinate

จากสมการ 6.7, 6.8 และ 6.9 สามารถหาค่า X,Y,Z ซึ่งเป็นตำแหน่งจริงบน world coordinate ได้

บทที่ 5
ทฤษฎีและหลักการของฮาร์ดแวร์

การควบคุมฮาร์ดแวร์ต่างๆ สามารถควบคุมโดยผ่านไมโครคอนโทรลเลอร์ และไมโครคอนโทรลเลอร์สามารถติดต่อกับคอมพิวเตอร์ได้ โดยผ่ายทางพอร์ตอนุกรม(Serial Port) ซึ่งในโครงงานนี้ใช้ไมโครคอนโทรลเลอร์เพื่อตรวจจับการคลื่อนไหวส่วนล่าง และควบคุมการทำงานส่วนตอบสนองกลับไปยังร่างกาย

ไมโครคอนโทรลเลอร์ MCS-51 เป็น ไมโครคอนโทรลเลอร์ แบบชิพเดี่ยว มีอยู่ด้วยกันหลายเบอร์ด้วยกันซึ่งแต่ละเบอร์นั้นมีคุณสมบัติที่แตกต่างกันตามแต่จะเลือกใช้งานให้เหมาะสม โดยทั่วไปแล้วคุณสมบัติที่แตกต่างกันในแต่ละเบอร์ก็คือ จำนวน Memory ROM RAM, bit I/O

5.1 ไมโครคอนโทรลเลอร์ AT89C51

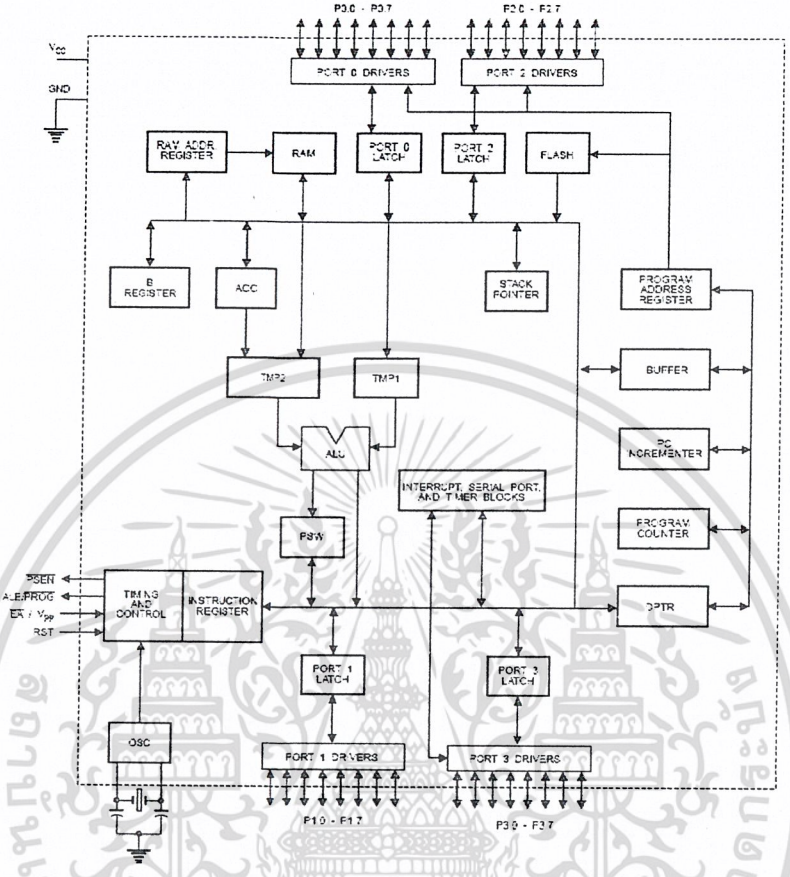
5.1.1 ความแตกต่างของไมโครคอนโทรลเลอร์กับไมโครโปรเซสเซอร์

คุณสมบัติ	ไมโครโปรเซสเซอร์	ไมโครคอนโทรลเลอร์
ขนาดของหน่วยประมวลผลกลาง	ไม่น้อยกว่า 8 บิต	ส่วนใหญ่จะมีขนาด 8 บิต
หน่วยคำนวณทางคณิตศาสตร์และลอจิก	มีอยู่ภายใน	มีอยู่ภายใน
วงจรกำเนิดสัญญาณนาฬิกา	มีอยู่ภายใน	มีอยู่ภายใน
การเชื่อมต่อกับหน่วยความจำโปรแกรม	เชื่อมต่อภายนอกเท่านั้น	ใช้ได้ทั้งภายในและภายนอก
การเชื่อมต่อกับหน่วยความจำข้อมูล	เชื่อมต่อภายนอกเท่านั้น	ใช้ได้ทั้งภายในและภายนอก
การเชื่อมต่อกับพอร์ตอินพุตเอาท์พุต	เชื่อมต่อภายนอกเท่านั้น	มีอยู่ภายในและสามารถขยายได้
ไทเมอร์/เคาน์เตอร์	ไม่มีในซีพียูขนาดเล็ก	มีอย่างน้อย 1 ตัวขนาด 8-16 บิต
วอตช์ด็อกไทเมอร์	ไม่มีในซีพียูขนาดเล็ก	มีอย่างน้อย 1 ตัว
จำนวนขาต่อใช้งาน	ไม่น้อยกว่า 40 ขา	มีตั้งแต่ 8 ขาขึ้นไป

ตารางที่ 5.1 เปรียบเทียบคุณสมบัติของไมโครโปรเซสเซอร์และไมโครคอนโทรลเลอร์

5.1.2 โครงสร้างไมโครคอนโทรลเลอร์ AT89C51

ไมโครคอนโทรลเลอร์ AT89C51 มีโครงสร้างดังรูปที่ 5.1



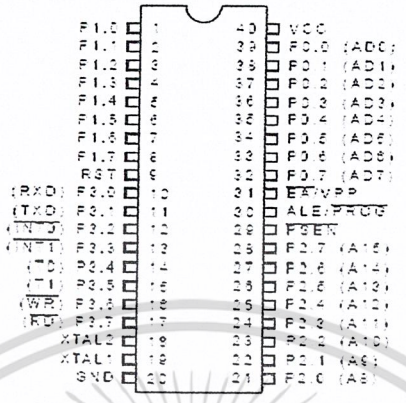
รูปที่ 5.1 รายละเอียดโครงสร้างภายในของไมโครคอนโทรลเลอร์ AT89C51

5.1.3 คุณสมบัติของไมโครคอนโทรลเลอร์ AT89C51

- เป็นไมโครคอนโทรลเลอร์ที่ใช้ซีพียูขนาด 8 บิต
- ภายในมีหน่วยความจำโปรแกรมเป็นแบบแฟลชสามารถลบและเขียนใหม่ได้พันครั้ง
- หน่วยความจำข้อมูลพื้นฐานเป็นหน่วยความจำแบบแรม ในบางเบอร์จะมีหน่วยความจำแบบอีอีพรอมเพิ่มเติม
- ขาพอร์ตเป็นแบบสองทิศทาง สามารถใช้งานเป็นได้ทั้งอินพุตและเอาท์พุต
- มีวงจรสื่อสารอนุกรมแบบฟูลดูเพล็กซ์
- ไทเมอร์/เคาน์เตอร์ขนาด 16 บิตอย่างน้อย 2 ตัว
- สามารถรองรับแหล่งกำเนิดอินเตอร์รัปต์ได้ 6 ประเภท
- สามารถขยายหน่วยความจำภายนอกเพิ่มเติมได้สูงสุด 64 กิโลไบต์
- มีวงจรกำเนิดสัญญาณนาฬิกาอยู่ภายในชิป

5.1.4 การจัดขาของไมโครคอนโทรลเลอร์ AT89C51

ไอซีไมโครคอนโทรลเลอร์ AT89C51 มีการจัดขาตามรูปที่ 5.2



รูปที่ 5.2 การจัดขามาตรฐานของไมโครคอนโทรลเลอร์ AT89C51

ไมโครคอนโทรลเลอร์ AT89C51 จะมีสถาปัตยกรรมและขาใช้งานพื้นฐานซึ่งเหมือนกับ MCS-51ทั่วไป

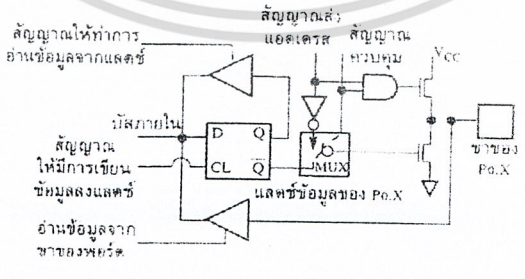
ขา Vcc ใช้สำหรับต่อไฟเลี้ยง +5V

ขา GND เป็นขากราวด์สำหรับต่อกับกราวด์ของระบบ

5.1.5 พอร์ตของไมโครคอนโทรลเลอร์ AT89C51

5.1.5.1 พอร์ต 0

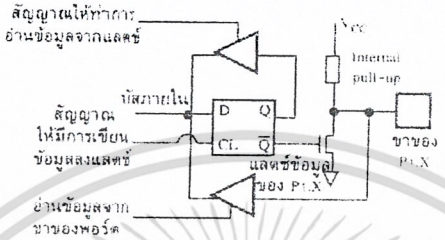
ขาพอร์ต 0 (P0.0-P0.7) มี 8 ขา แต่ละขาสามารถกำหนดให้เป็นได้ทั้งอินพุตและเอาท์พุตสำหรับใช้งานทั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ต 0 ขาใดขาหนึ่งเป็นอินพุตสามารถทำได้โดยการเขียนข้อมูล "1" ไปยังแต่ละบิตของพอร์ตที่ต้องการติดต่อด้วย ส่งผลให้ขาพอร์ตนั้นมีสถานะปล่อยลอย (float) จึงมีอินพุตอิมพีแดนซ์สูงสามารถใช้งานเป็นขาพอร์ตอินพุตได้ นอกจากนั้นขาพอร์ตนี้ยังถูกใช้งานในการติดต่อกับขาแอดเดรสไบต์ต่ำของหน่วยความจำภายนอก (A0-A7) และขาข้อมูล (D0-D7) โดยใช้กระบวนการมัลติเพล็กซ์เข้าช่วยเพื่อสลับการทำงานเป็นไปได้ทั้งขาติดต่อแอดเดรสและขาข้อมูล



รูปที่ 5.3 วงจรภายในพอร์ต 0

5.1.5.2 พอร์ต 1

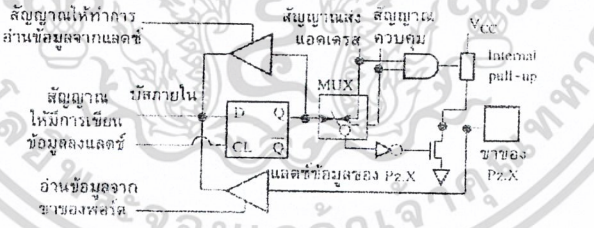
ขาพอร์ต 1 (P1.0-P1.7) มี 8 ขา แต่ละขาสามารถกำหนดให้เป็นได้ทั้งอินพุตและเอาท์พุต สำหรับใช้งานทั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ตใดเป็นอินพุต สามารถทำได้โดยการเขียนข้อมูล "1" ไปยังแต่ละบิตของพอร์ตที่ต้องการติดต่อด้วย นอกจากนั้นในอนุกรม AT89Sxx จะใช้ขา P1.0 เป็นขาอินพุตสำหรับนับค่าของไทเมอร์ 2 และ P1.1 เป็นขาอินพุตทริกเกอร์ของไทเมอร์ 2 ในขณะที่ขา P1.4 ถึง P1.7 เป็นขาสำหรับเชื่อมต่อแบบ SPI เพื่อทำการโปรแกรมข้อมูลในระบบ



รูปที่ 5.4 วงจรภายในพอร์ต 1

5.1.5.3 พอร์ต 2

ขาพอร์ต 2 (P2.0-P2.7) มี 8 ขา แต่ละขาสามารถกำหนดให้เป็นได้ทั้งอินพุตและเอาท์พุตสำหรับใช้งานทั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ตใดเป็นอินพุตสามารถทำได้โดยการเขียนข้อมูล "1" ไปยังแต่ละบิตของพอร์ตที่ต้องการติดต่อด้วย ส่งผลให้ขาพอร์ตนั้นมีสถานะปล่อยลอย (float) จึงมีอินพุตอิมพีแดนซ์สูงสามารถใช้งานเป็นขาพอร์ตอินพุตได้ นอกจากนั้นขาพอร์ตนี้ยังถูกใช้งานในการติดต่อกับขาแอดเดรสไบต์สูงของหน่วยความจำภายนอก (A8-A15)



รูปที่ 5.5 วงจรภายในพอร์ต 2

5.1.5.4 พอร์ต 3

ขาพอร์ต 3 (P3.0-P3.7) มี 8 ขา แต่ละขาสามารถกำหนดให้เป็นได้ทั้งอินพุตและเอาท์พุตสำหรับใช้งานทั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ตใดเป็นอินพุตสามารถทำได้โดยการเขียนข้อมูล "1" ไปยังแต่ละบิตของพอร์ตที่ต้องการติดต่อด้วย ส่งผลให้ขาพอร์ตนั้นมีสถานะปล่อยลอย (float) จึงมีอินพุตอิมพีแดนซ์สูงสามารถใช้งานเป็นขาพอร์ตอินพุตได้ นอกจากนั้นขาพอร์ต 3 ยังเป็นขาที่มีหน้าที่การใช้งานพิเศษ ดังมีรายละเอียดขั้นต้นต่อไปนี้

P3.0 ใช้เป็นขาอินพุตสำหรับรับข้อมูลจากการสื่อสารแบบอนุกรม หรือขา RxD

P3.1 ใช้เป็นขาอินพุตสำหรับรับข้อมูลจากการสื่อสารแบบอนุกรม หรือขา TxD

P3.2 ใช้เป็นขาอินพุตรับสัญญาณอินเตอร์รัปต์จากภายนอกช่อง 0 หรือขา INT0

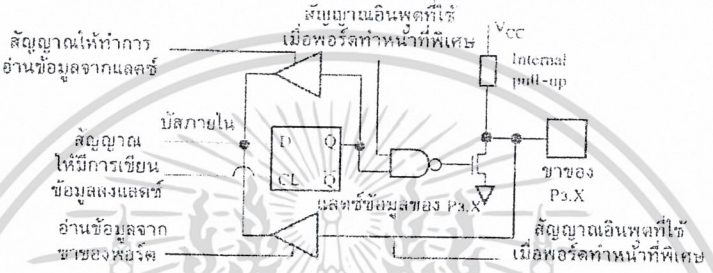
P3.3 ใช้เป็นขาอินพุตรับสัญญาณอินเตอร์รัปต์จากภายนอกช่อง 1 หรือขา INT1

P3.4 ใช้เป็นขาอินพุตสำหรับรับสัญญาณไทเมอร์จากภายนอกช่อง 0 หรือขา T0

P3.5 ใช้เป็นขาอินพุตสำหรับรับสัญญาณอินเตอร์รัปต์จากภายนอกช่อง 1 หรือขา T1

P3.6 ใช้เป็นขาสัญญาณ WR ในกรณีที่ใช้เชื่อมต่อกับหน่วยความจำภายนอก

P3.7 ใช้เป็นขาสัญญาณ RD ในกรณีที่ใช้เชื่อมต่อกับหน่วยความจำภายนอก



รูปที่ 5.6 วงจรภายในพอร์ต 3

ขารีเซต (Reset) ใช้ในการรีเซตการทำงานของไมโครคอนโทรลเลอร์ โดยในการป้อนสัญญาณเพื่อรีเซตสถานะที่ขานี้ต้องอยู่ในระดับรีเซตอย่างน้อย 2 แมชชีนไซเกิล โดยที่วงจรกำเนิดสัญญาณนาฬิกายังคงทำงานต่อเนื่องไปอย่างปกติ

ขา ALE/PROG (Address Latch Enable/Program pulse input) เป็นขาที่ใช้ในการควบคุมการแลตช์ของขาพอร์ต 0 เมื่อมีการใช้งานหน่วยความจำภายนอก นอกจากนั้นขานี้ยังใช้เป็นขาสำหรับพัลส์ของการโปรแกรมสำหรับโปรแกรมข้อมูลลงในไมโครคอนโทรลเลอร์ MCS-51 ในรุ่นที่มีหน่วยความจำโปรแกรมเป็นแบบอีพรอม

ขา PSEN (Program Store Enable) ขานี้ใช้ในการส่งสัญญาณเพื่อร้องขอติดต่อกับหน่วยความจำโปรแกรมภายนอก เมื่อไมโครคอนโทรลเลอร์ต้องการอ่านข้อมูลจากหน่วยความจำโปรแกรมภายนอกตัวไมโครคอนโทรลเลอร์ ตัวไมโครคอนโทรลเลอร์จะส่งสัญญาณออกมาที่ขานี้ 2 ครั้ง ในแต่ละแมชีนไซเกิล แต่ถ้าหากติดต่อกับหน่วยความจำข้อมูลภายนอก ขานี้จะไม่มีการส่งสัญญาณใดๆ ออกมา

ขา EA/Vpp (External Access enable/Programming voltage input) ใช้สำหรับเลือกการติดต่อหน่วยความจำโปรแกรมจากภายนอกหรือภายในตัวไมโครคอนโทรลเลอร์ ถ้าหากขานี้เป็น “0” เป็นการเลือกให้ไมโครคอนโทรลเลอร์ติดต่อกับหน่วยความจำโปรแกรมภายนอก แต่ถ้าหากขานี้เป็น “ 1 ” เป็นการเลือกให้ไมโครคอนโทรลเลอร์ติดต่อกับหน่วยความจำภายในตัวไมโครคอนโทรลเลอร์ นอกจากนี้ ที่ขานี้ยังใช้เป็นขาอินพุตสำหรับรับแรงดันไฟสูงสำหรับการโปรแกรมหน่วยความจำภายในไมโครคอนโทรลเลอร์ สำหรับไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชต้องการแรงดันสำหรับการโปรแกรม +12V

ขา XTAL1 และ XTAL2 เป็นขาสำหรับต่อคริสตอลเพื่อสร้างสัญญาณนาฬิกาในการกำหนดจังหวะการทำงานไมโครคอนโทรลเลอร์

5.1.6 หน่วยความจำภายในของ ไมโครคอนโทรลเลอร์

เพื่อให้การเขียนโปรแกรมนั้นเป็นไปได้อย่างคล่องตัว ไมโครคอนโทรลเลอร์ จึงสนับสนุนชุดคำสั่งที่ช่วยให้โปรแกรมเมอร์สามารถทำการอ้างอิงแอดเดรสของ RAM ภายในได้หลายวิธี โดยเราสามารถอ้างอิงหน่วยความจำ 32 ตำแหน่งแรกเป็นรีจิสเตอร์ หรืออ้างอิงเป็นตำแหน่งหนึ่งในหน่วยความจำก็ได้ การเข้าถึงข้อมูลในตำแหน่งที่ถูกอ้างอิงเป็น register นั้นเราสามารถทำได้โดยใช้คำสั่งที่มีขนาดเพียงแค่ 1 byte ซึ่งการใช้คำสั่งชนิดนี้จะช่วยให้โปรแกรมเมอร์สามารถเขียนโปรแกรมได้อย่างมีประสิทธิภาพ

หน่วยความจำของ ไมโครคอนโทรลเลอร์ นี้จะมีพื้นที่ส่วนหนึ่งขนาด 16 byte ซึ่งเราสามารถทำการอ้างอิงข้อมูลหน่วยความจำส่วนนี้ทีละบิตหรือทีละไบต์ก็ได้ โดยคำสั่งทั่วไปจะสามารถทำการอ้างอิงข้อมูลทีละไบต์จากหน่วยความจำส่วนนี้ แต่ถ้าเราต้องการอ้างอิงข้อมูลทีละบิตในหน่วยความจำส่วนนี้เราจะต้องใช้คำสั่งพิเศษ คำสั่งพิเศษนี้มีประโยชน์มากเมื่อเราต้องการทำการประมวลข้อมูลที่รับมาจากอุปกรณ์ภายนอก เช่น ในงานที่มีการใช้ระบบควบคุม ในหัวข้อต่อไปเราจะพบว่าคำสั่งที่ติดต่อกับ I/O เพื่อทำการเคลื่อนย้ายข้อมูลของ ไมโครคอนโทรลเลอร์ สามารถเคลื่อนย้ายข้อมูลที่มีขนาดเป็นไบต์หรือบิตก็ได้

รูปที่ 5.7 แสดงแผนภาพการจัดสรรหน่วยความจำ หน่วยความจำ ภายใน

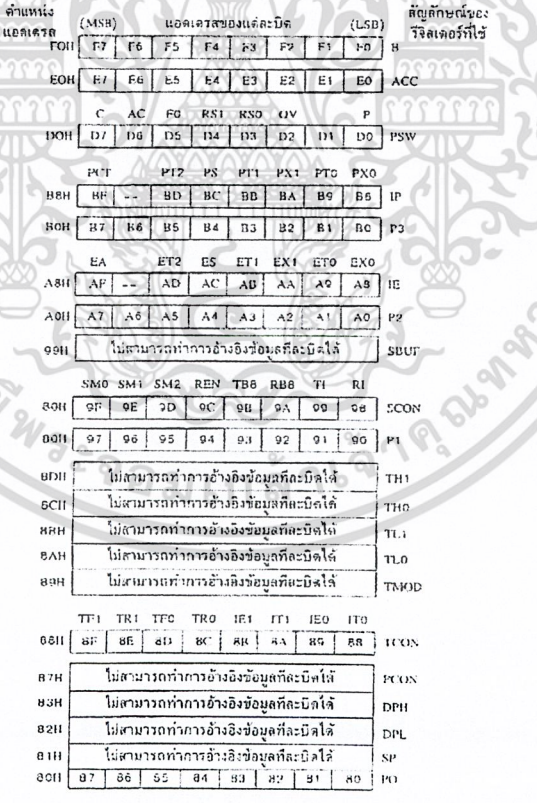
รูปที่ 5.7 แสดงแผนภาพการจัดสรรหน่วยความจำภายในช่วง 128 ตำแหน่งแรกของไมโครคอนโทรลเลอร์ จากรูปเราจะพบว่า 32 ตำแหน่งแรกในหน่วยความจำจะถูกกำหนดให้เป็นรีจิสเตอร์แบงค์ จำนวน 4 แบงค์ ในแต่ละแบงค์จะมีรีจิสเตอร์ 8 ตัว รีจิสเตอร์ในที่นี้เป็นตำแหน่งในหน่วยความจำสำหรับเขียนและอ่านข้อมูลขนาด 8 บิต ซึ่งเราสามารถอ้างอิงได้โดยใช้คำสั่งที่มีขนาด 1 byte และ

เนื่องจากว่าคำสั่งขนาด 1 byte นี้จะใช้บิตเพียงแค่ 3 บิต ในการระบุตำแหน่งของรีจิสเตอร์ที่เราต้องการ ดังนั้นเราจึงใช้คำสั่งนี้อ้างรีจิสเตอร์ ที่แตกต่างกันได้เพียง 8 ตัว

เราจะทราบได้อย่างไรว่าแบงก์รีจิสเตอร์แบงก์ใดจะถูกนำมาใช้งาน แบงก์รีจิสเตอร์ที่กำลังใช้งานอยู่ในขณะนั้นจะถูกเลือกโดยการตั้งค่าที่แฟลกสำหรับเลือกแบงก์ เมื่อแบงก์รีจิสเตอร์หนึ่งๆ ถูกเลือกแล้ว การเคลื่อนย้ายข้อมูลจะเกิดขึ้นกับรีจิสเตอร์ R0 และ R7 ของแบงก์นั้น ถ้าเราต้องการเปลี่ยนไปใช้รีจิสเตอร์ในแบงก์อื่น เราต้องเปลี่ยนค่าในแฟลกสำหรับเลือกแบงก์ก่อนเพื่อเลือกแบงก์ที่จะทำงาน

รูปที่ 5.7 แสดงพื้นที่ขนาด 16 byte ในหน่วยความจำตั้งแต่ตำแหน่งที่ 20H ถึง 30H ซึ่งเป็นส่วนของ RAM ที่เราสามารถอ้างอิงข้อมูลเป็นบิตได้ โดยเราสามารถใช้คำสั่งพิเศษเพื่ออ้างอิงข้อมูลในหน่วยความจำส่วนนี้ทีละบิต หรือใช้คำสั่งทั่วไปเพื่ออ้างอิงข้อมูลทีละไบต์ก็ได้ การใช้คำสั่งพิเศษในไมโครคอนโทรลเลอร์ ช่วยให้โปรแกรมเมอร์สามารถทำการคอมพลีเมนท์ (complement) บิตใดก็ได้หรือทำการแอนด์ (AND) บิตหนึ่งกับอีกบิตหนึ่งที่อยู่ในแอ็คคอมูเลเตอร์(accumulator) หรืออยู่ในรีจิสเตอร์สถานะเป็นต้น

พื้นที่ส่วนสุดท้ายในหน่วยความจำที่มีขนาด 80 ไบท์ นั้นมีชื่อว่า scratchpad ซึ่งโปรแกรมเมอร์สามารถนำข้อมูลที่เปลี่ยนแปลงได้ไปเก็บ



รูปที่ 5.8 แสดงรีจิสเตอร์หน้าที่พิเศษใน ไมโครคอนโทรลเลอร์ AT89C51

5.2 การติดต่อสื่อสารระหว่างคอมพิวเตอร์ และ ไมโครคอนโทรลเลอร์ผ่านพอร์ตอนุกรม

5.2.1 คุณสมบัติของพอร์ตอนุกรม

คุณสมบัติที่สำคัญข้อหนึ่งของ คือ พอร์ตแบบอนุกรมที่มีอยู่ในตัวชิป AT89C51 ในไมโครโปรเซสเซอร์ทั่วไป การเพิ่มพอร์ตอนุกรมจะต้องเพิ่ม UART และวงจรควบคุมไอซีที่สนับสนุนการเชื่อมต่อแบบอนุกรม ซึ่งอุปกรณ์เหล่านี้มีความซับซ้อนและมีราคาค่อนข้างสูง

การเชื่อมต่อแบบอนุกรมของ AT89C51 นั้นเป็นแบบส่งไปและกลับได้พร้อมกัน (full duplex) ซึ่งหมายความว่ามันสามารถทำการรับและส่งข้อมูลได้ในเวลาเดียวกัน การเชื่อมต่อแบบอนุกรมนี้จะมีการพักข้อมูลที่รับเข้ามา (receive-buffered) ซึ่งหมายความว่าก่อนที่ข้อมูลชิ้นแรกจะถูกส่งไปยังไมโครโปรเซสเซอร์ พอร์ตอนุกรมจะสามารถรอรับข้อมูลอนุกรมชิ้นที่สองได้ อย่างไรก็ตามข้อมูลชิ้นแรกจะต้องถูกส่งไปยังไมโครโปรเซสเซอร์ก่อนที่ข้อมูลชิ้นที่สองที่รับเข้ามาจะถูกนำไปเก็บในแลตช์ข้อมูล มิฉะนั้นข้อมูลชิ้นแรกที่อยู่ในนั้นจะถูกเขียนทับ

การเชื่อมต่อแบบอนุกรมนี้มีการใช้รีจิสเตอร์ 2 ตัว ที่ทำหน้าที่รับและส่งข้อมูล เราติดต่อกับรีจิสเตอร์ทั้งสองตัวนี้ได้โดยการอ้างอิงรีจิสเตอร์พิเศษตัวหนึ่งที่มีชื่อว่า SBUF ถ้าเราทำการเขียนข้อมูลลง SBUF แสดงว่าเราได้เขียนข้อมูลลงรีจิสเตอร์ส่งข้อมูล แต่ถ้าเราทำการอ่านข้อมูลจาก SBUF แสดงว่าเราอ่านข้อมูลที่อยู่ในรีจิสเตอร์รับข้อมูล จะเห็นได้ว่ารีจิสเตอร์ที่ทำหน้าที่รับและส่งข้อมูลทั้งสองนี้มีค่าแอดเดรสค่าเดียวกัน

รีจิสเตอร์ควบคุมพอร์ตอนุกรม (serial port control register :SCON) จะเก็บข้อมูลที่ควบคุมการทำงานของพอร์ตอนุกรม

SM1	SM2	SM3	REN	TB8	RB8	TI	RI

รูปที่ 5.9 รีจิสเตอร์ควบคุมพอร์ตอนุกรม

SM1 SCON.7	กำหนดโหมดการทำงานของพอร์ตอนุกรม
SM2 SCON.6	กำหนดโหมดการทำงานของพอร์ตอนุกรม
SM3 SCON.5	บิตนี้จะถูกนำมาใช้ในการสื่อสารระหว่างไมโครโปรเซสเซอร์หลายตัวในโหมดการทำงานที่ 2 และ 3 สำหรับในโหมด 2 หรือโหมด 3 นี้ถ้าบิต SM2 มีค่าเป็น 1 บิต RI จะไม่เกิดการเปลี่ยนแปลง หากค่าในบิตที่ 9 ที่รับเข้ามามีค่าเป็น 0 สำหรับในโหมด 1 ถ้าบิต SM2 มีค่าเป็น 1 บิต RI จะไม่เปลี่ยนแปลงถ้าเกิดการรับบิตสตอปผิด สำหรับในโหมด 0 บิต SM2 จะมีค่าเป็น 0
REN SCON.4	บิตนี้จะถูกเซตหรือเคลียร์โดยโปรแกรมเพื่อทำการรับ/ไม่รับข้อมูล
TB8 SCON.3	บิตที่ 9 ในข้อมูลอนุกรมที่จะถูกส่งไป
RB8 SCON.2	บิตนี้จะมีค่าเท่ากับบิตที่ 9 ในข้อมูลอนุกรมที่รับเข้ามาในโหมดที่ 2 และโหมดที่ 3

	สำหรับในโหมดที่ 1 ถ้าบิต SM2 มีค่าเป็น 0 บิต RB8 จะมีค่าเท่ากับค่าในบิตสตอปของข้อมูลที่รับเข้ามา สำหรับในโหมด 0 บิต RB8 จะไม่ถูกนำมาใช้งาน
TI SCON.1	แฟลก transmit interrupt จะถูกเซตโดยฮาร์ดแวร์ หลังจากได้มีการเลื่อนข้อมูลขนาด 8 บิตออกในโหมด 0 หรือเมื่อได้ทำการส่งข้อมูลจนพบบิตสตอปในโหมดอื่น หลังการส่งข้อมูลแบบอนุกรมค่าในบิตนี้จะถูกเคลียร์โดยซอฟต์แวร์
RI SCON.0	แฟลก receive interrupt จะถูกเซตโดยฮาร์ดแวร์ หลังจากได้มีการเลื่อนข้อมูลขนาด 8 บิตเข้ามาในโหมด 0 หรือที่จุดครึ่งทางของช่องบิตสตอปในโหมดอื่น (ยกเว้นกรณี SM2 มีค่าเป็น 1) หลังการรับข้อมูลแบบอนุกรมค่าในบิตนี้จะถูกเคลียร์โดยซอฟต์แวร์

ตารางที่ 5.2 รีจิสเตอร์ควบคุมการทำงานของพอร์ตอนุกรม (serial port control register: SCON)

SM1	SM2	โหมดการทำงานของพอร์ตอนุกรม	รายละเอียด	อัตราบอด
0	0	0	รีจิสเตอร์ที่เลื่อนค่าได้	ความถี่ออสซิลเลเตอร์ /12 เปลี่ยนแปลงได้
0	1	1	UART ขนาด 8 บิต	ความถี่ออสซิลเลเตอร์ /64
1	0	2	UART ขนาด 9 บิต	ความถี่ออสซิลเลเตอร์ /32
1	1	3	UART ขนาด 9 บิต	เปลี่ยนแปลงได้

ตารางที่ 5.3 โหมดการทำงานของพอร์ตอนุกรม

บิต SM0 และ SM1 จะใช้ในการกำหนดโหมดการทำงานของพอร์ตอนุกรม (ซึ่งมีทั้งหมด 4 โหมดคือ โหมด 0,1,2 และ 3) ต่อไปเราจะกล่าวถึงการทำงานของพอร์ตอนุกรมในแต่ละโหมด

ในโหมด 0 พอร์ตอนุกรมจะเป็นเหมือนรีจิสเตอร์ที่เลื่อนค่าได้ (shift register) ซึ่งจะเลื่อนค่าตามสัญญาณนาฬิกาที่มีอัตราความถี่เท่ากับ 1/12 ของความถี่ออสซิลเลเตอร์ใน 8051 สัญญาณนาฬิกานี้เป็นตัวกำหนดอัตราความเร็วในการเคลื่อนย้ายข้อมูลแบบอนุกรมที่เราเรียกว่า อัตราบอด (baud rate)

ในโหมดที่ 1 พอร์ตอนุกรมจะรับและส่งข้อมูลขนาด 10 บิต ซึ่งข้อมูลขนาด 10 บิตนี้ประกอบด้วยบิตสตาร์ต บิตข้อมูลขนาด 8 บิต และบิตสตอป 1 บิต ในโหมดการทำงานนี้ timer ตัวหนึ่งใน 8051 จะถูกนำมาใช้กำหนดอัตราเร็วในการเคลื่อนย้ายข้อมูลของพอร์ตอนุกรม

ในโหมดที่ 2 พอร์ตอนุกรมจะทำการรับและส่งข้อมูลทีละ 11 บิต ซึ่งข้อมูลขนาด 11 บิตนี้ประกอบด้วยข้อมูลขนาด 8 บิต บิตที่ 9 (เป็นบิตที่เราสามารถกำหนดค่าได้) บิตสตาร์ต และ บิตสตอป โดยบิตที่ 9 นี้จะมีค่าตรงกับบิต TB8 ของรีจิสเตอร์ควบคุมพอร์ตแบบอนุกรม (SCON) ในโหมดการทำงานนี้เราสามารถเลือก baud rate ให้มีค่าเท่าเป็น 1/32 หรือ 1/64 ของความถี่ออสซิลเลเตอร์ใน AT89C51 ได้

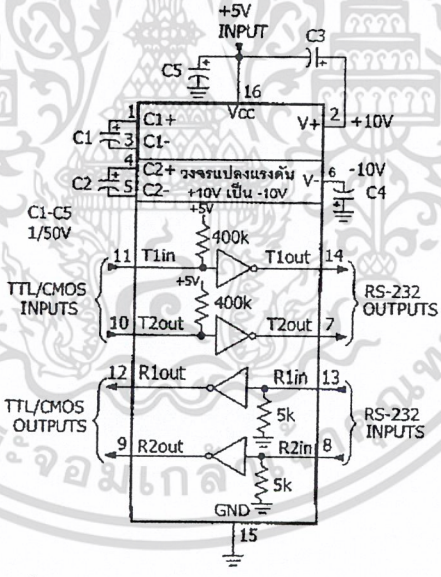
การทำงานของพอร์ตอนุกรมในโหมดที่ 3 จะทำการรับและส่งข้อมูลทีละ 11 บิตและมีบิตที่ 9 ที่กำหนดค่าได้ โดยบิตนี้จะมีค่าตรงกับบิต RB8 ในรีจิสเตอร์ควบคุมพอร์ตแบบอนุกรม (SCON) โหมดที่ 3 นี้จะทำหน้าที่เช่นเดียวกับโหมดที่ 2 ยกเว้นแต่ว่าอัตราเร็วในการเคลื่อนย้ายข้อมูลสามารถเปลี่ยนแปลงได้

ไมโครโปรเซสเซอร์ AT89C51 หลายตัวสามารถติดต่อกันได้โดยการเชื่อมต่อกันแบบอนุกรม เมื่อมีข้อมูลส่งมายังไมโครคอนโทรเลอร์ บิตที่ 9 ในข้อมูลที่รับเข้ามาจะทำให้เกิดการอินเตอร์รัปต์ซึ่งจะบอกให้ไมโครคอนโทรลเลอร์รู้ว่าพอร์ตอนุกรมของมันได้ทำการรับข้อมูลเข้ามาเก็บในที่พักข้อมูล และมันควรทำการโอนย้ายข้อมูลนี้เพื่อนำไปใช้ต่อไป

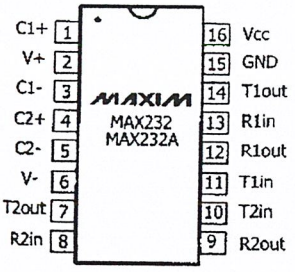
ใน AT89C51 อัตราบอดในโหมด 1 และโหมด 3 จะถูกกำหนดโดยอัตราการเกิดโอเวอร์โฟลว์ของไทเมอร์เบอร์ 1 และเราสามารถเพิ่มอัตราบอด ให้เป็นสองเท่าได้โดยการกำหนดค่าของบิต SMOD ในรีจิสเตอร์ควบคุมให้มีค่าเป็น 1

5.2.2 การติดต่อสื่อสารผ่านทางพอร์ตอนุกรม

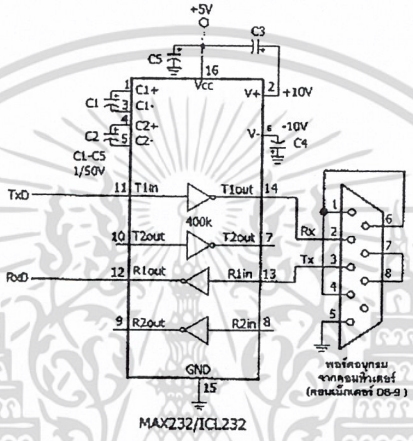
การเชื่อมต่อคอมพิวเตอร์กับไมโครคอนโทรลเลอร์โดยตรงไม่สามารถทำได้เนื่องจากระดับสัญญาณของไมโครคอนโทรลเลอร์ไม่สูงพอจึงต้องอาศัยการเชื่อมต่อผ่านไอซีพิเศษที่ทำหน้าที่แปลงระดับสัญญาณให้สูงขึ้นจนเพียงพอในการติดต่อกันได้ โดยหนึ่งในไอซีที่สามารถใช้ได้ คือ MAX232



รูปที่ 5.10 วงจรภายในของ MAX232



รูปที่ 5. 11 การจัดขาของ MAX232



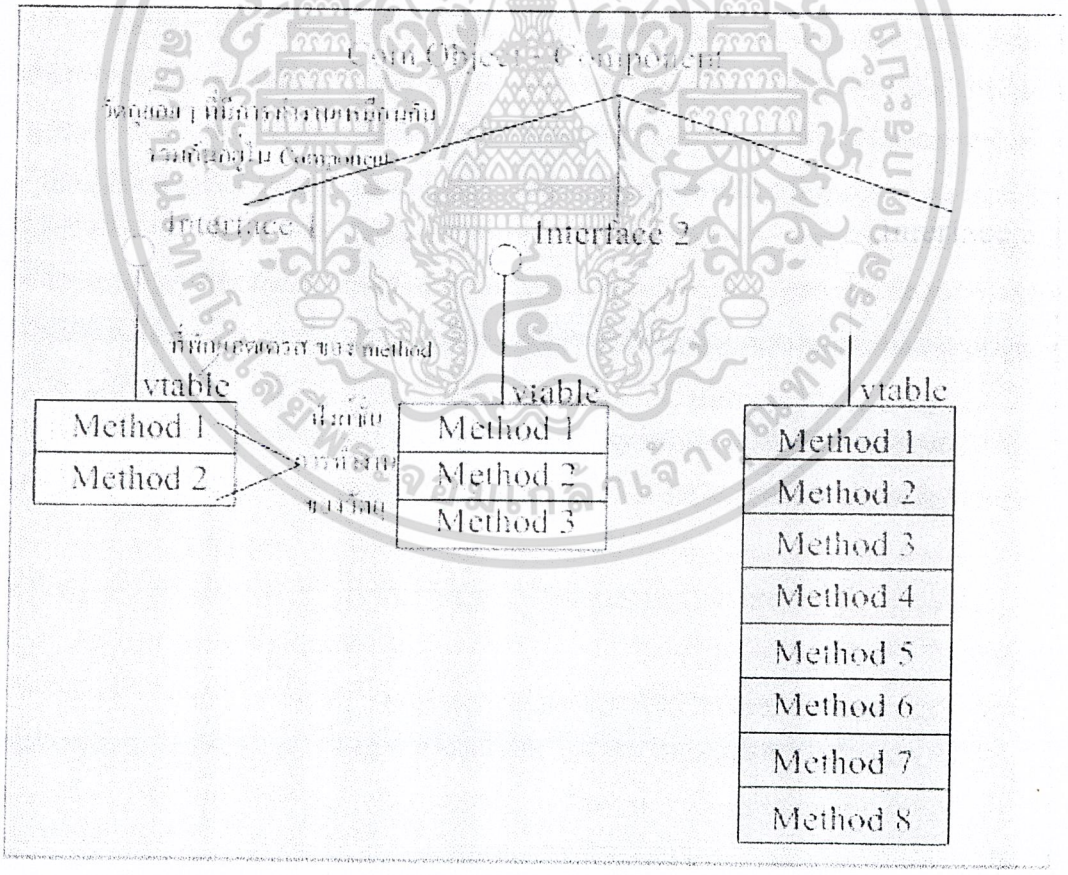
รูปที่ 5.12 วงจรเชื่อมต่อ MAX232 กับพอร์ตอนุกรม

บทที่ 6
ไดเร็กเอ็กซ์ (DirectX)

6.1 ทฤษฎีและหลักการคอมโพเน็นท์กับไดเร็กซ์เอ็กซ์

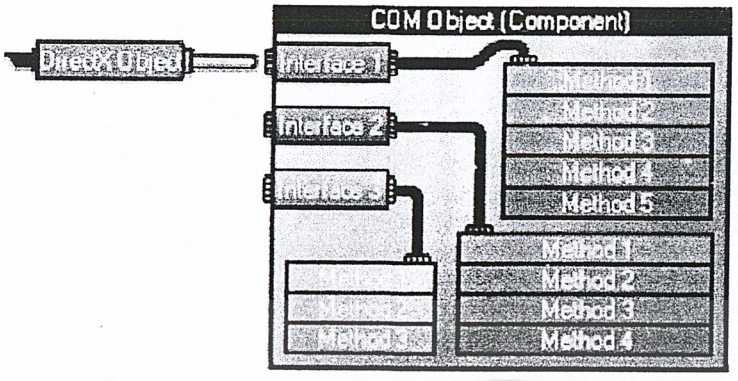
หลักการคอมโพเน็นท์ออบเจ็กท์โมเดลนี้ เป็นวิธีที่ทางไมโครซอฟท์ คิดขึ้นมาเพื่อทำให้ วัตถุต่างๆ ที่มีความเกี่ยวข้องหรือไม่เกี่ยวข้องกันก็ตาม สามารถติดต่อแลกเปลี่ยนข้อมูลกันได้ และสามารถนำส่วนการใช้งานมาใช้ใหม่ในโปรแกรมที่จะเขียนขึ้นใหม่ได้ โดยไม่ขึ้นกับภาษาที่ใช้เขียน ไม่ขึ้นกับระบบปฏิบัติการ แนวคิดวิธีที่จะทำให้ได้ผลดังกล่าว คือ ใช้หลักการออกแบบ วัตถุ (Object) ให้เป็นคอมโพเน็นท์ คือเป็นการรวมเอาวัตถุ (Object) ย่อยๆ ต่างๆ ที่มีหน้าที่การทำงานในแบบเดียวกัน มารวมกันเป็นคอมโพเน็นท์ โดยเรียก วัตถุ (Object) ย่อยๆ เหล่านั้นใหม่ว่าอินเตอร์เฟส (Interface)

คอมโพเน็นท์นั้น จะเป็นชุดที่รวมเอาวัตถุย่อยๆ หรืออินเตอร์เฟส ที่มีฟังก์ชันการทำงานในแบบเดียวกัน และในแต่ละอินเตอร์เฟสนั้น จะมีดีไวซ์ตัวหนึ่งที่มีชื่อว่า วีเทเบิล (vtable) สำหรับพักแอดเดรสของฟังก์ชัน (Method) ตามในรูปที่ 6.1



รูปที่ 6.1 แสดงความสัมพันธ์ของคอมโพเน็นท์ อินเตอร์เฟสและฟังก์ชัน

6.2 ไดเร็กซ์เอ็กซ์ออบเจ็กท์ กับ คอมโพเน็นท์ออบเจ็กท์



รูปที่ 6.2 แสดงการติดต่อกันระหว่าง COM ออบเจ็กท์ กับ ไดเร็กซ์เอ็กซ์ ออบเจ็กท์

คอมโพเน็นท์ เปรียบเสมือน กล่องที่มีรูหรือช่องไว้ให้เสียบ ซึ่งอาจจะมีหลายๆ รู แต่ละรู เปรียบเสมือนอินเตอร์เฟส โดยแต่ละอินเตอร์เฟส ก็จะมีฟังก์ชันการทำงาน (Method) สำหรับไดเร็กซ์เอ็กซ์ออบเจ็กท์ (DirectX Object) นั้น ก็เปรียบเสมือนแจ็ค ที่สามารถเสียบที่รูหรืออินเตอร์เฟสได้ หากวัตถุของไดเร็กเอ็กซ์ เสียบเข้าที่อินเตอร์เฟสใด ก็สามารถเรียกใช้งานฟังก์ชันการทำงานในอินเตอร์เฟสนั้นๆ ได้ หากต้องการใช้งานฟังก์ชันการทำงานอื่น ที่อยู่ในอินเตอร์เฟสอื่นๆ ก็ต้องสร้างไดเร็กซ์เอ็กซ์ออบเจ็กท์ขึ้นมาใหม่ เพื่อนำไปเสียบกับอินเตอร์เฟสที่จะใช้งานฟังก์ชันการทำงานที่ต้องการ และถ้าเลิกใช้อินเตอร์เฟสใด ก็ต้องถอดแจ็คหรือวัตถุนั้นออก

กล่าวได้ว่า อินเตอร์เฟสแต่ละอัน ก็จะมีฟังก์ชันการทำงานต่างๆ ของตัวเอง หากต้องการจะใช้งานฟังก์ชันการทำงานได้ จะต้องสร้าง ไดเร็กซ์เอ็กซ์ออบเจ็กท์ที่ชี้ไปยังอินเตอร์เฟสที่มีฟังก์ชันการทำงานที่ต้องการ โดยการสร้าง ไดเร็กซ์เอ็กซ์ออบเจ็กท์นี้ สามารถทำได้โดยการเรียกใช้ฟังก์ชัน ที่ ไดเร็กซ์เอ็กซ์เตรียมไว้ให้

การปรับปรุงความสามารถของวัตถุประเภท COM จะใช้การสร้างอินเตอร์เฟสตัวใหม่ขึ้นมา เพื่อใช้รองรับความสามารถใหม่มากกว่าจะเปลี่ยนแปลงฟังก์ชันที่มีอยู่ในอินเตอร์เฟสปัจจุบัน

6.3 ไดเร็กซ์เอ็กซ์ เวอร์ชัน 9 (DirectX 9)

6.3.1 ความรู้เบื้องต้นเกี่ยวกับไดเร็กซ์เอ็กซ์

ไมโครซอฟท์ ไดเร็กซ์เอ็กซ์ (Microsoft DirectX) เป็นชุดของแอพพลิเคชั่นโปรแกรมมิ่งอินเตอร์เฟส (Application Programming Interface – API) ได้รับการพัฒนาและออกแบบเพื่อใช้จัดเตรียมอินเตอร์เฟส สำหรับควบคุมฮาร์ดแวร์ทางด้านมัลติมีเดีย บนระบบปฏิบัติการ ไมโครซอฟท์ วินโดว์ (Microsoft Windows) ได้อย่างมีประสิทธิภาพ และเป็นเครื่องมือให้โปรแกรมเมอร์ใช้คำสั่งควบคุมฮาร์ดแวร์ได้อย่างใกล้ชิด โดยไม่ต้องยุ่งยากกับการสร้างส่วนโปรแกรมติดต่อในระดับล่าง ซึ่งเครื่องมือที่

ใช้ติดต่อจะแตกต่างกันไปตามประเภทของอุปกรณ์ ทำให้โปรแกรมเมอร์สร้างซอฟต์แวร์เพื่อทำงานทางด้านมัลติมีเดียได้เป็นอย่างดี เนื่องจากการติดต่อในแต่ละคอมโพเน็นท์จะเป็นของอุปกรณ์ประเภทหนึ่งๆ เป็นอิสระในการเขียนโปรแกรม

API ของ ไดเร็กซ์เอ็กซ์นั้น สร้างขึ้นมาบนรากฐานของ HAL (Hardware Abstraction Layer) สามารถซ่อนลักษณะของดีไวซ์ (Device) ที่เกี่ยวพันอยู่กับฮาร์ดแวร์ และออกแบบมาเพื่อให้สามารถรองรับการขยายตัวของฮาร์ดแวร์ใหม่ๆ ที่จะออกมาในอนาคต ดังนั้นจึงสามารถรองรับความสามารถของฮาร์ดแวร์ที่มีความสามารถเร่งความเร็วใหม่ๆ ที่ปัจจุบันยังไม่มี ด้วยความสามารถในการจำลองการทำงานผ่าน HEL (Hardware Emulation Layer) หรือเลือกที่จะหลีกเลี่ยงไม่ใช้ความสามารถนี้ ถ้า HEL ไม่สนับสนุน ทำให้โปรแกรมเมอร์พัฒนางานทางด้านมัลติมีเดียง่ายขึ้น ไม่จำเป็นต้องทดสอบกับอุปกรณ์ทุกตัวในท้องตลาด เพียงแต่ทดสอบกับไดเร็กซ์เอ็กซ์ก็เพียงพอแล้ว สำหรับผู้สร้างอุปกรณ์ฮาร์ดแวร์ ก็ต้องสร้างไดรเวอร์ของอุปกรณ์นั้นๆ เพื่อให้สามารถทำงานเข้ากับไดเร็กซ์เอ็กซ์ได้

6.3.2 ชุดพัฒนาซอฟต์แวร์สำหรับไดเร็กซ์เอ็กซ์ (DirectX software Development Kit : DirectX SDK)

ชุดพัฒนาซอฟต์แวร์สำหรับไดเร็กซ์เอ็กซ์ นี้ประกอบด้วยชุดของไลบรารี่, ไฟล์ DLL, ไฟล์ header รวมไปถึงเอกสาร และตัวอย่างโปรแกรม ซึ่งช่วยในการพัฒนาซอฟต์แวร์ทางด้านมัลติมีเดีย ซึ่งการที่จะใช้ไดเร็กซ์เอ็กซ์ได้นั้น ก็ต้องทำการลงชุดพัฒนาซอฟต์แวร์สำหรับไดเร็กซ์เอ็กซ์นี้ก่อน

6.3.3 คอมโพเน็นท์ในไดเร็กซ์เอ็กซ์ 9

ไดเร็กซ์เอ็กซ์ 9 (DirectX 9) จะประกอบด้วย Component 6 ตัว คือ

6.3.3.1 DirectX Graphics

DirectX Graphics เป็นคอมโพเน็นท์ใหม่ที่รวมคอมโพเน็นท์เดิมที่ชื่อ DirectDraw กับ Direct3D ไปเป็นอินเตอร์เฟสอันเดียวคือ Direct3D Interface ซึ่งอยู่ในคอมโพเน็นท์ DirectX Graphics นี้ ใช้จัดการทางด้านกราฟฟิกการแสดงรูปภาพ 2 มิติ และวัตถุ 3 มิติ ซึ่งช่วยจัดการการใช้อุปกรณ์แสดงผล 3 มิติ ในรูปแบบที่ไม่ขึ้นกับชนิดของอุปกรณ์ที่จะใช้ (Device-Independent)

6.3.3.2 DirectX Audio

DirectX Audio เป็นคอมโพเน็นท์ใหม่ที่รวมคอมโพเน็นท์เดิมที่ชื่อ DirectSound กับ DirectMusic มาเป็นอินเตอร์เฟสในคอมโพเน็นท์ Direct X Audio นี้ ใช้จัดการทางด้านเสียง สนับสนุนการเร่งความเร็วทางด้านฮาร์ดแวร์ (Hardware acceleration) ในการสร้างเสียงแบบไดนามิก (Dynamic soundtrack) และเอฟเฟ็คเสียง 3 มิติขั้นสูง (Advanced 3D position effect)

6.3.3.3 DirectInput

DirectInput ใช้จัดการทางด้านการรับข้อมูลจากอุปกรณ์อินพุต เช่น คีย์บอร์ด เมาส์ จอยสติ๊ก

เป็นต้น เพื่อเข้าถึงข้อมูลด้วยการติดต่อโดยตรงกับฮาร์ดแวร์ด้วยฮาร์ดแวร์ไดรเวอร์ (Hardware Driver) ซึ่งจะทำงานได้เร็วกว่าการติดต่อด้วยเมสเสจของวินโดว์ นอกจากนี้ยังสนับสนุนการทำงานกับอุปกรณ์แบบแรงตอบสนอง (Force Feedback) อีกด้วย

6.3.3.4 DirectPlay

DirectPlay เป็นมีเดียที่เป็นอิสระบนเน็ตเวิร์ก ซึ่งหมายถึง สามารถรันได้บนเครือข่ายแบบ TCP/IP เครือข่ายแบบ IPX หรือแม้แต่การต่อตรงจากโมเด็ม และการเป็นอิสระทางเน็ตเวิร์กนี้จะสามารถถูกพัฒนาต่อไปได้ในอนาคต นั่นคือ สามารถรองรับอุปกรณ์และโปรโตคอลใหม่ๆ เช่น IPV6, multicast โดยปกติ DirectPlay ใช้จัดการการทำงานด้านเน็ตเวิร์ค โดยไม่จำเป็นต้องไปยุ่งกับการทำงานในระดับล่าง มีหน้าที่การทำงานสำหรับการส่งข้อมูลทีเดียวให้หลายๆ เครื่องเป็นกลุ่มได้ สามารถเขียนโปรแกรมติดต่อทางด้านเน็ตเวิร์ค ทั้งในการติดต่อผ่านเน็ตเวิร์คระหว่างโปรแกรมแบบมัลติเพลเยอร์ (Multiplayer) แบบเพียร์ทูเพียร์ (Peer-to-Peer) และแบบไคลเอ็นท์/เซิร์ฟเวอร์ (Client/Server)

6.3.3.5 DirectShow

DirectShow ใช้สำหรับการทำงานด้านมัลติมีเดีย เป็นสถาปัตยกรรมทางด้านมีเดียสตรีมมิ่ง (media-streaming) ของไมโครซอฟท์ วินโดว์ ที่มีความสามารถในการแคปเจอร์ (Capture) สูง สามารถบรรจุข้อมูลที่เป็นวีดีโอ และการบีบอัดข้อมูลไฟล์ (Audio data compressed) ได้หลายรูปแบบ เช่น การเล่นไฟล์วีดีโอ ประเภท MPEG, Audio-Video(AVI), MPEG-1 Layer3(MP3) และไฟล์นามสกุล .wav เป็นต้น

6.3.3.6 DirectSetup

DirectSetup ใช้สำหรับการจัดการทางด้านการติดตั้งองค์ประกอบที่จะต้องใช้ในงานไดเร็กซ์เอ็กซ์บนเครื่องคอมพิวเตอร์ของผู้ใช้อย่างอัตโนมัติ หากเครื่องคอมพิวเตอร์นั้นยังไม่ได้ทำการติดตั้ง

6.4 ไดเร็กซ์เอ็กซ์ กราฟฟิก (DirectX Graphics)

6.4.1 ความรู้เบื้องต้นกับไดเร็กซ์เอ็กซ์ กราฟฟิก

ไดเร็กซ์เอ็กซ์ กราฟฟิก (DirectX Graphics) เป็นคอมโพเน็นท์ใหม่ ในไดเร็กซ์เอ็กซ์ เวอร์ชั่น 8 ซึ่งรวมเอาการทำงานของ คอมโพเน็นท์เดิม คือ ไดเร็กซ์ดรอว์ (DirectDraw) กับ ไดเร็กซ์ทรีดี (Direct3D) มาเป็นอินเตอร์เฟสเดียวคือ ไดเร็กซ์ทรีดี ซึ่งจะรวมการทำงานในการแสดงภาพ 2 มิติ และ 3 มิติ ไว้ในอินเตอร์เฟสนี้

ไดเร็กซ์ทรีดี ได้รับการออกแบบ เพื่อช่วยในการสร้างเกมที่แสดงภาพแบบ 2 และ 3 มิติ บนเครื่องคอมพิวเตอร์ภายใต้ระบบปฏิบัติการวินโดวส์ ซึ่งกล่าวได้ว่า ไดเร็กซ์ทรีดี คือซอฟต์แวร์อินเตอร์เฟสซึ่งจัดการการแสดงผลที่รองรับการเข้าถึงอุปกรณ์แสดงผล โดยยังคงรักษาความเข้ากันได้กับกราฟฟิก

ดีไวซ์อินเตอร์เฟส (Graphics Device Interface : GDI) และจัดการให้มีการใช้งานในแบบที่ไม่ขึ้นกับชนิดของอุปกรณ์ (Device Independent) และมีความสามารถในการใช้คุณสมบัติพิเศษที่อยู่ในฮาร์ดแวร์ได้

ไดเร็กซ์ทรีดี นี้เป็นแอพพลิเคชั่นโปรแกรมมิ่งอินเทอร์เฟสระดับล่าง ซึ่งเป็นเครื่องมือสำหรับโปรแกรมเมอร์ที่ต้องการสร้างเกมหรือโปรแกรมมัลติมีเดียที่แสดงผล 3 มิติ ที่มีประสิทธิภาพสูง บนระบบปฏิบัติการวินโดว์ ซึ่งการใช้เอพีไอไดเร็กซ์ทรีดีนี้ ทำให้โปรแกรมสามารถติดต่อโดยตรงกับฮาร์ดแวร์เร่งความเร็วในระดับล่าง จึงมีความยืดหยุ่นในการใช้งานสูง

6.4.2 คุณสมบัติของไดเร็กซ์ทรีดี

6.4.2.1 สนับสนุนการใช้งาน Depth Buffer (โดยใช้ z-buffer หรือ w-buffer)

6.4.2.2 สามารถเรนเดอร์ภาพได้ทั้งในแบบ Flat Shading และ Gouraud Shading

6.4.2.3 สามารถสร้างแหล่งกำหนดแสงได้หลายชนิดและหลายแหล่ง

6.4.2.4 รองรับการใช้วัสดุแมตทีเรียล (Material) และพื้นผิวเท็กซ์เจอร์ (texture) รวมไปถึงการทำมิพแมพ (mipmap)

6.4.2.5 สนับสนุนซอฟต์แวร์จำลองการใช้งานของฮาร์ดแวร์ (Software Emulation Driver) ที่มีประสิทธิภาพสูง

6.4.2.6 มีการทำงานโดยไม่ขึ้นกับชนิดของฮาร์ดแวร์

6.4.2.7 รองรับการแปลงพิกัดวัตถุ (Transformation) และการขริบภาพ (Clipping)

6.4.2.8 สนับสนุนคำสั่งพิเศษที่มีอยู่ในซีพียู สามารถรองรับสถาปัตยกรรมแบบ Intel MMX, Intel Streaming Single-Instruction, Multiple-Data (SIMD) Extensions (SSE) และสถาปัตยกรรมทรีดีของ AMD (AMD's 3D Architecture)

6.4.2.9 สนับสนุน Hardware Abstraction Layer (HAL)

6.4.2.10 สนับสนุนการทำ page flipping ด้วย back buffer จำนวนมากในโปรแกรมที่แสดงผลแบบเต็มจอภาพ (full screen)

6.4.2.11 สนับสนุนการตัด (Clipping) ในโปรแกรมประยุกต์ที่ทำงานในวินโดว์โหมด และแบบโหมดเต็มจอภาพ

6.4.2.12 รองรับการทำ 3D - z buffer

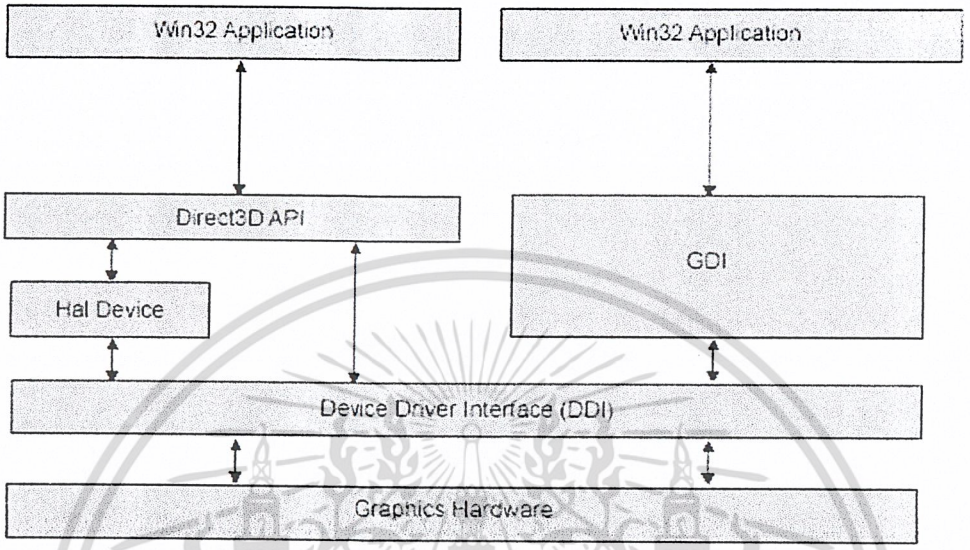
6.4.2.13 สามารถใช้งาน image-stretching hardware ได้

6.4.2.14 สามารถเข้าถึงหน่วยความจำของอุปกรณ์แสดงผล (display-device memory) แบบ Standard และ Enhance ได้ในขณะเดียวกัน

นอกจากนี้ยังมีคุณสมบัติอื่นๆ เช่น การใช้งานฮาร์ดแวร์แต่เพียงผู้เดียว (Exclusive hardware access) และการเปลี่ยนความละเอียดในการแสดงผล (Resolution switching)

ด้วยคุณสมบัติข้างต้นเหล่านี้ การพัฒนาโปรแกรมด้วยไดเร็กซ์ทรีดี จึงมีประสิทธิภาพสูงกว่าการใช้ กราฟฟิกดีไวซ์อินเตอร์เฟส (GDI) ของวินโดว์และการเขียนโปรแกรมบนระบบปฏิบัติการดอส

ตามแผนผังข้างล่างนี้ แสดงถึงความสัมพันธ์ระหว่างอินเตอร์เฟสไดเร็กซ์ทรีดี กราฟฟิกดีไวซ์อินเตอร์เฟส (Graphics Device Interface : GDI) ฮาร์ดแวร์แอ็บสแตร็กชั่นเลเยอร์ (Hardware abstraction Layer : HAL) และ ฮาร์ดแวร์



รูปที่ 6.3 แสดงความสัมพันธ์ระหว่างไดเร็กซ์ทรีดีกับระบบคอมพิวเตอร์

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

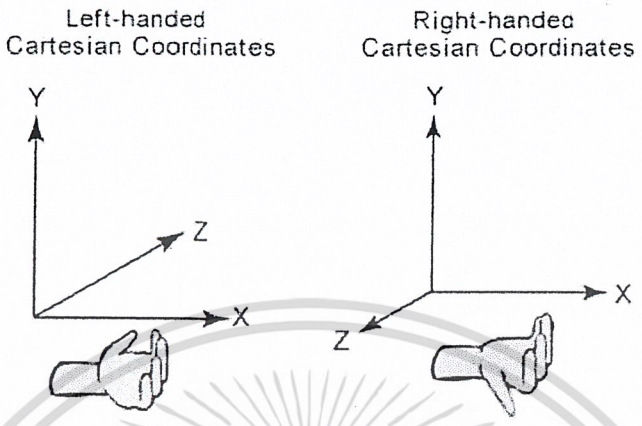
หากฮาร์ดแวร์ที่มีอยู่ในเครื่องคอมพิวเตอร์ใดไม่สนับสนุนการเร่งความเร็วในส่วนใดของไปป์ไลน์ก็ตาม ไดเร็กซ์ทรีดีจะใช้ซอฟต์แวร์ที่มีประสิทธิภาพสูงในการทำงานในส่วนนั้นแทน แต่การทำงานของซอฟต์แวร์ ก็ยังมีประสิทธิภาพต่ำกว่าใช้ฮาร์ดแวร์

6.5 ทฤษฎีระบบพิกัด 3 มิติ

6.5.1 ระบบพิกัด 3 มิติ (3-D Coordinate System)

แบบฉบับของโปรแกรมประยุกต์ที่แสดงผลกราฟฟิก 3 มิติ จะมี 2 แบบในการแสดงระบบพิกัด 3 มิติ คือ มือซ้าย และมือขวา ในระบบพิกัดทั้งคู่ แกนด้านบวกของ x จะไปทางด้านขวา และแกนด้านบวกของ y จะขึ้นไปข้างบน สามารถสังเกตเพื่อจดจำได้จากทิศทางของแกนบวกของ z ได้ โดยการหงายมือขึ้น จากนั้นกางนิ้วมือออกชี้ไปยังแกนบวกของ x แล้วงอนิ้วขึ้นด้านบนตามทิศทางของแกนบวกของ y ทิศทางที่นิ้วโป้งชี้ไป จะเป็นทิศทางของแกนบวกของ z ตามมือที่ใช้

ไมโครซอฟท์ ไดเร็กซ์ทรีดี ใช้ระบบพิกัด 3 มิติ แบบมือซ้าย ถ้าต้องการทำโปรแกรมที่เป็นระบบมือขวา จะต้องเปลี่ยนข้อมูลบางอย่าง ดังนี้



รูปที่ 6.4 แสดงระบบพิกัด 3 มิติ แบบมือซ้ายและมือขวา

• เปลี่ยนรายการของจุดของสามเหลี่ยม (triangle vertices) โดยสลับแบบตามเข็มนาฬิกาจากด้านหน้า ตัวอย่างเช่น ถ้าลำดับจุดเป็น v0, v1, v2 ให้เปลี่ยนข้อมูลก่อนที่จะส่งค่าไปให้ Direct3D คือ v0,v2, v1

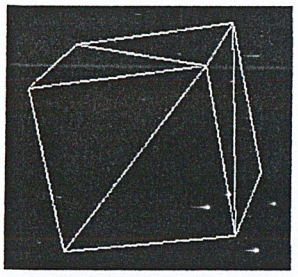
• ใช้การวิวแมทริกซ์เพื่อปรับพิกัดเวิลด์สเปซ (World Space) ด้วย -1 ในทิศทางของแกน Z ซึ่งทำได้โดย สลับเครื่องหมายของ _31, _32, _33, _34 แถวที่สามในแมทริกซ์ ของโครงสร้าง D3DMATRIX ที่ใช้สำหรับวิวแมทริกซ์

สำหรับฟังก์ชันที่จะใช้บอกไดเร็กซ์ทรีดีว่า เราจะใช้ระบบพิกัด 3 มิติ แบบไหนนั้น ก็คือ D3DXMatrixLookAtLH( ) สำหรับระบบมือซ้าย และ D3DXMatrixLookAtRH( ) สำหรับระบบมือขวา

6.5.2 ชุดของจุดของวัตถุ 3 มิติ (3-D Primitive)

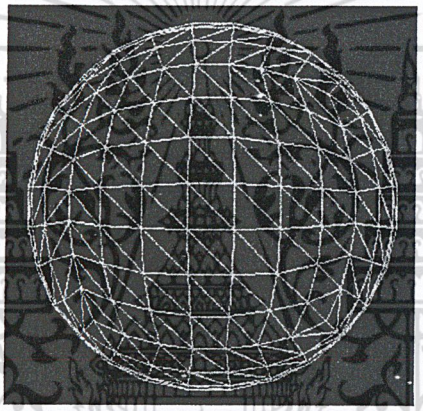
3-D Primitive เป็นชุดของจุด (vertices) ที่เป็นของวัตถุ 3 มิติ และพริมิทีฟ ที่ง่ายที่สุด คือ ชุดของจุดในระบบพิกัด 3 มิติ ที่ถูกเรียกว่า รายการจุด (Point List)

โดยปกติ 3-D Primitive จะหมายถึงโพลีกอน เป็นรูปปิด 3 มิติ ที่มีจุดที่มีการเชื่อมกันอย่างน้อย 3 จุด โพลีกอนแบบพื้นฐานที่สุดคือรูปสามเหลี่ยม ไมโครซอฟท์ ไดเร็กซ์ทรีดี ส่วนใหญ่ใช้สามเหลี่ยมในการประกอบโพลีกอน เพราะจุดสามจุดในรูปสามเหลี่ยมนั้นจะรับประกันได้ว่าอยู่ในระนาบเดียวกัน เพื่อให้การเรนเดอร์นั้นเป็นไปอย่างมีประสิทธิภาพ จึงสามารถประกอบรูปสามเหลี่ยมเป็นโพลีกอนหรือวัตถุที่มีความซับซ้อนได้



รูปที่ 6.5 แสดงรูปลูกบาศก์ที่ประกอบกันขึ้นมาจากรูปสามเหลี่ยม

รูปข้างบนนี้แสดงรูปสี่เหลี่ยมลูกบาศก์ ในแต่ละด้านประกอบขึ้นมาจากสามเหลี่ยม 2 รูป ซึ่งรูปสี่เหลี่ยมลูกบาศก์นี้ประกอบขึ้นจากชุดของรูปสามเหลี่ยม นอกจากนี้ ยังสามารถลงเท็กซ์เจอร์และชนิดวัสดุ (material) บนพื้นผิววัตถุได้ เพื่อสร้างพื้นผิวของพริมิทีฟ ให้เห็นเป็นลูกสี่เหลี่ยมลูกบาศก์ทรงตันได้



รูปที่ 6.6 แสดงภาพทรงกลมที่ประกอบขึ้นมาจากรูปสามเหลี่ยม

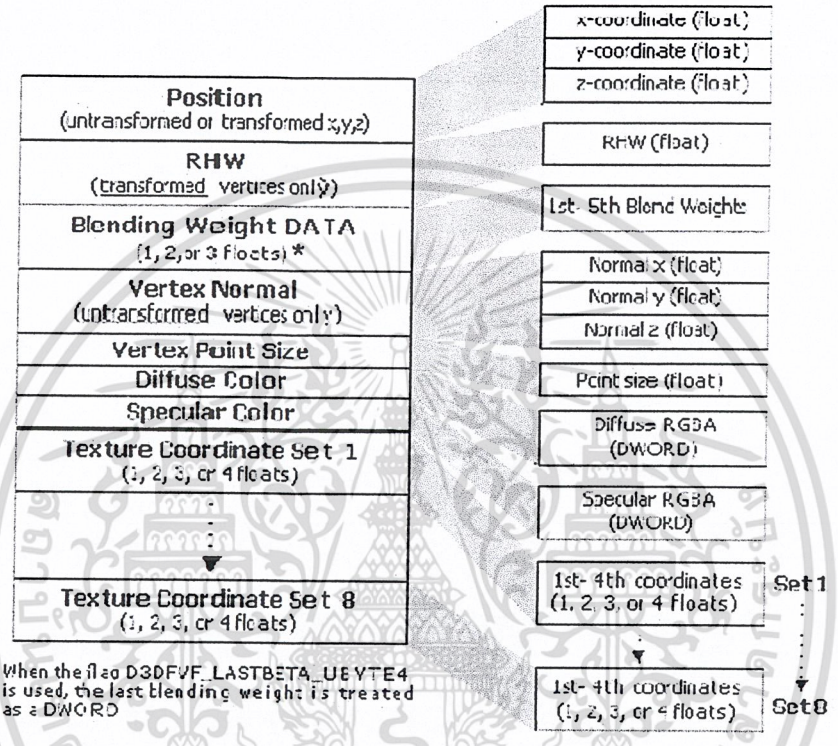
จากรูปด้านบนนี้ แสดงถึงทรงกลมที่ประกอบขึ้นมาจากรูปสามเหลี่ยม ซึ่งแสดงให้เห็นว่าสามารถใช้สามเหลี่ยมเพื่อสร้างพริมิทีฟสำหรับรูปทรงที่มีพื้นผิวราบเรียบได้ เมื่อทำการลงพื้นผิวจะเห็นเป็นรูปทรงกลมที่พื้นผิวเรียบได้

6.5.3 เวกเตอร์ (Vector)

เวกเตอร์ในไดเร็กซ์ทรีดี มีความหมายว่า ค่าที่แสดงถึงระยะและทิศทางของวัตถุ เช่น เวกเตอร์ที่แสดงพิกเซลบนจอภาพ จะประกอบด้วยค่าในแกน x และ y ถ้าทิศทางในแนวแกน x เป็นบวก จะหมายถึงพิกเซล นั้นจะอยู่ไปทางด้านขวาของจอภาพ ถ้าทิศทางในแนวแกน y เป็นบวก จะหมายถึงพิกเซลที่ตำแหน่ง y นั้นจะอยู่ไปทางด้านล่างของจอภาพ เพราะตำแหน่งเริ่มต้นของจอภาพ คือมุมบนซ้ายสุด (0, 0) ซึ่งเวกเตอร์ที่ประกอบด้วยค่า 2 ค่าอย่างนี้เราเรียกว่าเวกเตอร์ 2 มิติ(2-D Vector) ซึ่งในไดเร็กซ์ทรีดี จะใช้เวกเตอร์อยู่ 3 ประเภท คือ เวกเตอร์ 2 มิติ เวกเตอร์ 3 มิติ และเวกเตอร์ 4 มิติ

6.5.4 เวอร์เท็กซ์ (Vertex)

เวอร์เท็กซ์ คือจุดในโลก 3 มิติ โดยอย่างน้อยจะมีเวกเตอร์ 1 เวกเตอร์ การสร้างเวอร์เท็กซ์ในไดเร็กซ์ทรีดี จะใช้วิธีสร้างแบบเฟล็กซิเบิลเวกเตอร์ (Flexible Vector Format : FVF) ซึ่งเป็นโครงสร้างที่ประกอบไปด้วยเวกเตอร์ที่ต้องการจะใช้งานเท่านั้น เวกเตอร์ไหนที่ไม่ใช้ก็ไม่ต้องใส่ โดยจะต้องเรียงลำดับเวกเตอร์และใช้ประเภทของตัวแปรตามรูปแบบในรูป 6.7 นี้



รูปที่ 6.7 รูปแบบลำดับเวกเตอร์ในเวอร์เท็กซ์

6.5.5 โมเดลสเปซ (Model Space)

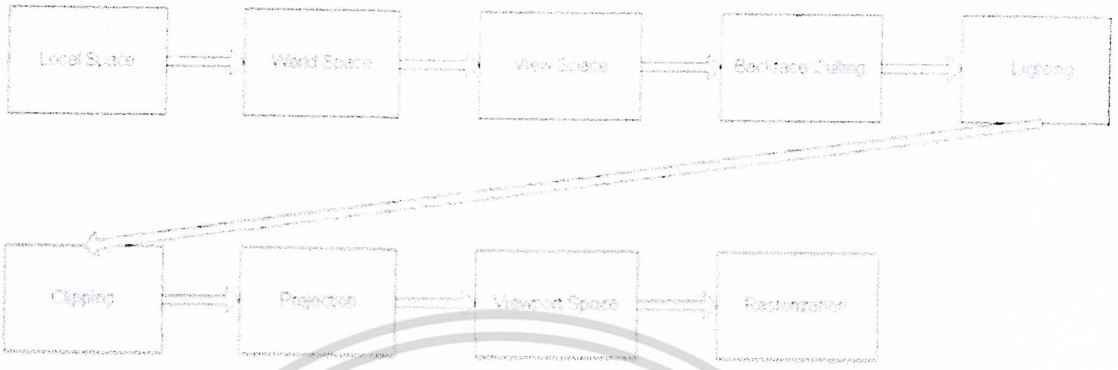
โมเดลสเปซ (Model Space) เป็นพิกัดในโลก 3 มิติ (3-D Coordinate) ที่ใช้ภายในโมเดลวัตถุแต่ละอัน ซึ่งมีสามเหลี่ยมแต่ละชิ้นนั้นเกิดขึ้นจากเวอร์เทกซ์ จำนวน 3 จุด โดยแต่ละจุดจะมีจุดกำเนิดอ้างอิงจุดเดียวกัน ซึ่งจุดกำเนิดนี้จะอยู่ที่กลางโมเดลแต่ละอัน

สำหรับพิกัดต่างๆ ที่โมเดลแต่ละอันใช้นี้จะเรียกว่าพิกัดของตนเอง (Local Coordinate) ขอบเขตของโมเดลสเปซ ขึ้นอยู่กับขนาดของโมเดล ว่าจะมีความกว้าง ยาว และสูงเพียงใด

6.6 กระบวนการเรนเดอร์ออบเจ็กท์ 3 มิติ

ในการพัฒนาโปรแกรมประยุกต์ที่ใช้ไดเร็กซ์ทรีดีนั้น จะใช้พื้นฐานอยู่บนทฤษฎีเกี่ยวกับ 3 มิติ ได้แก่ เวอร์เท็กซ์ โพลีกอนและคำสั่งที่ใช้ควบคุมข้อมูลเหล่านี้ รวมทั้งการเข้าถึงการแปลงพิกัดของวัตถุ ซึ่งเป็นความสามารถในการเข้าถึงไปป์ไลน์ของกราฟฟิก 3 มิติ

Rendering Pipeline คือ อนุกรมของการทำงานตั้งแต่การสร้างวัตถุใน 3 มิติ จนถึงการนำภาพออกไปแสดงบนหน้าจอในลักษณะของภาพ 2 มิติ



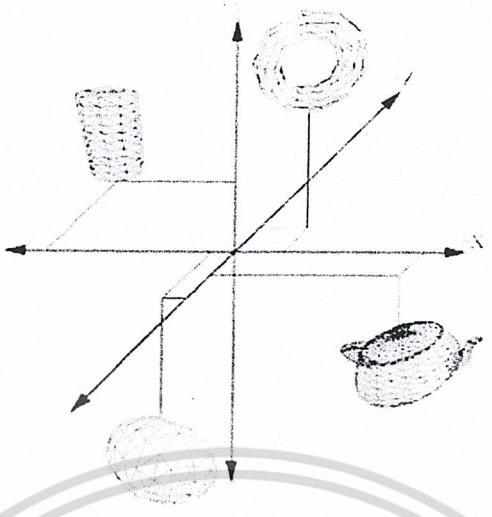
รูปที่ 6.8 ไปป์ไลน์ในการสร้างภาพ 3 มิติของไดเร็กซ์เอ็กซ์

คำอธิบายแต่ละขั้นตอนของไปป์ไลน์

6.6.1 Local space เป็นขั้นตอนในการกำหนดตำแหน่งของสามเหลี่ยมที่ประกอบกันขึ้นเป็นวัตถุในระบบพิกัดของวัตถุนั้น ๆ

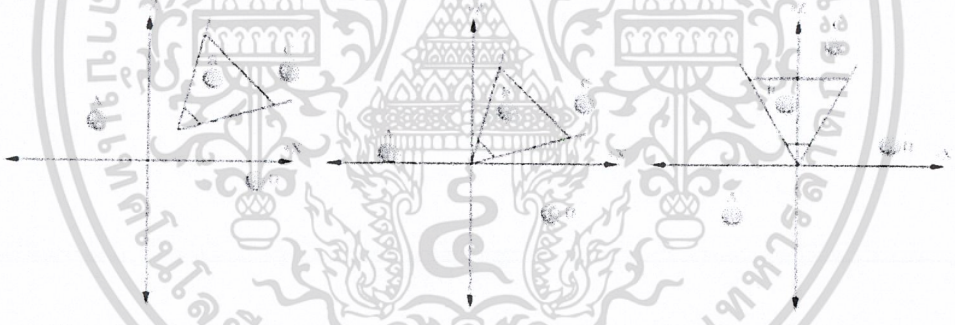
รูปที่ 6.9 แสดงการกำหนดตำแหน่งของวัตถุในระบบพิกัดของวัตถุนั้นๆ

6.6.2 World space เป็นขั้นตอนในการกำหนดตำแหน่งของวัตถุซึ่งถูกสร้างในขั้นตอนของ Local space ให้อยู่ในตำแหน่งที่ต้องการภายในระบบพิกัดของเวิร์ล



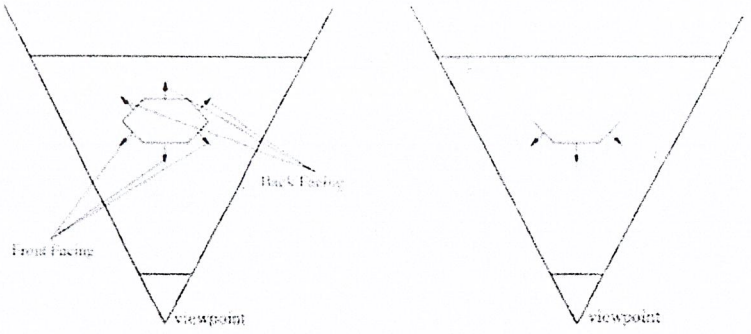
รูปที่ 6.10 การแปลงตำแหน่งของวัตถุให้อยู่ในระบบพิกัดของเวิร์ล

6.6.3 View space เป็นขั้นตอนในการเปลี่ยนตำแหน่งกล้องให้มาอยู่ที่จุดออริจิน (0,0,0) และให้หน้ากล้องหันไปทางทิศ +z โดยการเปลี่ยนตำแหน่งกล้องนี้จะทำให้ตำแหน่งของวัตถุเปลี่ยนแปลงตามไปด้วย



รูปที่ 6.11 แสดงขั้นตอนในการเปลี่ยนตำแหน่งกล้องให้มาอยู่ที่จุดออริจิน

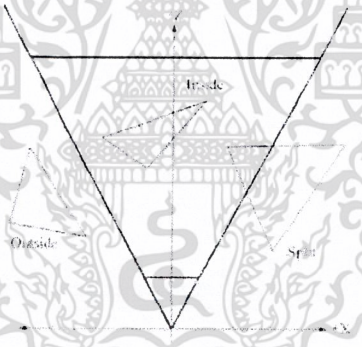
6.6.4 Backface Culling เป็นขั้นตอนการตัดส่วนของวัตถุ (สามเหลี่ยม) ที่หันหลังให้กับกล้องซึ่งเป็นส่วนที่ไม่สามารถมองเห็นได้ออกไป



รูปที่ 6.12 แสดงขั้นตอนการตัดส่วนของวัตถุที่หันหลังให้กับกล้อง

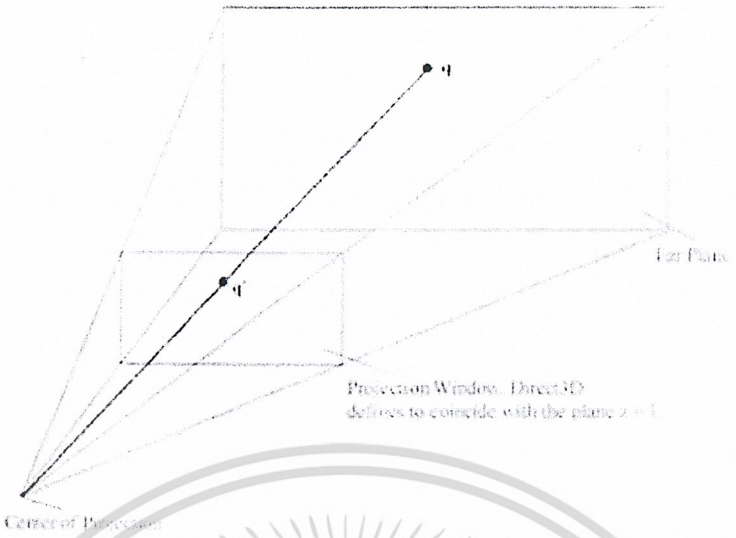
6.6.5 Lighting เป็นการคำนวณค่าสีต่างๆ ให้กับเวอร์เท็กซ์ เช่น สีที่ได้จากเท็กเซล (Texel) ของเท็กซ์เจอร์ (Texture) ที่นำมาลงให้ตรงกับสีของจุดเวอร์เท็กซ์นั้นๆ หรือสีจากแหล่งกำเนิดแสงต่างๆ เป็นต้น

6.6.6 Clipping เป็นขั้นตอนการตัดวัตถุหรือส่วนของวัตถุที่อยู่ภายนอกขอบเขตการมองเห็น (Viewing Volume) ออกไป



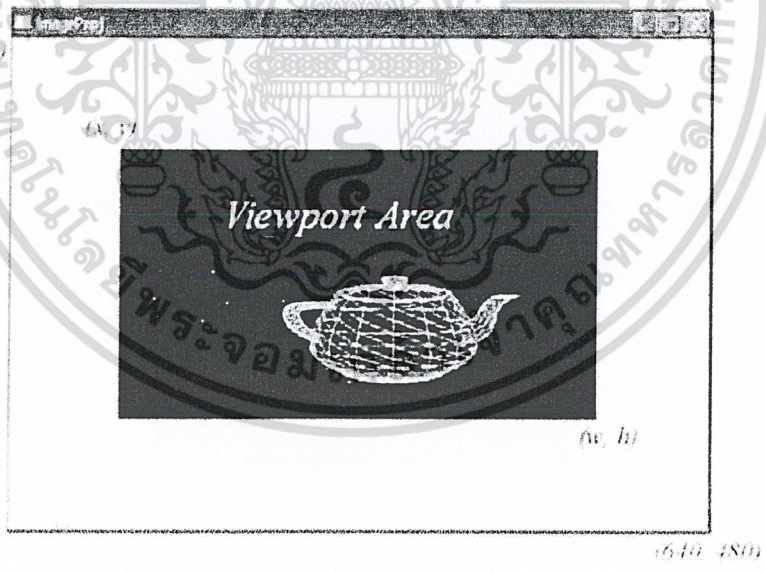
รูปที่ 6.13 แสดงขั้นตอนการตัดวัตถุหรือส่วนของวัตถุที่อยู่ภายนอกขอบเขตการมองเห็น

6.6.7 Projection เป็นขั้นตอนในการทำให้วัตถุที่อยู่ใน 3 มิติ เกิดเป็นภาพ 2 มิติบนฉาก ซึ่งการทำโปรเจคชันนั้น มีอยู่หลายแบบ เช่น Axonometric, Oblique, Perspective เป็นต้น โดยรูปที่ 6.14 เป็นการทำโปรเจคชันแบบ Perspective



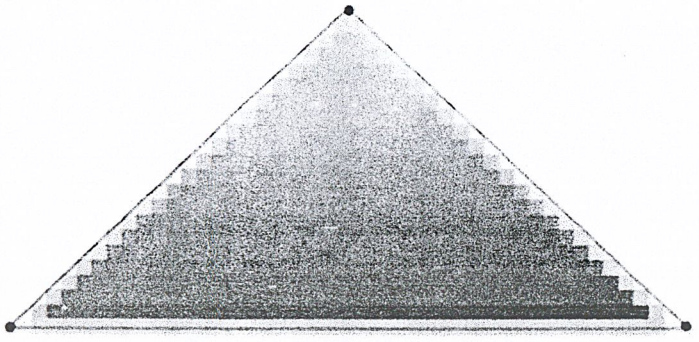
รูปที่ 6.14 การทำโปรเจคชันแบบ Perspective

6.6.8 Viewport Transform เป็นขั้นตอนการแปลงจากระบบพิกัดของโปรเจคชันวินโดว์ (Projection window) ที่เกิดจากการทำโปรเจคชัน ให้กลายเป็นระบบพิกัดของฉากสี่เหลี่ยมบนจอ ที่เรียกว่า Viewport



รูปที่ 6.15 การแปลงระบบพิกัดของโปรเจคชันวินโดว์ให้เป็นของฉากสี่เหลี่ยมบนจอ

6.6.9 Rasterization เป็นขั้นตอนการใช้เทคนิคต่าง ๆ มาปรับคุณภาพของรูปให้ดูเรียบและสวยงามขึ้น



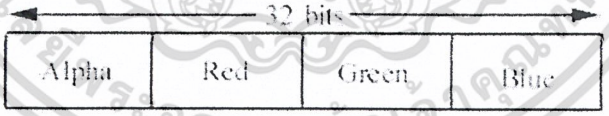
รูปที่ 6.16 แสดงการทำ Rasterization

6.7 เทคนิคเกี่ยวกับไดเร็กซ์สามดีที่นำมาใช้

6.7.1 การให้สี (Color)

ในไดเร็กซ์ทรีดี จะใช้รูปแบบสีแบบ RGB โดยใช้แม่สี 3 สี คือ แดง เขียว และน้ำเงิน มารวมกันด้วยความเข้มของแม่สีที่แตกต่างกันจนได้มาซึ่งสีที่ต้องการ สำหรับโครงสร้างของสี RGB ในไดเร็กซ์ทรีดีจะมีอยู่ด้วยกัน 2 แบบ

6.7.1.1 D3DCOLOR ซึ่งจะมีขนาด 32 บิต โดยแม่สีแต่ละสีจะแทนด้วยข้อมูลขนาด 8 บิต โดยที่บางรูปแบบจะมีค่าความทึบแสงและโปร่งแสงด้วย เรียกค่านี้ว่า อัลฟา (Alpha)



รูปที่ 6.17 แสดงรูปแบบการให้สีแบบ D3DCOLOR

6.7.1.2 D3DCOLORVALUE โดยโครงสร้างนี้จะใช้การกำหนดค่าให้กับแม่สีโดยใช้ชนิดข้อมูล float ซึ่งค่าที่กำหนดนี้จะอยู่ในช่วงระหว่าง 0.0 ถึง 1.0

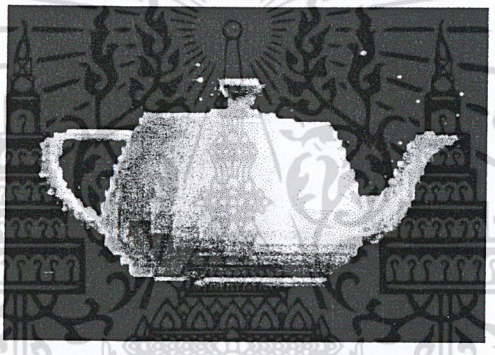
6.7.2 สีของเวอร์เท็กซ์

สีของ Primitive ต่าง ๆ เช่น รูปสามเหลี่ยมจะถูกกำหนดโดยสีของเวอร์เท็กซ์ ดังนั้น ในการสร้างสีให้กับวัตถุสามมิติใด ๆ จะต้องกำหนดสีให้กับเวอร์เท็กซ์ของวัตถุนั้น ๆ เสียก่อน

6.7.3 การเกลี่ยสี (Shading Mode)

รูปแบบในการเกลี่ยสี ใช้สำหรับการเรนเดอร์โพลีกอนให้มีผลกระทบจากแสงอย่างทั่วถึง ซึ่งรูปแบบในการเกลี่ยสี จะกำหนดจากความหนาแน่นของสีและแสงที่จุดใดๆ บน พื้นผิวของโพลีกอน สำหรับไมโครซอฟท์ ไดเร็กซ์ทรีดีสนับสนุนการให้แสงเงา 2 รูปแบบ ได้แก่

6.7.3.1 การเกลี่ยสีแบบแบนราบ (Flat Shading) ในรูปแบบการเกลี่ยสีแบบแบนราบ ไดเร็กซ์ทรีดี จะทำการเรนเดอร์โพลีกอนโดยใช้สีของวัสดุ (material) ของโพลีกอน ที่จุด (vertex) แรก ให้กับโพลีกอนทั้งหมด วัตถุ 3 มิติ ที่มีการเรนเดอร์ด้วยรูปแบบการเกลี่ยสีแบบแบนราบ จะมีขอบอย่างชัดระหว่างโพลีกอน ถ้าหากว่าเป็นคนละระนาบ ดังรูปที่ 6.18 เป็นการเกี่ยสีแบบแบนราบให้กับกาน้ำชา ซึ่งจะเห็นได้ว่าจะเห็นขอบของแต่ละโพลีกอนอย่างชัดเจน แต่การเกลี่ยสีแบบนี้ จะมีข้อดีคือใช้เวลาในการคำนวณผลน้อยสุด



รูปที่ 6.18 แสดงการเกลี่ยสีแบบแบนราบให้กับกาน้ำชา

6.7.3.2 รูปแบบการเกลี่ยสีแบบเการาด์ (Gouraud Shading) เมื่อไดเร็กซ์ทรีดีเรนเดอร์โพลีกอนโดยใช้รูปแบบการเกลี่ยสีแบบเการาด์ จะคำนวณสีสำหรับแต่ละจุดโดยใช้จุดปกติและค่าตัวแปรของแสง จากนั้นแทรกสีไปผ่านผิวหน้าของโพลีกอน การแทรกนี้จะทำแบบเป็นเชิงเส้น ตัวอย่างเช่น ถ้ามีส่วนประกอบที่เป็นสีแดงของสีในจุดแรก เท่ากับ 0.8 และส่วนประกอบที่เป็นสีแดงของจุด 2 เท่ากับ 0.4 ใช้การเกลี่ยสีแบบเการาด์และโครงสร้างสี RGB จะกำหนดค่าส่วนประกอบสีแดงได้เท่ากับ 0.6 ให้กับจุดที่ตำแหน่งกลางของเส้นระหว่างจุด (vertice) ทำให้สีเรียบเนียนกว่าการเกลี่ยสีแบบแบนราบ

รูปที่ 6.19 แสดงการเกลี่ยสีแบบเการาด์ให้กับกาน้ำชา

6.7.4 การให้แสง (Lighting)

ในการทำให้ภาพที่ได้ออกมามีความสมจริงเพิ่มมากขึ้นนั้น จะต้องมีการกำหนดแหล่งกำเนิดแสง เพื่อให้แสงกับวัตถุ โดยการคำนวณสีของวัตถุในกรณีที่มีแสงมาเกี่ยวข้องนั้น จะต้องคำนวณจาก 3 ปัจจัยหลัก ได้แก่

- แหล่งกำเนิดแสง (light source)
- ลักษณะของวัตถุ (material)
- ทิศทางการหันหน้าของวัตถุที่ทำกับแหล่งกำเนิดแสง (orientation of surface to light source)

ในไดเร็กซ์ทรีดีนั้น จะกำหนดแสงที่ออกมาจากแหล่งกำเนิดแสงด้วยส่วนประกอบ 3 อย่าง คือ

6.7.4.1 Ambient Light เป็นส่วนประกอบของแสงที่ให้ความสว่างกับฉากทั้งหมดอย่างเท่าเทียมกันโดยไม่ขึ้นกับระยะห่างจากแหล่งกำเนิดแสง และไม่ขึ้นกับทิศทางการหันหน้าของวัตถุที่ทำกับแหล่งกำเนิดแสง

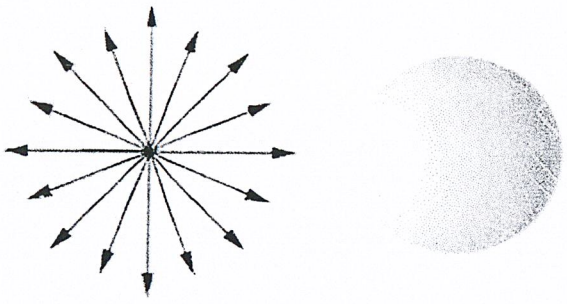
6.7.4.2 Diffuse Light เป็นส่วนประกอบของแสงที่เมื่อตกกระทบวัตถุแล้วจะสะท้อนออกไปทุกทิศทุกทาง ดังนั้นแสงแบบนี้จะขึ้นกับทิศทางของแสงและลักษณะของวัตถุเท่านั้น ไม่ขึ้นกับตำแหน่งของกล้อง

6.7.4.3 Specular Light เป็นส่วนประกอบของแสงที่เมื่อตกกระทบวัตถุแล้วจะสะท้อนออกไปในทิศทางใดทิศทางหนึ่งเท่านั้น โดยแสงแบบนี้จะทำให้เห็นแสงสว่างได้ในบางมุม ดังนั้นจะต้องนำทั้งตำแหน่งของกล้อง ทิศทางของแสง และลักษณะของวัตถุมาคำนวณ

6.7.5 ชนิดของแหล่งกำเนิดแสง

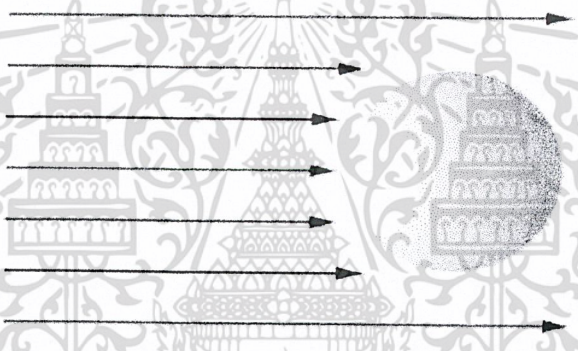
ไดเร็กซ์ทรีดีมีชนิดของแหล่งกำเนิดแสงอยู่ 3 ชนิด ได้แก่

6.7.5.1 แหล่งกำเนิดแสงแบบจุด (Point light) เป็นแหล่งกำเนิดแสงที่ต้องระบุตำแหน่งแหล่งกำเนิดแสงแบบจุดนี้จะส่องแสงออกมาทุกทิศทุกทาง



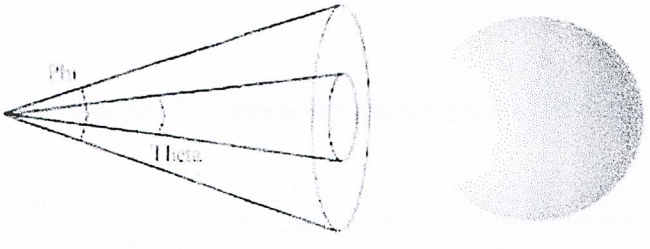
รูปที่ 6.20 แหล่งกำเนิดแสงแบบจุด (Point light)

6.7.5.2 แหล่งกำเนิดแสงแบบขนาน (Directional light) เป็นแหล่งกำเนิดแสงที่ไม่มีตำแหน่ง และส่องแสงเป็นลำแสงขนานในทิศทางใดทิศทางหนึ่งตามที่กำหนด



รูปที่ 6.21 แหล่งกำเนิดแสงแบบขนาน (Directional light)

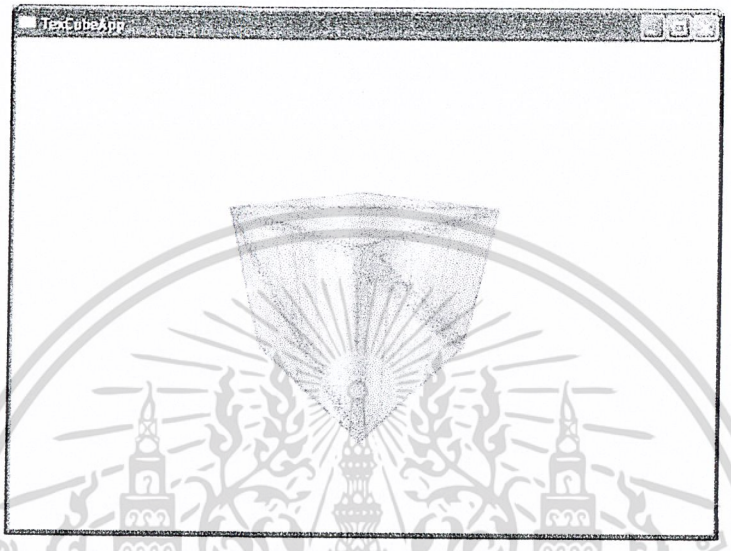
6.7.5.3 แหล่งกำเนิดแสงแบบสปอตไลท์ เป็นแหล่งกำเนิดแสงที่มีตำแหน่ง และส่องแสงออกมาเป็นรูปกรวยในทิศทางที่กำหนด โดยกรวยแสงจะถูกระบุด้วยค่า 2 ค่า ได้แก่ มุม θ (theta) และมุม φ (phi) โดยมุม θ จะใช้เป็นมุมของกรวยใน และมุม φ จะเป็นมุมของกรวยนอก



รูปที่ 6.22 แหล่งกำเนิดแสงแบบสปอตไลท์

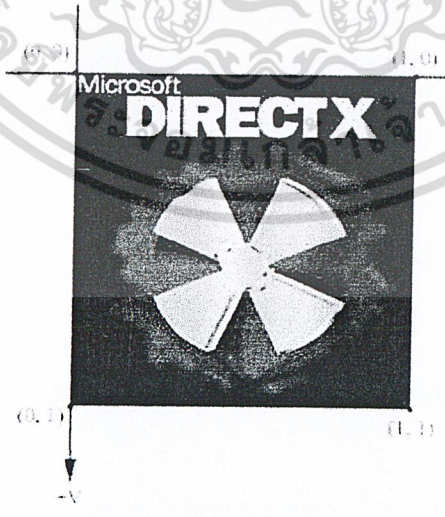
6.7.6 การแปะภาพเข้ากับวัตถุ (Texturing)

Texturing คือ เทคนิคการนำเอาภาพมาแปะติดเข้ากับวัตถุ 3 มิติ ทำให้วัตถุมีสีสันที่สวยงามและดูสมจริงมากขึ้น รูปที่ 6.23 เป็นการนำเทคนิคนี้มาใช้สร้างลังที่มีความสมจริงมากขึ้น โดยแปะภาพของลังไว้ที่แต่ละด้านของลูกบาศก์ที่สร้างขึ้นมา



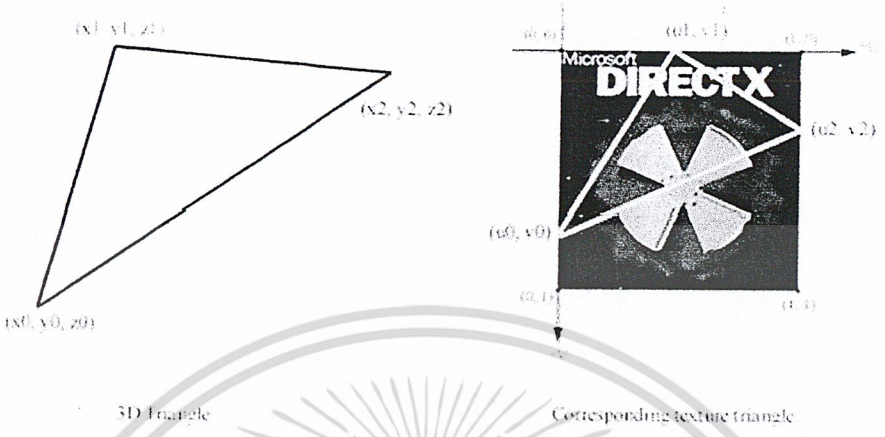
รูปที่ 6.23 การแปะภาพเข้ากับแต่ละด้านของวัตถุ

ไดเร็กซ์ทรีดีจะใช้ระบบพิกัดของการทำการแปะภาพที่ประกอบด้วยแกน U ในแนวนอน และแกน V ในแนวตั้ง โดยทั้งสองแกนจะมีค่าตั้งแต่ 0 ถึง 1



รูปที่ 6.24 แสดงระบบพิกัดของการทำการแปะภาพ

ในการกำหนด Texture ให้กับสามเหลี่ยมแต่ละรูปนั้น จะต้องกำหนดตำแหน่งของแกน U และแกน V ของ Texture ที่ใช้ให้กับเวอร์เท็กซ์แต่ละเวอร์เท็กซ์ของรูปสามเหลี่ยมด้วย



รูปที่ 6.25 การกำหนดตำแหน่งของแกน U และ V ให้กับเวอร์เท็กซ์ของวัตถุ

6.8 การนำไฟล์นามสกุล .x มาใช้ในแอพพลิเคชัน

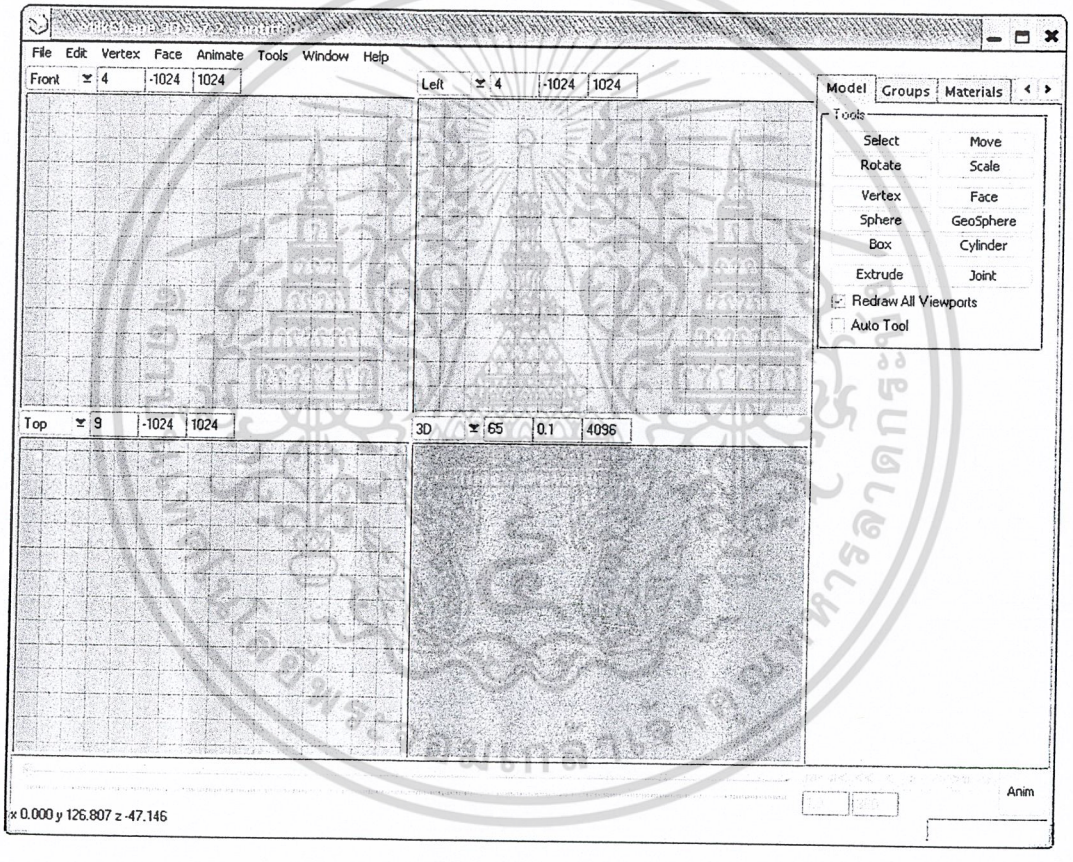
ไฟล์ที่มีนามสกุล .x เป็นไฟล์ที่เกิดมาจากการสร้างรูป 3 มิติโดยใช้โปรแกรม 3 มิติ ต่าง ๆ เช่น ทรีดีสตูดิโอ แม็กซ์ (3D Studio Max), มายา (Maya), ไลท์เวฟ 3 มิติ (LightWave3D) ฯลฯ แล้วแปลงรูปแบบการจัดเก็บข้อมูลมาเป็นรูปแบบไฟล์นามสกุล .x

โปรแกรม 3 มิติแต่ละโปรแกรมมีรูปแบบการจัดเก็บข้อมูลของเวอร์เท็กซ์แต่ละจุดที่ประกอบเป็นวัตถุ รวมถึงระบบพิกัดของไฟล์แต่ละนามสกุลนั้นมีความแตกต่างกัน ดังนั้นก่อนการนำวัตถุใดๆ ที่สร้างได้จากโปรแกรม 3 มิติมาใช้ในแอพพลิเคชันจะต้องมีการแปลงไฟล์ให้มาเป็นไฟล์นามสกุล .x ก่อน

บทที่ 7
โปรแกรมสร้างโมเดล 3 มิติ มิลค์เชพ 3 มิติ (Milkshape 3D)

7.1 มิลค์เชพ 3 มิติ (Milkshape 3D)

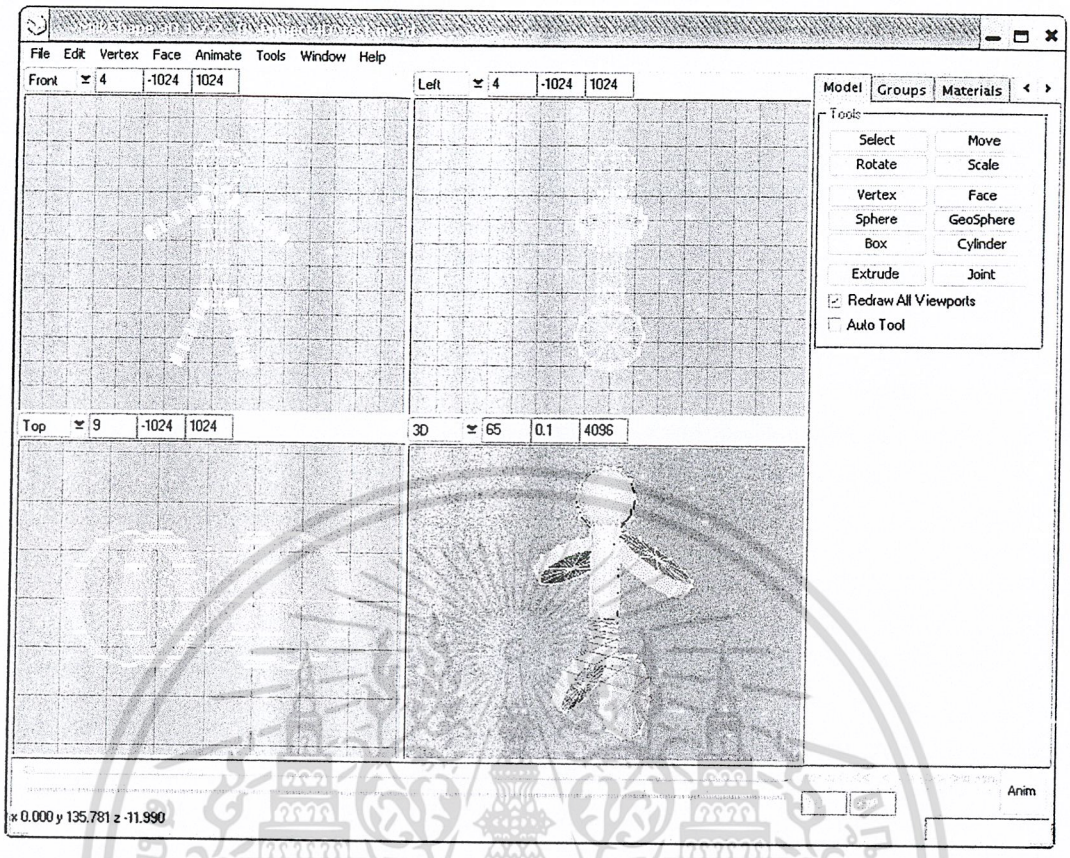
โครงงานนี้เลือกใช้โปรแกรมมิลล์เชฟในการสร้างตัวละครที่มีการเคลื่อนไหว เนื่องจากโปรแกรมนี้มีจุดเด่นอยู่ที่สามารถจัดการกับแอนิเมชันต่างๆ ของตัวละครได้เป็นอย่างดี อีกทั้งยังสามารถเอ็กซ์พอร์ตไฟล์โมเดลที่สร้างจากโปรแกรมนี้ออกมาได้หลายฟอร์แมท ซึ่งสามารถนำไปใช้ใน DirectX SDK ได้ จึงเหมาะสมที่จะใช้ทำโมเดลตัวละคร



รูปที่ 7.1 แสดงหน้าจอการทำงานของโปรแกรมมิลล์เชฟ 3 มิติ

7.2 การสร้างรูปโมเดล 3 มิติ

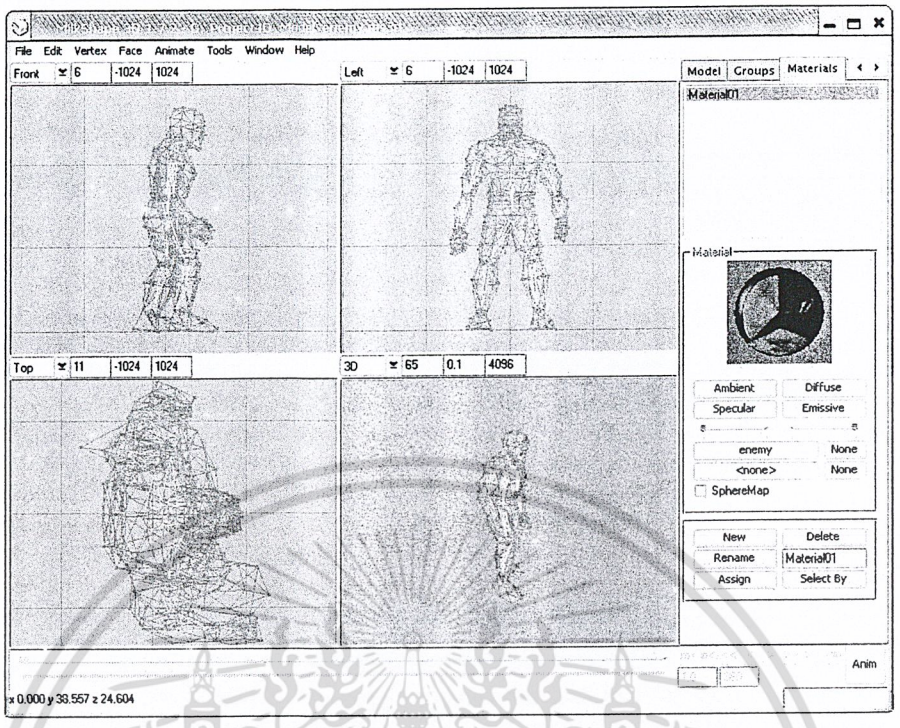
การสร้างโมเดล 3 มิติสามารถสร้างได้โดยอิสระจากการวาด 3 เหลี่ยมหลายๆอันมาต่อกันเป็น face ของวัตถุ หรือสามารถเริ่มจากรูปทรงเรขาคณิตที่ใกล้เคียงกับโมเดลที่ต้องการแล้วค่อยทำการปรับแต่งรูปร่างให้ได้โมเดลตามต้องการ โดยการปรับขนาด ปรับตำแหน่งจุดเวอร์เท็กซ์ เป็นต้น



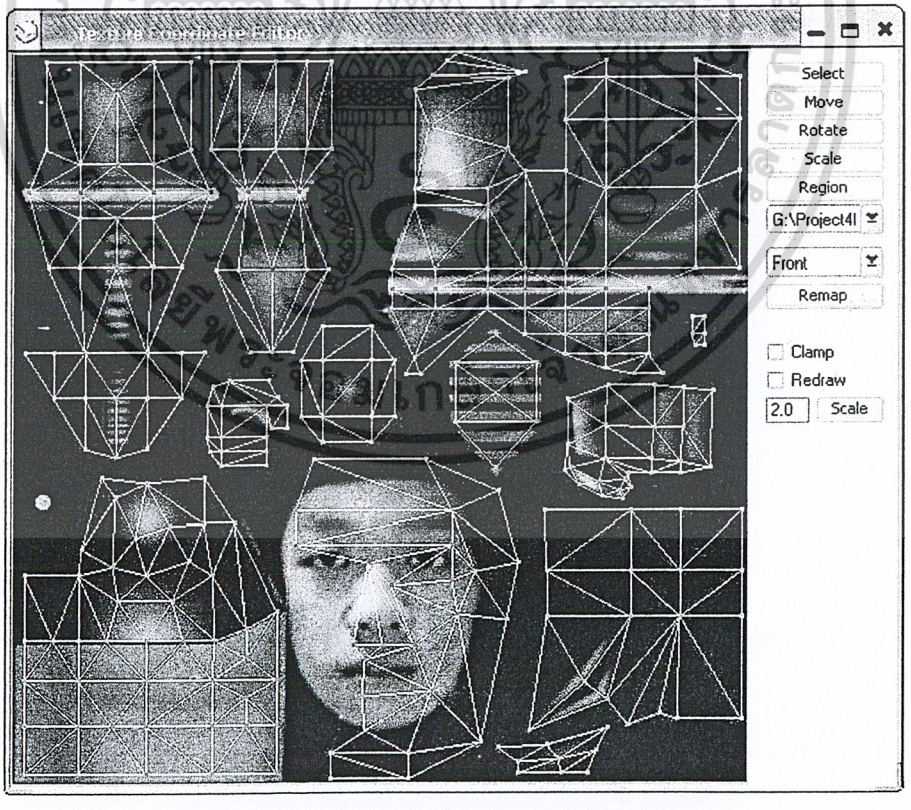
รูปที่ 7.2 แสดงการสร้างโมเดลอย่างง่ายจากรูปทรงเรขาคณิต

7.3 การกำหนดพื้นผิว (Texture Mappting)

การกำหนดพื้นผิวในโปรแกรมมิลค์เชพ 3 มิติ นั้นสามารถทำได้อย่างอิสระเพราะแต่ละ face จะมีข้อมูลของวัตถุ (material) และ โคออดิเนต (coordinate) เป็นของตัวเอง นอกจากนี้เรายังสามารถกำหนดให้พื้นผิวแต่ละส่วนมีวัตถุมาจากภาพบิตแมบภาพเดียวก็ได้ โดยการจัดโคออดิเนตของพื้นผิวแต่ละส่วนลงบนภาพตามต้องการซึ่งสามารถทำได้อย่างอิสระ ซึ่งโปรแกรมมิลค์เชพ 3 มิติ มีเครื่องมือชื่อว่า "Texture Coordinate Editor" มาช่วยในการกำหนดพื้นผิวให้วัตถุ



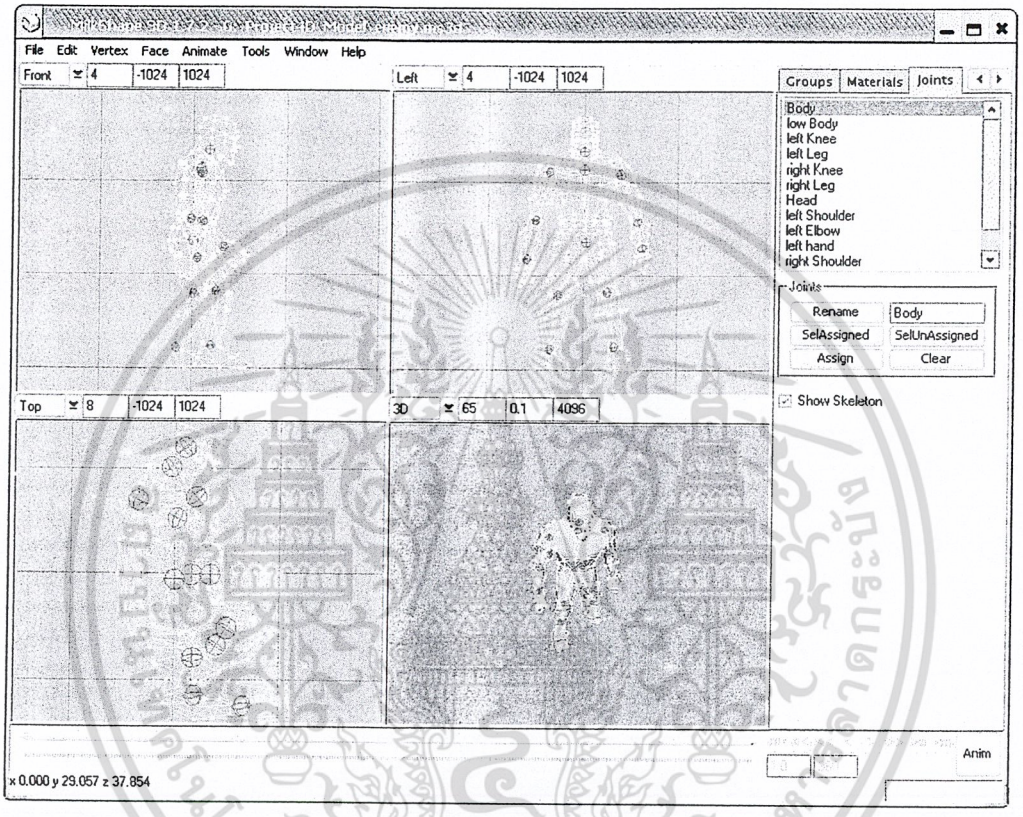
รูปที่ 7.3 แสดงการกำหนดพื้นผิวให้กับวัตถุซึ่งเป็นโมเดลศัตรู



รูปที่ 7.4 แสดงเครื่องมือ Texture Coordinate Editor ในการกำหนดพื้นผิวให้วัตถุ

7.4 การกำหนดกระดูกและข้อต่อให้โมเดล 3 มิติ

หากต้องการทำโมเดล 3 มิติให้เคลื่อนไหวได้ เราต้องทำการบันทึกการเคลื่อนไหวของตัวละครเอาไว้ก่อน แล้วจึงนำมาแสดงตามเงื่อนไขต่างๆ ในโปรแกรม ในโปรแกรมมิลค์เชพ 3 มิติ หากต้องการทำการเคลื่อนไหวของตัวละคร 3 มิติ จำปีนต้องทำการกำหนดกระดูกและข้อต่อให้โมเดลเสียก่อนว่ากระดูกส่วนไหนเชื่อมต่อกับโมเดลส่วนใด



รูปที่ 7.5 แสดงการกำหนดข้อต่อและกระดูกให้กับโมเดล 3 มิติ

7.5 การจัดการแอนิเมชันของโมเดล 3 มิติ

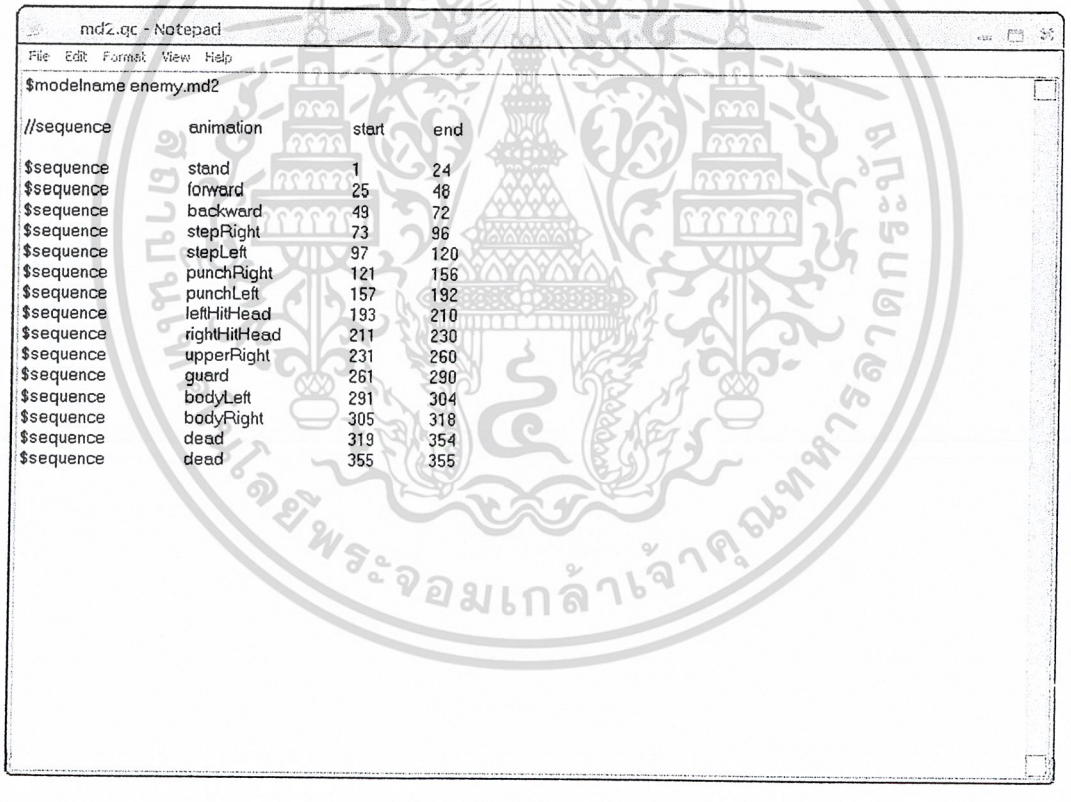
เมื่อเราทำการกำหนดกระดูกและข้อต่อของโมเดลทั้งหมดแล้ว เราสามารถบันทึกการเคลื่อนไหวของโมเดล 3 มิติได้ โดยการกำหนดจำนวนเฟรมขึ้นมาก่อนว่าเราจะให้มีการเคลื่อนไหวทั้งหมดกี่เฟรม ต่อจากนั้นก็กำหนดคีย์เฟรมขึ้นมาในเฟรมที่โมเดลมีการเปลี่ยนแปลงลักษณะท่าทาง เมื่อทำการรันแอนิเมชันโปรแกรมจะทำการคำนวณการเคลื่อนไหวของโมเดลระหว่างคีย์เฟรมให้เองโดยอัตโนมัติ

รูปที่ 7.6 ส่วนที่ใช้ในการทำแอนิเมชันของโมเดล 3 มิติ

7.6 การเอ็กซ์พอร์ตเพื่อนำไปใช้ใน DirectX SDK

DirectX SDK สามารถสร้างคลาสที่รองรับการทำงานของไฟล์โมเดล 3 มิติที่มีนามทสกุล .md2 ซึ่งมีการบันทึกทั้งรูปร่างโมเดลและแอนิเมชันการเคลื่อนไหวทั้งหมดไว้ภายในไฟล์ รูปแบบไฟล์ .md2 ทำการเรียงต่อแอนิเมชันทั้งหมดเป็นแอนิเมชันเดียว โดยแต่ละเฟรมจะมีชื่อเฟรมอยู่ เฟรมที่มีชื่อนำหน้าเหมือนกันจะถูกจัดให้อยู่ในกลุ่มเดียวกัน ถือเป็นแอนิเมชันหนึ่งท่า เช่น ท่ายืน มีชื่อเรียกคือ “stand” มีแอนิเมชันในช่วงเฟรมที่ 1-24 ของไฟล์สกุล .md2 ก็จะมีชื่อเฟรมที่ 1-24 คือ “stand01”, “stand02”, stand03”, …, “stand24” หรือ ท่าชกหมัดขวา มีชื่อเรียกคือ “punchRight” มีแอนิเมชันในช่วงเฟรมที่ 121-156 ของไฟล์สกุล .md2 ก็จะมีชื่อเฟรมที่ 121-156 คือ “punchRight01”, “punchRight02”, …, “punchRight36”

ในโปรแกรมมิลค์เชพ 3 มิติ นั้นสามารถเอ็กซ์พอร์ตโมเดล 3 มิติในรูปแบบไฟล์สกุล .md2 ได้ โดยจะต้องทำการเขียน QC ไฟล์ ชื่อ md2.qc เพื่อกำหนดชื่อเฟรมแต่ละเฟรม ส่วนแอนิเมชันให้ทำการเรียงต่อกันทั้งหมดเป็นแอนิเมชันเดียว ตัวอย่างไฟล์ md2.qc แสดงไว้ดังรูปที่ 7.7



รูปที่ 7.7 ตัวอย่างไฟล์ md2.qc

สำหรับคำสั่งในไฟล์ md2.qc นั้นมีความหมายดังนี้

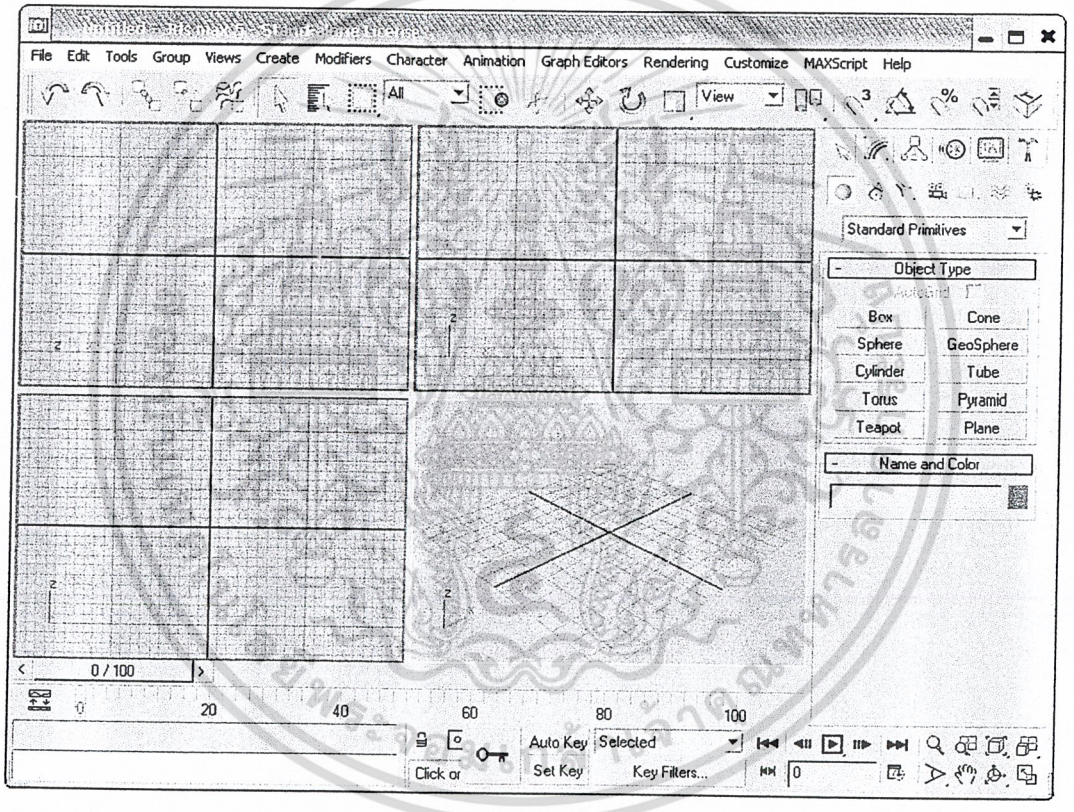
&modelname <ชื่อไฟล์โมเดล>	เป็นการกำหนดชื่อไฟล์ของโมเดลที่จะทำการเอ็กซ์พอร์ต
&sequence<ชื่อแอนิเมชัน><เฟรมเริ่มต้น><เฟรมสิ้นสุด>	เป็นการกำหนดเฟรมให้แอนิเมชันว่ามีช่วงอยู่ในเฟรมที่เท่าไรบ้าง โดยเมื่อทำการเอ็กซ์พอร์ตเสร็จแล้ว แต่ละเฟรมจะมีชื่อเป็น <ชื่อแอนิเมชัน><หมายเลขเฟรมที่>
// <ข้อความ>	เป็นหมายเหตุ

ตารางที่ 7.1 แสดงคำสั่งในไฟล์ md2.qc

บทที่ 8
โปรแกรมสร้างโมเดล 3 มิติ ทรีดีสตูดิโอ แม็กซ์ (3DStudio Max)

8.1 ทรีดีสตูดิโอ แม็กซ์ (3DStudio Max)

โครงงานนี้เลือกใช้โปรแกรมทรีดีสตูดิโอ แม็กซ์ ในการสร้างโมเดลที่ต้องการความละเอียด เนื่องจากโปรแกรมนี้มีจุดเด่นอยู่ที่การมีเครื่องมือหลายชนิดให้จัดการกับโมเดล 3 มิติที่มีความซับซ้อนได้อย่างสมบูรณ์ อีกทั้งยังสามารถเอ็กซ์พอร์ตไฟล์โมเดลที่สร้างจากโปรแกรมนี้ออกมาไปใช้ใน DirectX SDK ได้ จึงเหมาะสมที่จะใช้ทำโมเดลตัวละครที่ต้องการความละเอียดสูง

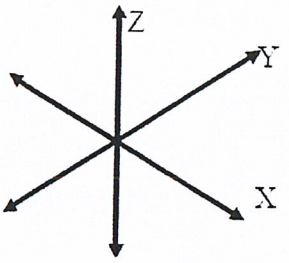


รูปที่ 8.1 แสดงหน้าจอการทำงานของโปรแกรมทรีดีสตูดิโอ แม็กซ์

8.2 ความรู้พื้นฐานเกี่ยวกับโลก 3 มิติของโปรแกรมทรีดีแม็กซ์

8.2.1 แกนอ้างอิงในการสร้างวัตถุ 3 มิติ

โปรแกรมทรีดีสตูดิโอ แม็กซ์ ใช้ระบบคาร์ทีเซียน (Cartesian) ซึ่งอ้างอิงจุด 3 จุดในแนวแกน X, Y และ Z โดยจุดเริ่มต้นจะมีพิกัด (0,0,0) เราสามารถทราบตำแหน่งของวัตถุ 3 มิติได้โดยวัดกับจุดนี้

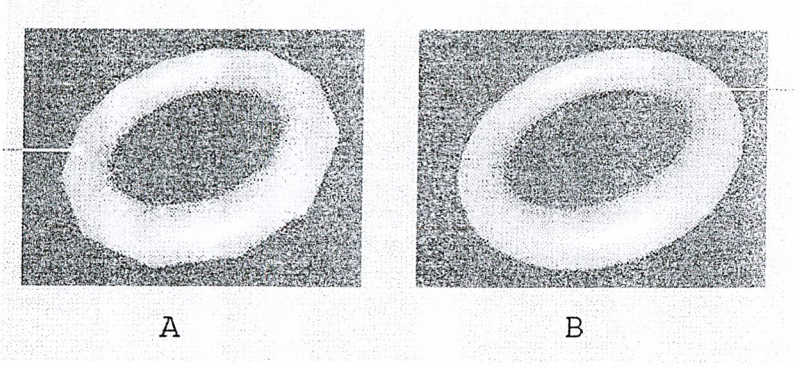


รูปที่ 8.2 ระบบพิกัด 3 มิติ

8.2.2 โพลีกอน (Polygon)

โพลีกอนคือ แผ่นระนาบ 2 มิติที่มีหลายเหลี่ยม และเป็นชิ้นส่วนที่โปรแกรมนำมาต่อกันจนเกิดเป็นรูปร่างต่าง ๆ ของวัตถุที่เราต้องการสร้าง ดังนั้นถ้าเราใช้จำนวนโพลีกอนมาก จะได้ภาพที่ละเอียดเสียเวลาคำนวณมาก ถ้าเราใช้จำนวนโพลีกอนน้อย จะได้ภาพความละเอียดน้อย ประหยัดเวลาคำนวณ

รูปที่ 8.3 แสดงการนำโพลีกอนมาต่อกันจนเป็นวัตถุแบบต่าง ๆ



รูปที่ 8.4 ภาพ A ใช้โพลีกอนน้อย ส่วนภาพ B ใช้โพลีกอนมาก

8.3 ความสามารถในการทำงานสามมิติของโปรแกรมทรีดีสตูดิโอ แม็กซ์

• **ความสามารถในการสร้างวัตถุรูปทรงต่าง ๆ** โปรแกรมทรีดีสตูดิโอ แม็กซ์ ได้พัฒนาชุดคำสั่งและเครื่องมือในการสร้างวัตถุรูปแบบต่าง ๆ ขึ้นมาให้เลือกใช้งานมากมาย ไม่ว่าจะเป็นเครื่องมือในการสร้างวัตถุรูปทรงเรขาคณิต หรือการสร้างวัตถุที่มีรูปร่างซับซ้อน เช่น ตัวการ์ตูน ฯลฯ

• **ความสามารถในการกำหนดพื้นผิวให้วัตถุ** การกำหนดพื้นผิวเป็นอีกส่วนหนึ่งที่ช่วยให้วัตถุสามมิติสวยงามและเหมือนจริง โปรแกรมมีชุดคำสั่งในการกำหนดพื้นผิวมากมาย ทำให้การกำหนดพื้นผิวให้กับวัตถุเป็นเรื่องง่าย

• **ความสามารถในการเรนเดอร์ (Render)** การเรนเดอร์คือ การประมวลผลภาพของวัตถุที่สร้างขึ้นมาทั้งหมด ทำให้เราเห็นรายละเอียดของวัตถุได้ทั้งพื้นผิว แสงและเงา การเรนเดอร์ภาพในโปรแกรมทรีดีสตูดิโอ แม็กซ์ ได้พัฒนาขึ้นจนสามารถกำหนดองค์ประกอบต่างๆ ของภาพได้เหมือนจริงมาก ไม่ว่าจะเป็นเรื่องความละเอียดของพื้นผิว แสงและเงาที่เกิดขึ้น นอกจากนี้ยังสามารถใส่เอ็ฟเฟ็ค (Effect) เพิ่มเติมเข้าไปในขณะเรนเดอร์เช่น การใส่ประกายแสงลงบนภาพ

• **ความสามารถในการสร้างภาพเคลื่อนไหว** ทรีดีสตูดิโอ แม็กซ์ได้นำคำสั่งต่าง ๆ ในการกำหนดการเคลื่อนไหวให้กับวัตถุที่ต้องการมาให้เราใช้งานได้อย่างสะดวกสบายมากขึ้น เช่น กำหนดให้วัตถุเคลื่อนไหวตามแรงดึงดูดของโลก หรือคำนวณทิศทางของวัตถุที่เกิดจากการชนกัน เพียงแค่เรากำหนดทิศทางที่ต้องการให้เกิดแรงดึงดูดและแรงในการชนเท่านั้น ที่เหลือโปรแกรมจะคำนวณให้โดยอัตโนมัติ

8.4 ประเภทของภาพวัตถุ 3 มิติของโปรแกรมทรีดีแม็กซ์

ประเภทของวัตถุ 3 มิติ แบ่งออกเป็น 2 ประเภท

8.4.1 วัตถุ 3 มิติรูปทรงพื้นฐาน (Primitive Object) สร้างโดยใส่ขนาดตามความต้องการจากคีย์บอร์ดหรือใช้เมาส์คลิกแล้วลากตามความต้องการ เช่น กล่องสี่เหลี่ยม ทรงกลม ทรงกระบอก กาน้ำโดนัท ฯลฯ

8.4.2 วัตถุ 3 มิติจากเส้น 2 มิติ แบ่งได้เป็น

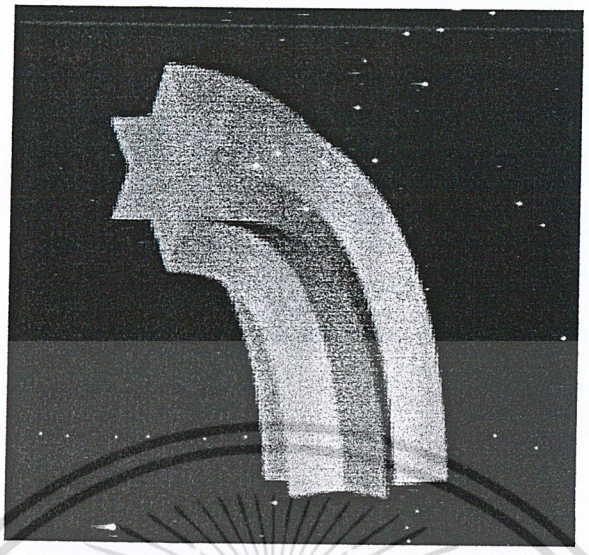
8.4.2.1 วิธีการเอ็กซ์ทรูดส์ (Extrude) คือ เป็นคำสั่งที่ใช้เพิ่มส่วนสูงให้กับเส้น 2 มิติที่สร้าง สามารถกำหนดและแก้ไขส่วนสูงได้ตามต้องการ

รูปที่ 8.5 การสร้างวัตถุ 3 มิติด้วยวิธีการ Extrude

8.4.2.2 วิธีการแล็ธส์ (Lathe) เป็นคำสั่งในการสร้างวัตถุ 3 มิติ จากเส้น 2 มิติ โดยการเขียนเส้นด้านข้างของวัตถุ แล้วนำมาหมุนในแกนอ้างอิงที่กำหนดจนเกิดเป็นรูปทรง 3 มิติ

รูปที่ 8.6 การสร้างวัตถุ 3 มิติด้วยวิธีการ Lathe

8.4.2.3 วิธีการล็อฟท์ (Loft) เป็นการนำเส้น 2 มิติ มาสร้างเป็นวัตถุ 3 มิติ ด้วยการกำหนดให้เส้น 2 มิติ รูปทรงต่าง ๆ ที่สร้างขึ้นเพื่อเป็นวัตถุต้นแบบ วิ่งไปตามเส้น 2 มิติอีกเส้นหนึ่ง ซึ่งเรียกเส้นนี้ว่า พาธ (Path) เพื่อให้ได้รูปทรงตามที่เราต้องการ



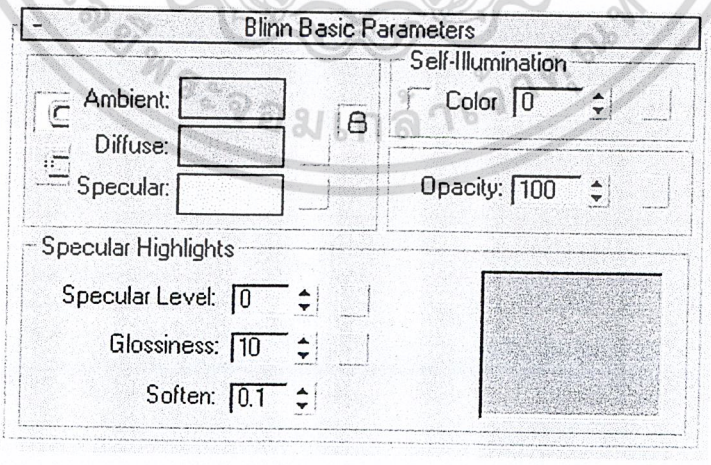
รูปที่ 8.7 การสร้างวัตถุ 3 มิติด้วยวิธีการ Loft

8.5 การกำหนดพื้นผิวให้กับวัตถุ 3 มิติ

โดยปกติพื้นผิวของวัตถุ 3 มิติ แบ่งออกเป็น 3 ส่วน คือ

- Specular คือ บริเวณพื้นผิวที่แสงจ้าที่สุด
- Diffuse คือ บริเวณที่พื้นผิวถูกแสงไล่เฉดสีตามปริมาณของแสง
- Ambient คือ บริเวณด้านที่ไม่ถูกแสง

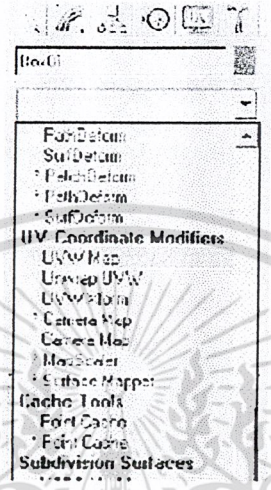
นอกจากนั้นยังมีการกำหนดคุณสมบัติทางกายภาพที่มีต่อแสงของวัตถุ เช่น ความเงา ความมันวาว ความโปร่งใสของวัตถุ เป็นต้น



รูปที่ 8.8 แสดงการกำหนดพื้นผิวให้กับวัตถุ 3 มิติ

8.6 การแปะภาพ 2 มิติลงบนวัตถุ 3 มิติ

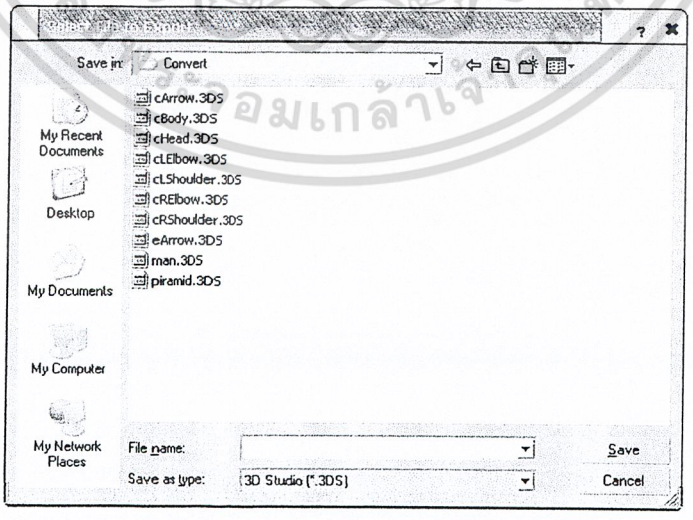
แนวแกนในการแปะภาพลงบนวัตถุจะไม่เหมือนแกนในการสร้างวัตถุ 3 มิติทั่วไป การแปะภาพลงบนวัตถุจะใช้แกนอ้างอิง U, V และ W แทนแกน X, Y และ Z ในการทำงาน



รูปที่ 8.9 แสดงการกำหนดการแปะภาพ 2 มิติลงบนวัตถุ 3 มิติ

8.7 การนำไฟล์เข้าและการนำไฟล์ออก (Import and Export file)

การนำไฟล์เข้าและการนำไฟล์ออกมีไว้เพื่อให้โปรแกรมทรีดีสตูดิโอ แม็กซ์ สามารถติดต่อกับโปรแกรมสามมิติอื่น ๆ ได้ เช่น ออโต้แค็ด (AutoCad) ฯลฯ การนำไฟล์ออกจากโปรแกรมทรีดีสตูดิโอ แม็กซ์ จะแปลงจากไฟล์สกุล .max ไปเป็นไฟล์สกุล .3ds

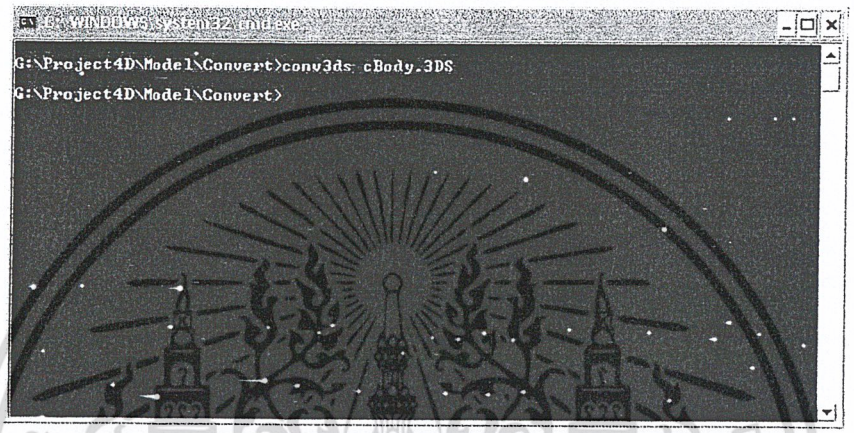


รูปที่ 8.10 แสดงการนำเอ็กซ์พอร์ตไฟล์สกุล .3ds

8.8 การแปลงไฟล์สกุล .3ds เป็นไฟล์สกุล .x

การแปลงไฟล์สกุล .3ds ให้เป็นไฟล์สกุล .x มีความจำเป็นเพราะไฟล์ทั้ง 2 สกุลมีการจัดเก็บเวอร์เท็กซ์ของโมเดลต่างรูปแบบกัน จึงต้องใช้โปรแกรม conv3ds ที่มีมากับโปรแกรมทรีดีสตูดิโอ แม็กซ์ ในการแปลงการจัดเก็บเวอร์เท็กซ์ของโมเดล เพื่อนำไฟล์สกุล .x ไปใช้ใน DirectX SDK

การแปลงทำได้โดยเรียกใช้โปรแกรม conv3ds โดยส่งชื่อไฟล์สกุล .3ds ที่ต้องการแปลงให้เป็นไฟล์สกุล .x เข้าไป



รูปที่ 8.11 แสดงการแปลงไฟล์สกุล .3ds เป็นไฟล์สกุล .x

บทที่ 9

วินโดว์ซ็อกเก็ต (Window Socket)

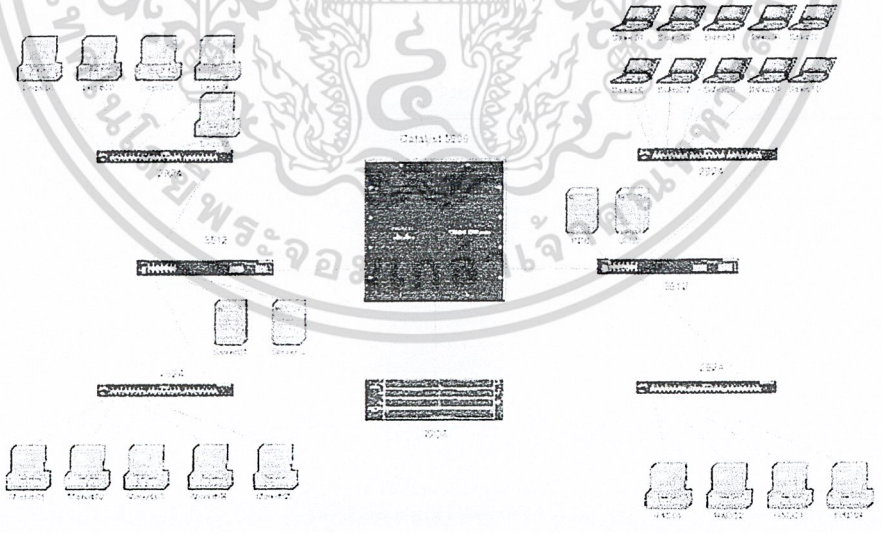
9.1 ความรู้เบื้องต้นเกี่ยวกับเครือข่ายและโปรโตคอล

9.1.1 เครือข่าย (Neworks)

โดยหลักแล้วจะหมายถึงกลุ่มของคอมพิวเตอร์หรืออุปกรณ์ซึ่งเชื่อมโยง และสื่อสารกันได้ผ่านการเชื่อมต่อในรูปแบบใดรูปแบบหนึ่ง โดยที่อุปกรณ์ใด ๆ (เช่น เครื่องคอมพิวเตอร์ พรินเตอร์ สวิทซ์) ซึ่งเชื่อมต่ออยู่กับโครงข่าย(Network) จะถูกเรียกว่า โหนด (Node) โดยที่โหนดแต่ละโหนดจะเชื่อมต่อกันโดยลิงค์ (Links) ซึ่งลิงค์นั้นอาจเป็นได้ทั้งสายสัญญาณ สัญญาณวิทยุ และแต่ละโหนดในโครงข่ายนั้นสามารถสื่อสารโดยการส่งข้อมูล/ข้อความ (Message) ถึงกันได้ผ่านทางลิงค์เหล่านี้

สำหรับตัวโครงข่ายเองแล้วสามารถจำแนกออกเป็นประเภทต่าง ๆ โดยใช้ขนาดของโครงข่ายเป็นเกณฑ์ในการจำแนกได้ดังนี้

9.1.1.1 แลน (LAN : Local Area Network) เป็นโครงข่ายขนาดเล็ก เป็นโครงข่ายภายในพื้นที่ซึ่งจำกัด เช่น โครงข่ายซึ่งเกิดจากการเชื่อมต่อคอมพิวเตอร์ 2 เครื่องภายในบ้าน ไปจนกระทั่งถึงการเชื่อมต่อคอมพิวเตอร์เป็นร้อยเครื่องภายในอาคารสำนักงานขนาดใหญ่เป็นต้น โดยในปัจจุบันอีเทอร์เนต(Ethernet) เป็นเทคโนโลยีที่นิยมใช้ในโครงข่ายประเภทนี้



รูปที่ 9.1 เครือข่ายแลน

9.1.1.2 แวน (WAN : Wide Area Network) กล่าวอย่างง่ายๆ ก็คือโครงข่ายซึ่งเกิดจากการเชื่อมต่อแลนหลายๆ วงเข้าด้วยกัน โดยมีโครงข่ายความเร็วสูงเป็นแกนในการเชื่อมต่อโครงข่ายความเร็วต่ำกว่าหลายๆ วงเข้าด้วยกัน โครงข่ายความเร็วสูงซึ่งทำหน้าที่เป็นแกนหลักในการเชื่อมต่อดังกล่าวมีชื่อเรียกว่า แบ็คโบน (Backbone)

9.1.1.3 แมน (MAN : Metropolitan Area Network) เป็นการเชื่อมต่อแลนหลายๆ วงเข้าด้วยกัน เช่นเดียวกับแวน หากแต่ว่าขอบเขตของแมนจะอยู่ในระดับของการเชื่อมต่อแลนระหว่างเมือง หรือพื้นที่ซึ่งอยู่ห่างไกลกันด้วยสายสัญญาณความเร็วสูง เช่น ใยแก้วนำแสง (Fiber Optic) หรือดาวเทียม

9.1.2 โปรโตคอล (Protocol)

ในการที่จะให้แต่ละโหนดในโครงข่ายสามารถสื่อสาร (ส่งข้อมูล) ระหว่างกันได้นั้น นอกเหนือจากการเชื่อมต่อแต่ละโหนดเข้าด้วยกันทางกายภาพด้วยลิงค์แล้วนั้น ยังมีความจำเป็นที่จะต้องกำหนดรูปแบบและขั้นตอนของการส่งสัญญาณโดยรูปแบบและขั้นตอนดังกล่าวจะถูกเรียกว่า โปรโตคอล เช่น โปรโตคอลทีซีพี/ไอพี (TCP/IP)

9.2 โปรโตคอลทีซีพี/ไอพี (TCP/IP)

โดยทั่วไปในการติดต่อสื่อสารที่เป็นที่นิยมใช้กันในปัจจุบันจะกระทำบนพื้นฐานของโปรโตคอลทีซีพี/ไอพี และชุดของซอฟแวร์ (Software) ซึ่งทำงานกับแพ็กเก็ต (Packets) ตามมาตรฐานของ ทีซีพี/ไอพี (TCP/IP) นั้นจะถูกสร้างขึ้นมาเป็นส่วนหนึ่งของตัวระบบปฏิบัติการ (Operating System, OS) ดังนั้น เมื่อแอพพลิเคชันซอฟแวร์ (Application Software) ใด ๆ ที่ต้องการสื่อสารผ่านระบบเครือข่ายตามมาตรฐานโปรโตคอลทีซีพี/ไอพี จะต้องติดต่อกับระบบปฏิบัติการเพื่อขอรับบริการ (Services)

สำหรับทางผู้พัฒนามาตรฐานโปรโตคอลทีซีพี/ไอพี ไม่ต้องการให้โปรโตคอลทีซีพี/ไอพีใช้งานได้กับแพลตฟอร์ม (Platform) ใดเป็นการเฉพาะ หรือใช้งานได้กับเพียงระบบปฏิบัติการระบบใดระบบหนึ่ง ดังนั้นผู้พัฒนาจึงมีความระมัดระวังในการกำหนดมาตรฐานมิให้มีการอ้างถึงข้อมูลการเชื่อมต่อภายในซึ่งเป็นรูปแบบเฉพาะของแพลตฟอร์มใดแพลตฟอร์มหนึ่ง หรืออ้างถึงวิธีการเชื่อมต่อกับ แอพพลิเคชันซอฟแวร์ ในลักษณะหรือรูปแบบเฉพาะของระบบปฏิบัติการจากผู้ผลิตรายใดรายหนึ่ง และด้วยแนวคิดในการออกแบบที่ไม่ยึดติดกับแพลตฟอร์มหรือระบบปฏิบัติการใด ๆ นี้ทำให้อาจกล่าวได้ว่า โปรโตคอลทีซีพี/ไอพี มีลักษณะเป็น Loosely Specified Protocol Software Interface

ข้อดีของการออกแบบโปรโตคอลทีซีพี/ไอพีให้เป็นแบบ Loosely Specified Protocol Software Interface นั้นคือความคล้องตัว (Flexible) และ ความคงทน (Tolerance) ทั้งนี้เพราะการไม่กำหนดรายละเอียดปลีกย่อยทำให้นักออกแบบระบบสามารถที่จะใส่โปรโตคอลทีซีพี/ไอพีลงในระบบปฏิบัติการได้โดยง่ายไม่ว่าระบบปฏิบัติการนั้นจะเป็นระบบปฏิบัติการแบบง่าย ๆ อย่างที่ใช้กันใน ระบบฝังตัว (Embedded System) ไปจนถึงระบบปฏิบัติการที่สลับซับซ้อนอย่างในระบบปฏิบัติการที่ใช้ในเครื่องซูเปอร์คอมพิวเตอร์(Super Computer) เป็นต้น นอกจากนั้นด้วยการที่ไม่กำหนดวิธีการเชื่อมต่อกับแอพ

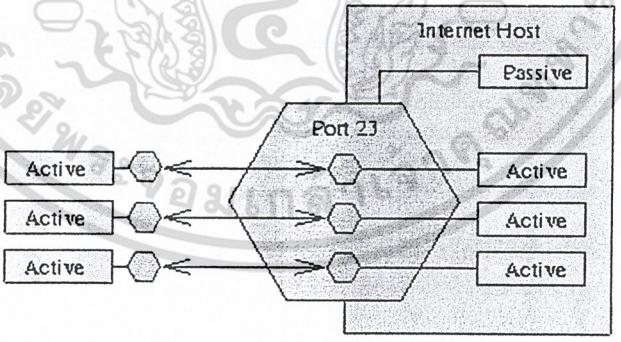
พลิเคชันซอฟแวร์ ทำให้นักออกแบบระบบสามารถเลือกใช้รูปแบบในการเชื่อมต่อ (Interface) ได้อย่างอิสระ

9.3 Sockets

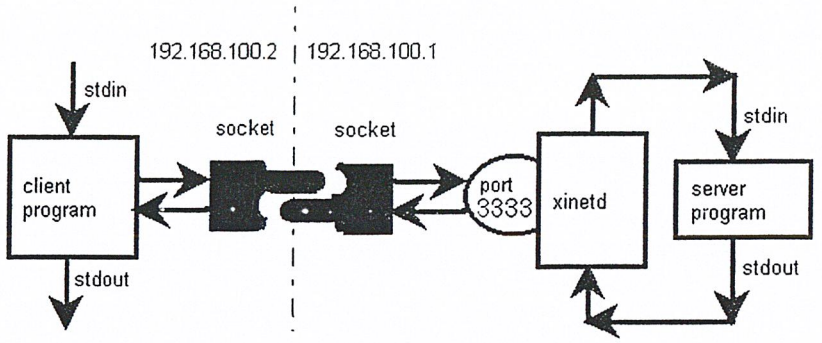
ช่วงต้นทศวรรษที่ 1980 Advanced Research Projects Agency (ARPA) ได้ให้ทุนกับ University of California at Berkeley เพื่อพัฒนาซอฟแวร์ทีซีพี/ไอพีบนระบบปฏิบัติการยูนิกซ์ (UNIX) และผลส่วนหนึ่งของโครงการดังกล่าวคือการออกแบบและพัฒนาส่วนการเชื่อมต่อ (Interface) ซึ่งแอพพลิเคชันซอฟแวร์จะใช้ในการสื่อสารข้อมูล ผ่านเครือข่าย และผลที่ได้รับคือ Berkeley Sockets Interface หรือที่เรียกกันสั้น ๆ ว่า การเชื่อมต่อแบบซ็อกเก็ต (Sockets Interface) โดยจุดเด่นของการเชื่อมต่อแบบซ็อกเก็ตนั้นคือ การสนับสนุนการสื่อสารข้อมูลผ่านเครือข่ายด้วยโปรโตคอลที่หลากหลายที่สุดเท่าที่จะเป็นไปได้ ทำให้การเชื่อมต่อแบบซ็อกเก็ตมีผู้ใช้งานอย่างกว้างขวาง จนกระทั่งอาจกล่าวได้ว่า การเชื่อมต่อแบบซ็อกเก็ตนั้นได้กลายเป็นมาตรฐานสำหรับการเชื่อมต่อ (Interface) ไปโดยปริยาย

9.3.1 การใช้ซ็อกเก็ต

ซ็อกเก็ตที่ถูกสร้างขึ้นนั้นจะมีวัตถุประสงค์ในการใช้งานอยู่เพียง 2 ลักษณะได้แก่ การรอรับการเชื่อมต่อและการเชื่อมต่อไปยังเครื่องปลายทาง (Initiate a connection) โดยซ็อกเก็ตที่รอรับการเชื่อมต่อนั้นจะเป็นซ็อกเก็ตที่ใช้ในแอพพลิเคชันฝั่งเซิฟเวอร์ (Server Application) ซึ่งซ็อกเก็ตชนิดนี้มีชื่อเรียกว่าแพสซิฟซ็อกเก็ต (Passive Socket) ในขณะที่ซ็อกเก็ตที่พยายามเชื่อมต่อไปยังเครื่องปลายทางซึ่งใช้ในแอพพลิเคชันฝั่งไคลเอนท์ (Client Application) นั้นจะมีชื่อเรียกว่าแอกทีฟซ็อกเก็ต (Active Socket)



รูปที่ 9.2 แพสซิฟซ็อกเก็ตและแอกทีฟซ็อกเก็ต



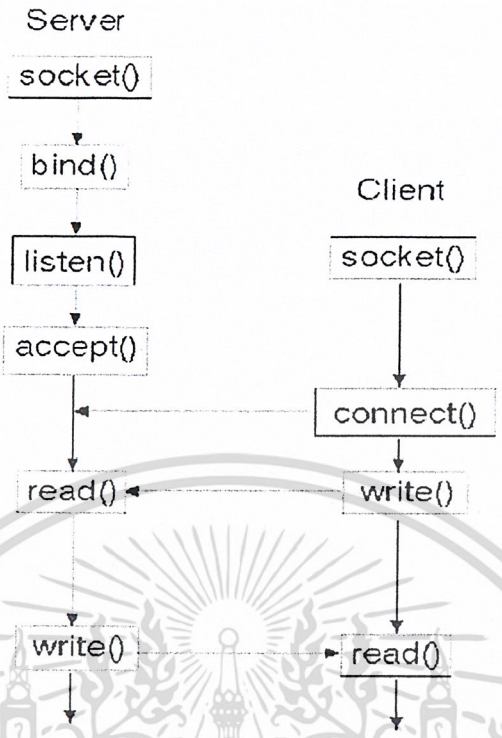
รูปที่ 9.3 แสดงการเชื่อมต่อระหว่างเซิฟเวอร์ซ็อกเก็ตกับไคลเอนท์ซ็อกเก็ต

9.3.2 ชนิดของซ็อกเก็ต (Socket Types)

ซ็อกเก็ตสามารถแบ่งออกเป็นประเภทตามลักษณะการรับส่งข้อมูลผ่านเครือข่ายได้เป็น

9.3.2.1 สตรีมซ็อกเก็ต (Stream Sockets) หรือ Connection Oriented Sockets เป็นซ็อกเก็ตที่ต้องสร้างเส้นทางการเชื่อมต่อก่อนการส่งข้อมูล และมีการติดตามสถานะของการเชื่อมต่อ ทำให้รับประกันได้ว่าข้อมูลที่ส่งได้ไปถึงปลายทาง

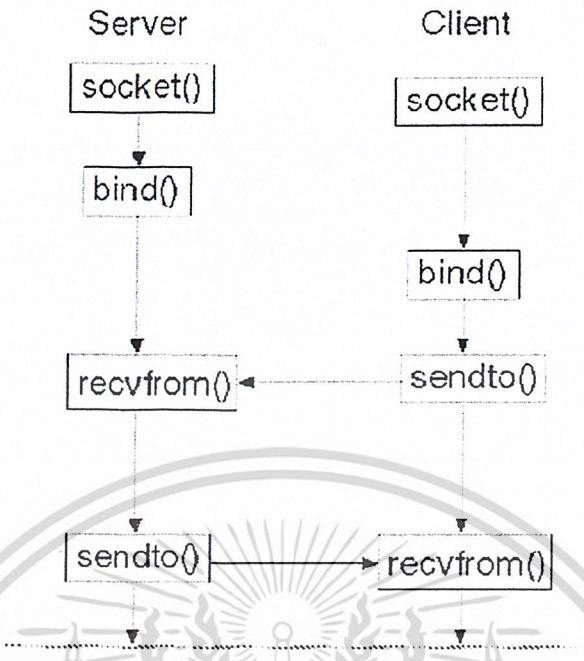
รูปที่ 9.4 แสดงการเปรียบเทียบระหว่างการสื่อสารของมนุษย์กับการส่งข้อมูลของซ็อกเก็ต



รูปที่ 9.5 แสดงขั้นตอนการสื่อสารผ่านสตรีมซ็อกเก็ต

9.3.2.2 ดาต้าแกรมซ็อกเก็ต (Datagram Sockets) หรือ Connectionless Sockets เป็นซ็อกเก็ตซึ่งใช้งานกับ โปรโตคอลยูดีพี (UDP Protocol) โดยที่การส่งข้อมูลโดยผ่านดาต้าแกรมซ็อกเก็ตนี้ไม่จำเป็นต้องสร้างเส้นทางการเชื่อมต่อเสมือน กล่าวคือ เมื่อต้องการจะส่งข้อมูลโปรแกรมก็จะทำการส่งข้อมูลนั้นออกไปเลย โดยไม่มีการตรวจสอบสถานะ/เส้นทางไปยังฝั่งที่จะรับ ดังนั้นการสูญหายของข้อมูลจึงอาจเกิดขึ้นได้

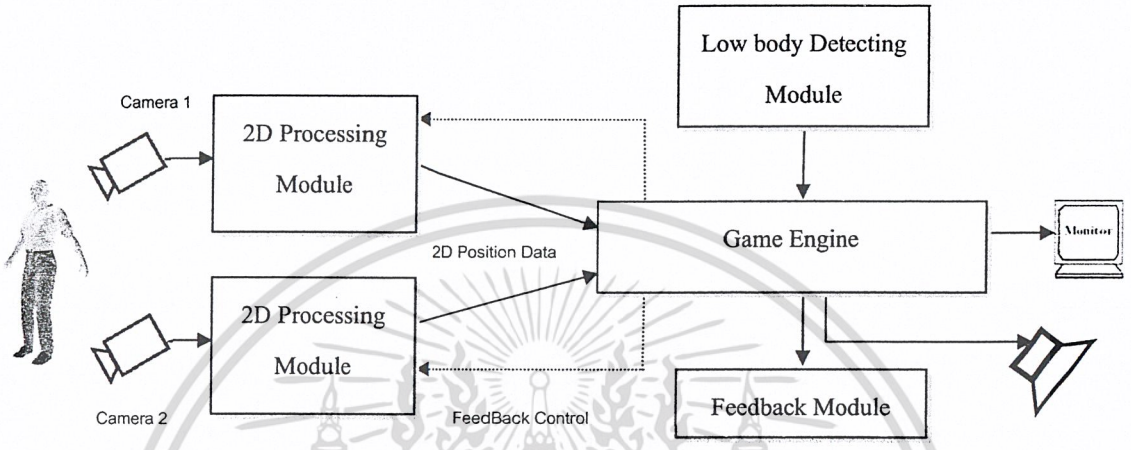
จากคุณลักษณะของยูดีพี (UDP) ที่ไม่รับประกันในเรื่องของข้อมูลสูญหายระหว่างทาง ไม่รับประกันการเรียงลำดับของข้อมูล ดังนั้นในโปรแกรมที่ใช้งานกับดาต้าแกรมซ็อกเก็ตจะต้องมีส่วนของโปรแกรมที่ใช้จัดการเกี่ยวกับปัญหาดังกล่าว จึงอาจกล่าวได้ว่า ดาต้าแกรมซ็อกเก็ตนั้นหากมองในด้านการรับและส่งข้อมูลสามารถทำได้ง่ายและมีกระบวนการที่ซับซ้อนน้อยกว่า สตรีมซ็อกเก็ตเป็นอย่างมาก แต่ถ้ามองด้านการจัดการข้อมูลซึ่งได้รับเข้ามาแล้วจะพบว่าต้องมีกระบวนการรองรับที่ซับซ้อนมากกว่าการใช้งานสตรีมซ็อกเก็ต เนื่องจากการจัดการเกี่ยวกับข้อมูลที่รับเข้ามาหลาย ๆ อย่างไม่มีการรองรับโดยตัวโปรโตคอลยูดีพี ดังที่ได้กล่าวมาข้างต้น



รูปที่ 9.6 แสดงขั้นตอนการสื่อสารผ่านดาต้าแกรมซ็อกเก็ต

บทที่ 10
การออกแบบระบบ

10.1 ภาพรวมของโครงงาน



รูปที่ 10.1 ภาพรวมของระบบ

การทำงานจะเริ่มจากส่วนประมวลผลการเคลื่อนไหว 2 มิติดึงภาพจากกล้องวีดีโอ เพื่อทำการประมวลผลหาตำแหน่งมาร์กเกอร์บนภาพ แล้วส่งตำแหน่งของมาร์เกอร์ของแต่ละกล้องไปให้ส่วนของเกม เพื่อทำการคำนวณหาตำแหน่งของมาร์กเกอร์บน world coordinate แล้วนำตำแหน่งที่ได้ไปควบคุมตัวละครให้เคลื่อนไหวตามผู้เล่น แล้วเกมจะวิเคราะว่าการเคลื่อนไหวนั้นมีผลอย่างไรกับเกม และในส่วนของเกมนี้ยังรับข้อมูลการเคลื่อนไหวของร่างกายส่วนล่างของผู้เล่นเพื่อนำไปย้ายตำแหน่งตัวละคร อีกทั้งเมื่อผู้เล่นโดนโจมตี เกมจะส่งข้อมูลไปให้ส่วนตอบสนองเพื่อให้มีการตอบสนองกลับไปยังผู้เล่นด้วย

10.2 การตรวจจับการเคลื่อนไหว 2 มิติ

การตวจจับการเคลื่อนไหวนั้นต้องมีการจัดระบบและสภาพแวดล้อมที่เหมาะสมและต้องเขียนโปรแกรมมาเพื่อรองรับการจับการเคลื่อนไหวนั้นด้วย

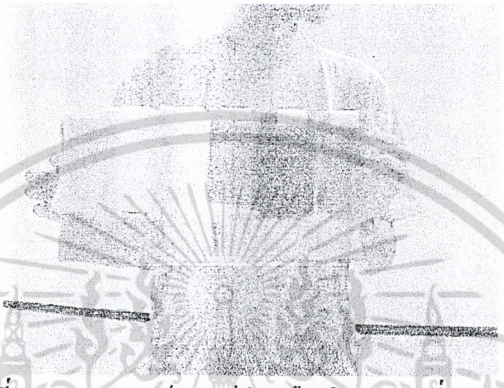
10.2.1 การจัดระบบและสภาพแวดล้อมในการถ่าย

การจับการเคลื่อนไหว 2 มิตินั้นมีข้อจำกัด ที่ต้องมีการจัดสภาพแวดล้อมให้เหมาะสมเพื่อให้กล้องวีดีโอสามารถถ่ายภาพได้โดยให้เห็นมาร์กเกอร์ชัดที่สุด และมีการบดบังมาร์กเกอร์น้อยที่สุด

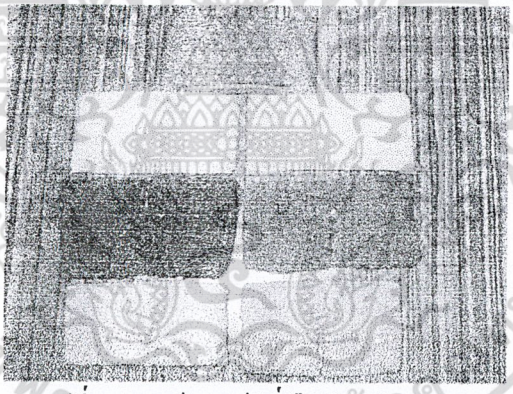
10.2.1.1 มาร์กเกอร์

มาร์กเกอร์ที่ใช้ต้องหามาร์กเกอร์ที่มีสีแตกต่างกัน 7 สี ซึ่งแทนตำแหน่งของจุดที่สนใจคือ มือขวา มือซ้าย ศอกขวา ศอกซ้าย ไหล่ซ้าย ไหล่ขวา และศีรษะ

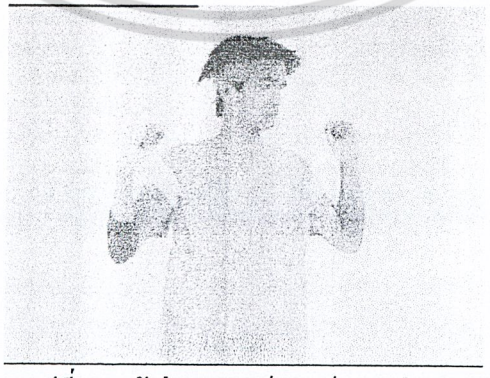
ในโครงงานนี้ใช้มาร์กเกอร์สียางยืดซึ่งการเลือกมาร์กเกอร์นั้นต้องนำมาร์กเกอร์สีหลายสีมาวิเคราะห์ค่าในระบบสี HSI ให้มีความแตกต่างกันของค่าสีมากที่สุด เพื่อให้มีการเหลื่อมล้ำกันของค่าสีบนมาร์กเกอร์น้อยที่สุด



รูปที่ 10.2 การนำมาร์กเกอร์สีมาเลือกในสภาวะที่กำหนดไว้



รูปที่ 10.3 มาร์กเกอร์สีที่เลือกใช้ในโครงงาน

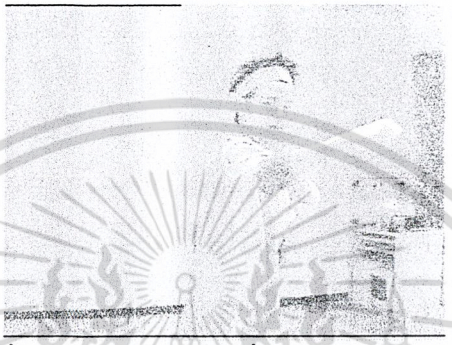


รูปที่ 10.4 ผู้เล่นสวมมาร์กเกอร์ขณะเล่นจริง

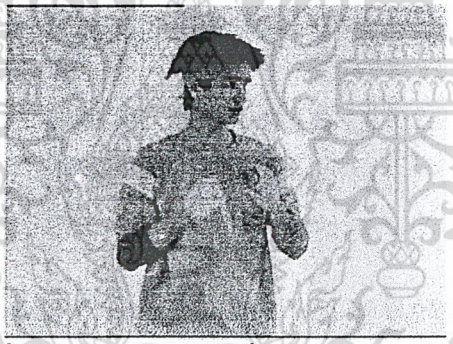
10.2.1.2 การจัดระบบแสง

การจัดระบบแสงต้องทำให้แสงที่ตกระทบที่มาร์กเกอร์แล้วเห็นสีของมาร์กเกอร์ชัดที่สุด โดยถ้ามีการจัดให้บริเวณจุดยืนมีแสงสว่างมาตกกระทบมากเกินไปจะทำให้บางส่วนของมาร์กเกอร์เป็นสีขาวแล้วจะทำให้ส่วนที่เป็นสีขาวนั้นไม่สามารถวิเคราะห์ได้

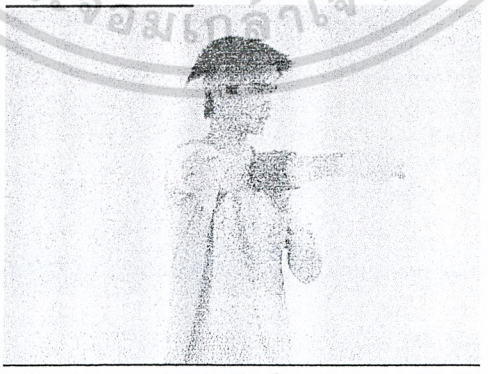
ในทางกลับกันถ้าให้บริเวณจุดยืนมีแสงสว่างน้อยเกินไปก็จะทำให้มาร์กเกอร์แต่ละสีมืดเกินไป ซึ่งทำให้ความสามารถในการแยกแยะสีของมาร์กเกอร์แย่ลง เพราะช่วงสีที่จะนำมาแยกแยะอาจเกิดการเหลื่อมล้ำของช่วงสีกันได้



รูปที่ 10.5 การจัดบริเวณยืนที่มีความสว่างมากเกินไป



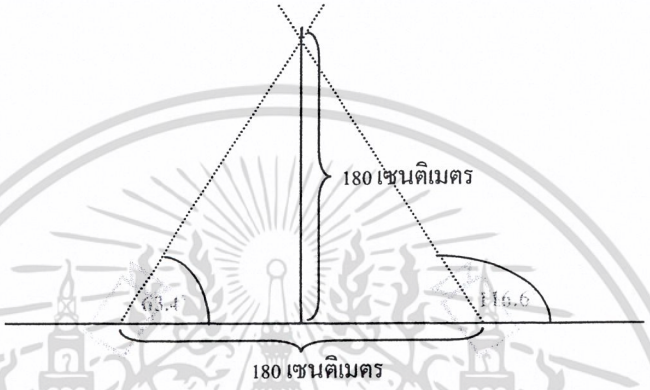
รูปที่ 10.6 การจัดบริเวณยืนที่มีความสว่างน้อยเกินไป



รูปที่ 10.7 การจัดบริเวณยืนที่มีความสว่างเหมาะสม

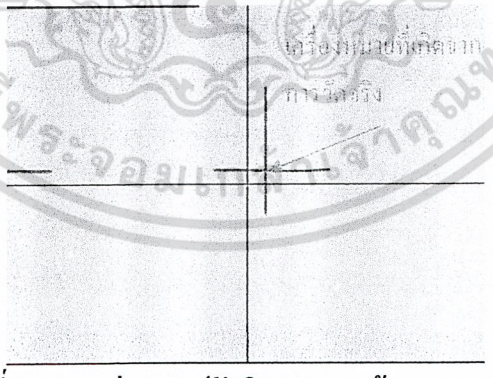
10.2.1.3 การตั้งกล้อง

ตั้งกล้องให้มีการบดบังของมาร์กเกอร์น้อยที่สุด ในโครงงานนี้ไม่จำเป็นต้องตั้งกล้องให้ห่างกันมากนักเพราะว่า โครงงานนี้ไม่ต้องการพื้นที่ในการเคลื่อนไหวมาก และการตั้งกล้องให้ห่างกันมากนั้นทำให้มุมการมองเห็นของกล้องแคบลง ซึ่งทำให้โอกาสเกิดการบดบังของมาร์กเกอร์มีมากตามไปด้วย และในโครงงานนี้ไม่จำเป็นต้องตั้งกล้องไกลจากผู้เล่นมาก เพราะต้องการเก็บข้อมูลแค่การเคลื่อนไหวส่วนบนเท่านั้น

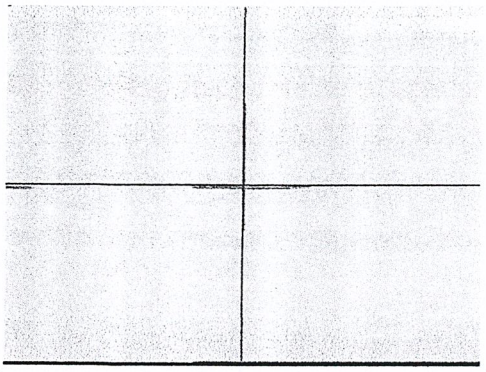


รูปที่ 10.8 การตั้งกล้องเพื่อเก็บภาพการเคลื่อนไหว

การตั้งกล้องให้อยู่ในรูปแบบและทิศทางที่ถูกต้องนั้นทำได้โดยทำเครื่องหมายที่ถูกต้องจากการวัดจริง ซึ่งส่วนที่ต้องทำเครื่องหมาย คือ ส่วนจุดตั้งกล้อง ทิศทางของกล้อง และความสูงของกล้อง และเมื่อทำเครื่องหมายแล้วจึงเขียนโปรแกรมขึ้นมาช่วยในการปรับกล้องอีกครั้งหนึ่ง



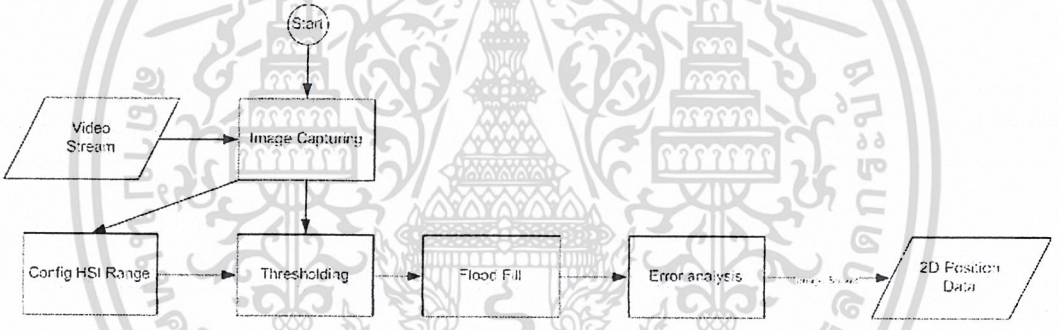
รูปที่ 10.9 ภาพก่อนการปรับทิศทางของกล้องและความสูง



รูปที่ 10.10 ภาพหลังการปรับทิศทางของกล้องและความสูง

การปรับกล้อง เริ่มจากตั้งกล้องให้ตรงกับจุดวางกล้องที่กำหนดไว้ที่พื้น และหลังจากนั้นให้ปรับทิศทางของกล้องให้ได้มุมที่กำหนดไว้ และปรับความสูงของกล้อง ซึ่งการปรับสองส่วนหลัง คือ การปรับกล้องให้เครื่องหมายบนฉากตรงกับเส้นแนวนอนและแนวตั้งบนภาพ

10.2.2 การออกแบบโปรแกรมประมวลผลการเคลื่อนไหว 2 มิติ



รูปที่ 10.11 ผังการทำงานของโปรแกรมประมวลผลการเคลื่อนไหว 2 มิติ

10.2.2.1 รับภาพจากกล้องวีดีโอ

การทำงานของโปรแกรมเริ่มจากการรับภาพจากกล้องวีดีโอโดยโปรแกรมนี้ใช้ วีดีโอโอซีเอ็กซ์ เป็นตัวรับภาพมาจากกล้อง เพื่อนำไปประมวลผลภาพหาตำแหน่งของมาร์กเกอร์ต่อไป โดยโปรแกรมนี้มีการตั้งค่าให้ดึงภาพที่ 320*240 พิกเซล เพราะมีความละเอียดที่ไม่มาก หรือน้อยเกินไป ถ้าเลือกความละเอียดของภาพมากเกินไปการประมวลผลจะช้ามาก ในทางกลับกันถ้าน้อยเกินไปจะทำให้ค่าที่คำนวณได้คลาดเคลื่อนจากความเป็นจริงมากเกินไป

10.2.2.2 การตั้งช่วงค่าสีในระบบสี HSI

เมื่อได้รับภาพจากกล้องวีดีโอมาแล้ว ให้กำหนดช่วงสีของมาร์กเกอร์จากภาพวีดีโอโดยให้ผู้ใช้เป็นคนคลิ้กที่ตำแหน่งของมาร์กเกอร์ โดยคลิ้กมาร์กเกอร์ละ 5 ครั้งเพื่อทำการหาค่าเฉลี่ยของสี โดยค่าสีเฉลี่ยที่ได้จะอยู่ในระบบสี RGB ต้องทำการแปลงค่า RGB นี้ไปเป็นค่าสีในระบบสี HSI ค่าที่ได้นี้จะเป็น

ค่ากึ่งกลางของช่วงสีที่จะนำไปตรวจหาตำแหน่งของมาร์กเกอร์ โดยช่วงสีที่นำไปตรวจหามาร์กเกอร์นั้น กำหนดโดย ช่วงของค่า H ขยายจากค่าเฉลี่ย H ไปทางบวก และลบข้างละ 4 องศา และ ช่วงค่า S,I ขยายจากค่าเฉลี่ยไปทางบวก และลบข้างละ 10% ซึ่งค่าที่นำมากำหนดนี้มาจากการทดลองว่า ถ้ากำหนดค่าประมาณนี้แล้วจะตรวจพบพิกเซลที่เป็นของมาร์กเกอร์มากที่สุด และมีการตรวจหาที่ผิดพลาดน้อยที่สุด (เช่น การตรวจพบว่าพิกเซลนั้นเป็นของข้อมือขวา แต่แท้จริงเป็นของศอกซ้าย เป็นต้น)

10.2.2.3 การนำภาพวีดีโอไปประมวลผล

เมื่อกำหนดค่าช่วงสีของแต่ละมาร์กเกอร์ได้แล้ว จึงนำภาพวีดีโอมาประมวลผลเพื่อหาแยกแยะมาร์กเกอร์โดยใช้หลักการของ Thresholding โดยตรวจสอบแต่ละพิกเซลว่ามีค่าที่อยู่ในช่วงที่กำหนดไว้หรือไม่ และถ้าค่าสีพิกเซลนั้นอยู่ในช่วงที่กำหนดแล้ว แต่พิกเซลนั้นอยู่ไกลจากจุดเดิมมากเกินไป(ในโครงงานนี้กำหนดว่าระยะห่างมากที่สุด คือ 90 พิกเซล) ก็จะไม่สนใจ หรือถือว่าเป็นสิ่งรบกวน (noise) แต่ถ้าอยู่ในระยะที่กำหนดไว้ก็ให้เก็บค่าไว้ว่าพิกเซลนี้เป็นของมาร์กเกอร์ใด ทำจนครบทุกพิกเซลของภาพแล้วก็จะได้ข้อมูลที่บอกว่าพิกเซลใดมีค่าอยู่ในช่วงมาร์กเกอร์ใด ข้อมูลเหล่านี้ จะนำไปใช้หาตำแหน่งของมาร์กเกอร์ต่อไป

10.2.2.4 การทำ Flood Fill

การทำ Flood Fill นั้นเป็นการทำเพื่อหาจุดกึ่งกลางของกลุ่มสี ซึ่งวิธีการของ Flood Fill เป็นการขยายตัวไปเรื่อยๆ เพื่อหาค่าขอบเขตของกลุ่มสี แล้วจึงนำขอบเขตนั้นมาหาจุดกึ่งกลางของกลุ่มสี (หลักการทำ Flood Fill มีอธิบายในบทที่3.2.2) โดยการขยายตัวนี้จะมีการเก็บขนาดของกลุ่มสีไว้ด้วย

10.2.2.5 การวิเคราะห์ความผิดพลาดเพื่อหาตำแหน่งที่ถูกต้อง

เมื่อได้ทำการหาจุดกึ่งกลางของกลุ่มสีแล้ว ก็จะนำค่าเหล่านั้นมาหาตำแหน่ง โดยมีแนวคิดที่ว่ากลุ่มสีของมาร์กเกอร์ใดที่มีขนาดใหญ่และอยู่ใกล้ตำแหน่งเดิมของมาร์กเกอร์นั้นมากที่สุด จะมีแนวโน้มว่าเป็นตำแหน่งของมาร์กเกอร์นั้นในเฟรมปัจจุบัน

จึงสามารถกำหนดได้ว่ากลุ่มสีใดที่ใกล้ตำแหน่งเดิมที่สุด และมีขนาดไม่เล็กเกินไป(ในโครงงานนี้ขนาดของกลุ่มสีที่ยอมรับได้ต้องมีขนาดใหญ่กว่า 15 พิกเซล) เป็นตำแหน่งของมาร์กเกอร์นั้นในเฟรมปัจจุบัน และการตรวจสอบแบบนี้จะถือว่ากลุ่มสีที่มีขนาดเล็กเป็นสิ่งรบกวน

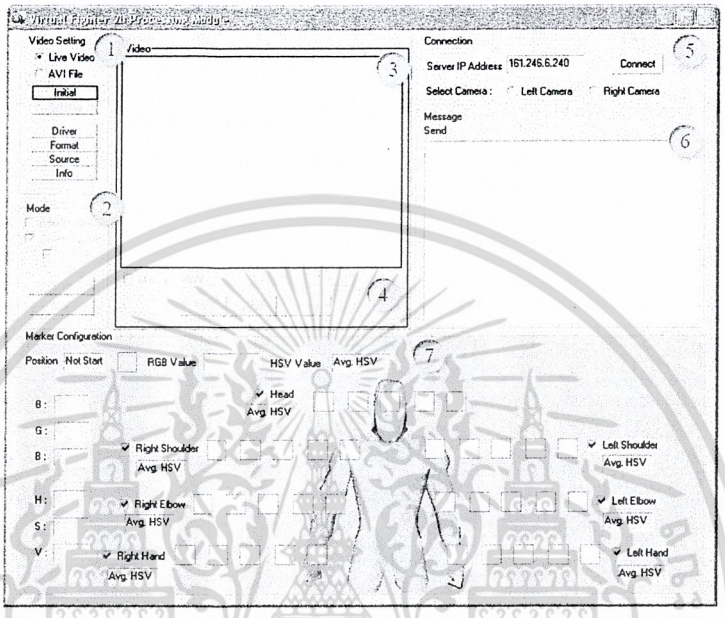
สุดท้ายถ้าเฟรมนั้น ไม่สามารถหาตำแหน่งของบางมาร์กเกอร์ได้ ก็จะสรุปว่าตำแหน่งของมาร์กเกอร์ที่หายไปนั้น มีตำแหน่งเดียวกับเฟรมที่แล้ว

10.2.2.6 ส่งข้อมูลการเคลื่อนไหว 2 มิติ ไปที่เกม

เมื่อโปรแกรมทำการประมวลผลภาพ และทำการวิเคราะห์หาตำแหน่งได้แล้วก็จะส่งข้อมูลการเคลื่อนไหว 2 มิติออกไปให้เกม เพื่อใช้ในการคำนวณหาตำแหน่งบน 3 มิติโดยผ่านทาง โปรโตคอล UDP

ซึ่งการส่งข้อมูลไปนี้จะมีการส่งเวลาไปด้วยเพื่อทำให้เกม สามารถเลือกเฟรมที่จะนำมาคำนวณตำแหน่ง 3 มิตินั้นได้อย่างถูกต้อง คือ เลือกเฟรมที่ถ่าย ณ เวลาเดียวกันมาคำนวณตำแหน่ง 3 มิติ

10.2.3 ยูสเซอร์อินเตอร์เฟส

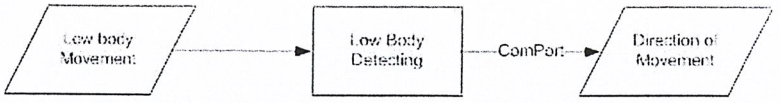


รูปที่ 10.11 แสดงยูสเซอร์อินเตอร์เฟสของโปรแกรมจับภาพการเคลื่อนไหว

- จุดที่ 1 ส่วนสร้างการติดต่อกับกล้องหรือไฟล์วีดีโอ เพื่อดึงภาพมาใช้งาน
- จุดที่ 2 ส่วนสั่งให้เริ่มการทำงาน ไม่ว่าจะเป็นการปรับกล้อง หรือการหาตำแหน่งมาร์กเกอร์
- จุดที่ 3 ส่วนแสดงผลของภาพที่จับได้ หรือภาพที่ประมวลผลแล้ว ซึ่งถ้าประมวลผลแล้วจะมีตำแหน่งบอกด้วย
- จุดที่ 4 ส่วนการเลือกที่จะควบคุมการดึงภาพและการประมวลผลเอง
- จุดที่ 5 เป็นส่วนสร้างการเชื่อมต่อกับตัวเกม
- จุดที่ 6 เป็นการแสดงข้อมูลตำแหน่งมาร์กเกอร์ 2 มิติที่ส่งออกไปยังเกม
- จุดที่ 7 ส่วนการเลือกค่าช่วงสีของมาร์กเกอร์

10.3 การตรวจจับการเคลื่อนไหวส่วนล่าง

เนื่องจากระบบไม่สามารถตรวจจับการเคลื่อนที่ของตัวละครได้ด้วยการเคลื่อนที่จริง จึงมีฮาร์ดแวร์ที่รับการเคลื่อนไหวจากเท้าแทน ซึ่งฮาร์ดแวร์ส่วนนี้ใช้เซนเซอร์สัมผัสเป็นตัวกำหนดการเคลื่อนที่ส่วนล่าง และทำฐานเหยียบขึ้นมารองรับการทำงานของเซนเซอร์สัมผัสนั้น



รูปที่ 10.12 ผังการทำงานส่วนตรวจจับการเคลื่อนไหว

10.3.1 เซนเซอร์สัมผัส

เซนเซอร์สัมผัสที่ใช้เป็นสวิทช์ที่มีลักษณะกดติดปล่อยดับ ซึ่งสามารถนำไปตรวจสอบได้ว่าถ้ามีการเหยียบจะสามารถบอกได้ว่ามีการเหยียบอยู่

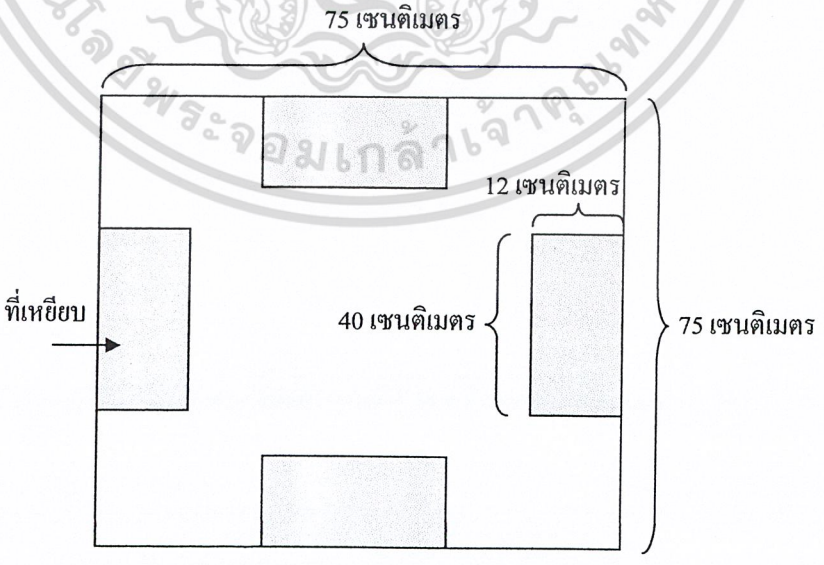
รูปที่ 10.13 เซนเซอร์สัมผัส

10.3.2 วงจร

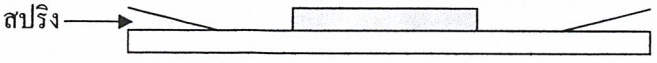
วงจรที่ใช้เป็นวงจรซึ่งใช้กับ สวิทช์ทั่วไป เมื่อกดได้ค่า '0' ถ้าไม่กดได้ค่า '1'

รูปที่ 10.14 วงจรสวิทช์

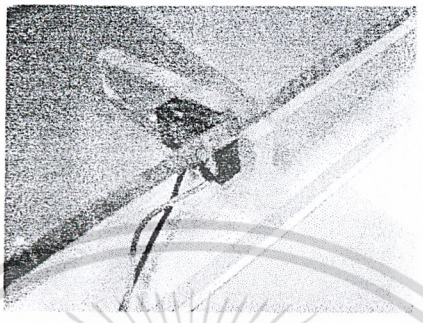
10.3.3 ฐานเหยียบ



รูปที่ 10.15 ฐานเหยียบ(มุมมองด้านบน)



รูปที่ 10.16 ฐานเหยียบ(มุมมองด้านข้าง)



รูปที่ 10.17 เซนเซอร์สัมผัสที่ติดกับฐานเหยียบ

ถ้าผู้เล่นเหยียบที่เหยียบแผ่นหน้าจะเป็นการบอกให้ตัวละครเคลื่อนที่ไปด้านหน้า ถ้าเหยียบที่เหยียบแผ่นหลังเป็นบอกให้ตัวละครเคลื่อนที่ไปด้านหลัง ถ้าเหยียบที่แผ่นซ้าย หรือ ขวา จะเป็นการบอกให้ตัวละครเคลื่อนที่ไปด้านซ้าย หรือขวาตามลำดับ

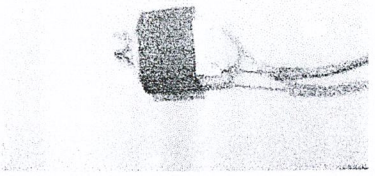
10.4 การตอบสนองกลับไปยังผู้เล่น

เพื่อการเล่นเกมที่สมจริงขึ้น ระบบมีการออกแบบให้มีการตอบสนองกลับไปยังผู้เล่นเมื่อผู้เล่นโจมตีโดนคอมพิวเตอร์หรือ ถูกคอมพิวเตอร์โจมตี

รูปที่ 10.18 ผังการทำงานส่วนตอบสนองกลับ

10.4.1 ตัวตอบสนอง

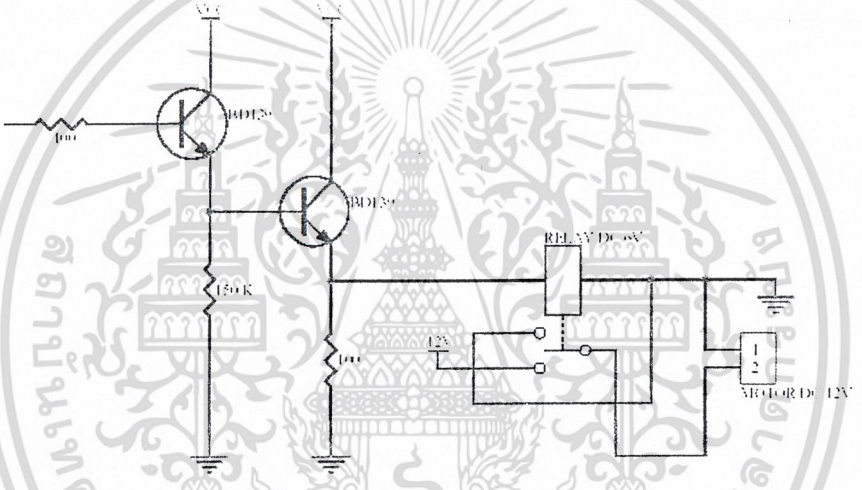
ในโครงงานนี้เลือกตัวตอบสนองเป็น มอร์เตอร์สั่นซึ่งจะออกแบบให้สั่นก็ต่อเมื่อมีการจ่ายไฟให้มอร์เตอร์



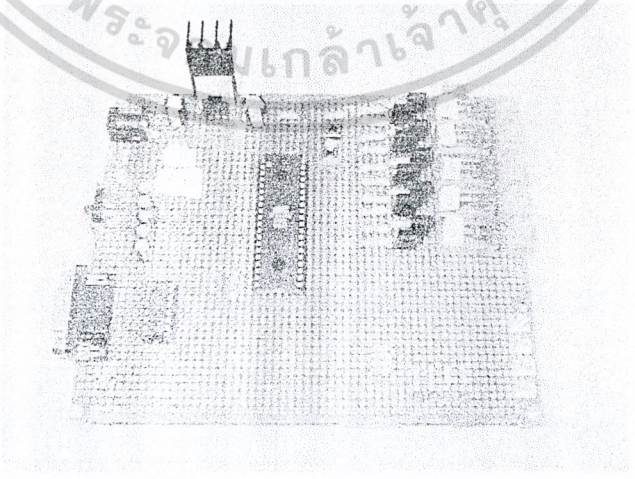
รูปที่ 10.19 ตัวตอบสนอง

10.4.2 วงจร

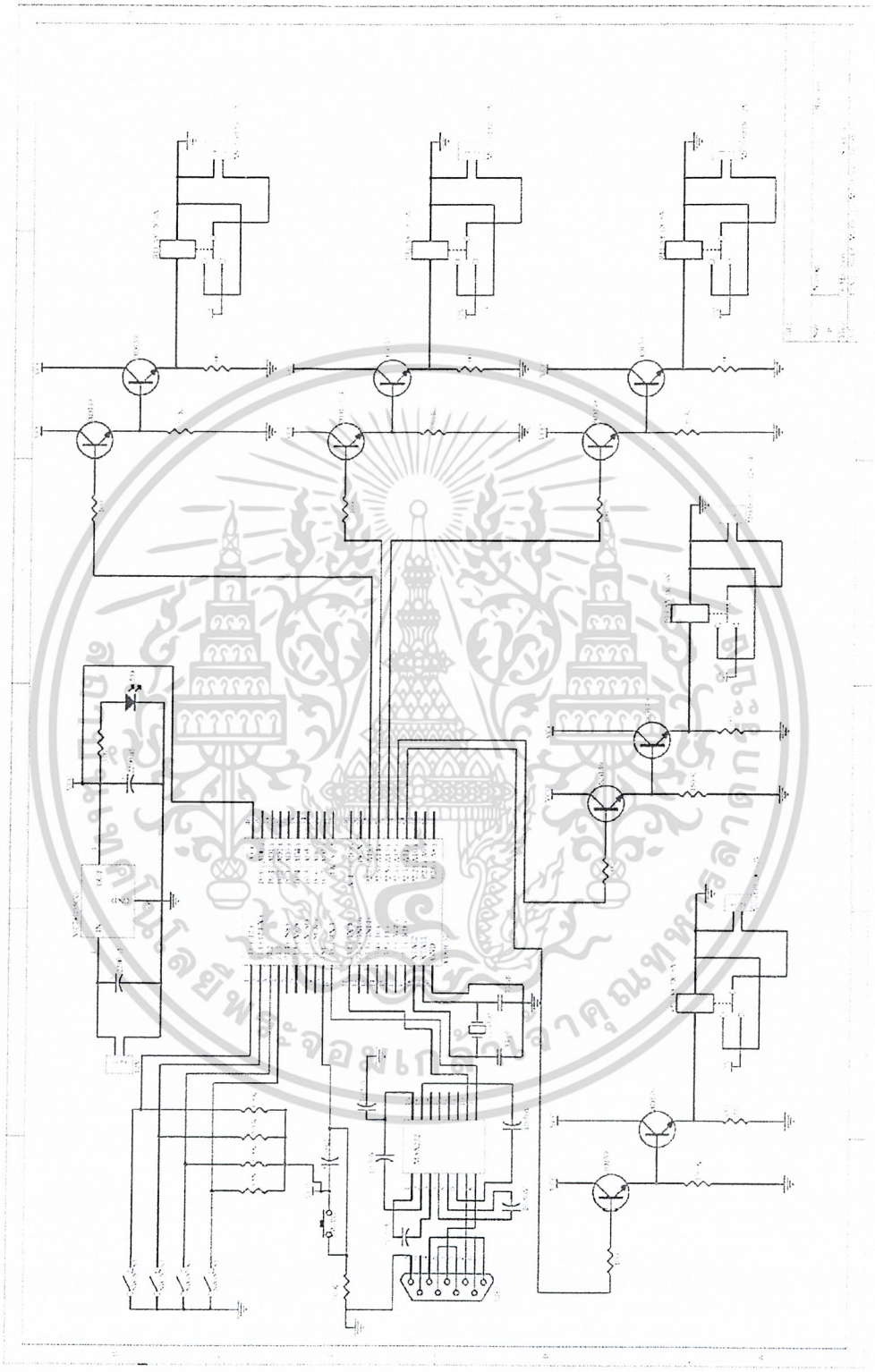
วงจรในที่นี้คือวงจรที่นำไปควบคุมมอร์เตอร์กระแสตรง เมื่อจ่าย '1' มอร์เตอร์จะหมุน



รูปที่ 10.20 วงจรควบคุมมอร์เตอร์กระแสตรง



รูปที่ 10.21 วงจรที่ทำขึ้น



รูปที่ 10.22 วงจรรวมทั้งหมด

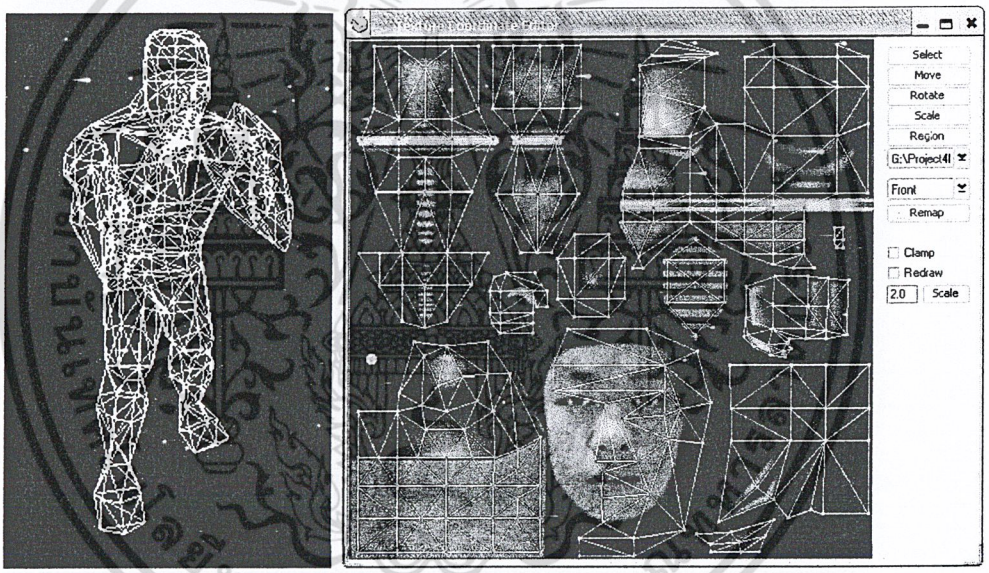
10.5 การออกแบบตัวละคร

ใช้โปรแกรมมิลค์เชพ 3 มิติ ในการสร้าง การออกแบบตัวละครแยกเป็น 2 ส่วนคือ ตัวละครฝ่ายศัตรู และ ตัวละครฝ่ายผู้เล่น ซึ่งทั้ง 2 ส่วนแยกจุดประสงค์ของการสร้างออกจากกัน

10.5.1 ตัวละครฝ่ายศัตรู

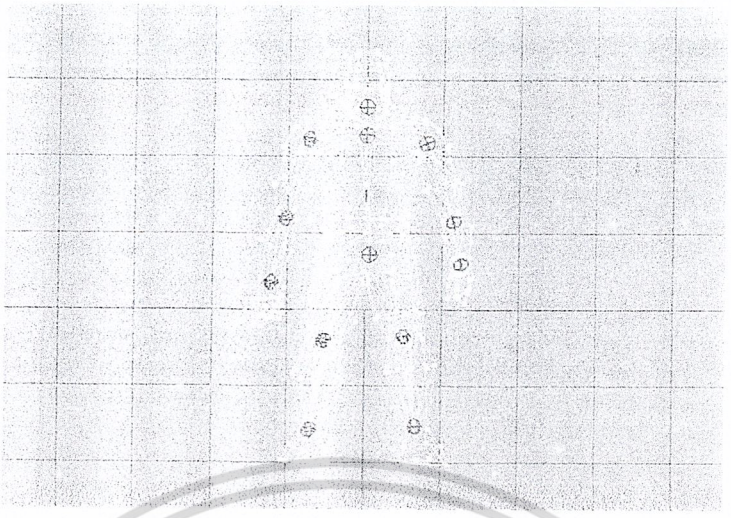
ตัวละครฝ่ายศัตรูมีจุดประสงค์คือแอนิเมชันของตัวละครแบบเฟรม คือ การแสดงท่าทางของตัวละครจะเลือกจากเฟรมซึ่งแยกออกเป็นกลุ่มๆ ตามท่าทางการเคลื่อนไหว เช่น การชก การเดิน เป็นต้น

10.5.1.1 สร้างโมเดลตัวละครและพื้นผิวของตัวละคร โดยตัวละครสร้างมาจากจุดเวอร์เท็กหลายๆจุดมาเรียงต่อกันกลายเป็นพื้นผิว และนำภาพ 2 มิติมาแปะลงบนพื้นผิวของตัวละคร



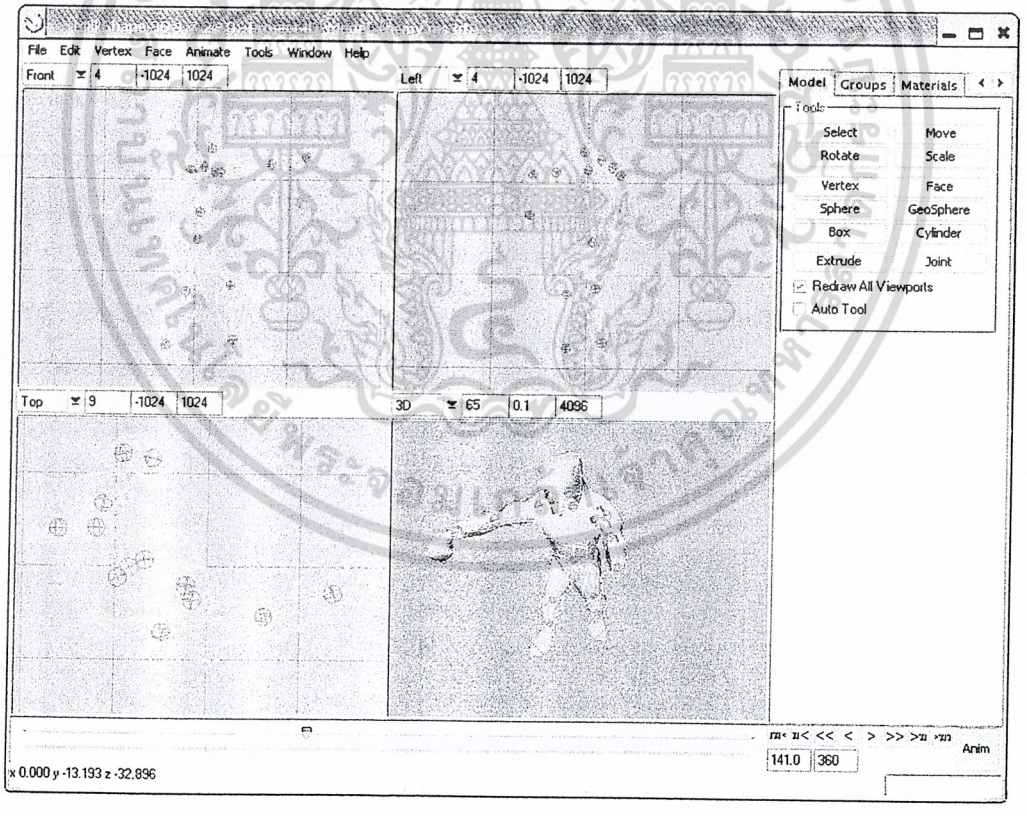
รูปที่ 10.23 โครงร่างของตัวละครศัตรูและการแปะพื้นผิวให้ตัวละคร

10.5.1.2 กำหนดข้อต่อและกระดูกให้กับตัวละคร เพื่อนำไปใช้ในการกำหนดท่าทางของตัวละครในแต่ละคีย์เฟรม เพื่อทำการสร้างแอนิเมชันการเคลื่อนไหวแบบต่างๆ



รูปที่ 10.24 กำหนดข้อต่อและกระดูกให้กับตัวละคร

10.5.1.3 กำหนดท่าทางของตัวละคร กำหนดในแต่ละคีย์เฟรมเพื่อสร้างแอนิเมชันของท่าต่างๆ



รูปที่ 10.25 กำหนดท่าทางการขยกให้กับตัวละคร

10.5.1.4 สร้างไฟล์ md2.qc เพื่อนำไปกำหนดกลุ่มท่าทางของแอนิเมชันของตัวละคร และเอ็กซ์พอร์ตตัวละครให้เป็นไฟล์สกุล .md2

md2.qc - Notepad

File Edit Format View Help

```
$modelname enemy.md2

//sequence     animation      start    end

$sequence      stand          1        24
$sequence      forward        25       48
$sequence      backward       49       72
$sequence      stepRight      73       96
$sequence      stepLeft       97       120
$sequence      punchRight     121      156
$sequence      punchLeft      157      192
$sequence      leftHitHead    193      210
$sequence      rightHitHead   211      230
$sequence      upperRight     231      260
$sequence      guard          261      290
$sequence      bodyLeft       291      304
$sequence      bodyRight      305      318
$sequence      dead           319      354
$sequence      dead           355      355
```

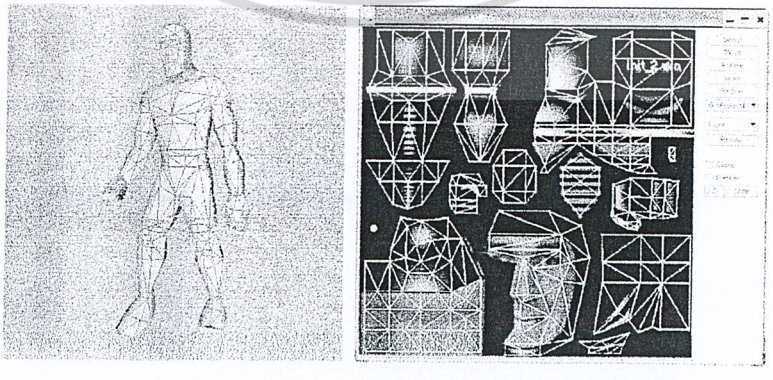
รูปที่ 10.26 ไฟล์ md2.qc

10.5.1.5 เอ็กซ์พอร์ตตัวละครออก เป็นไฟล์ enemy.md2 เพื่อนำไปใช้ใน Game Engine

10.5.2 ตัวละครฝ่ายผู้เล่น

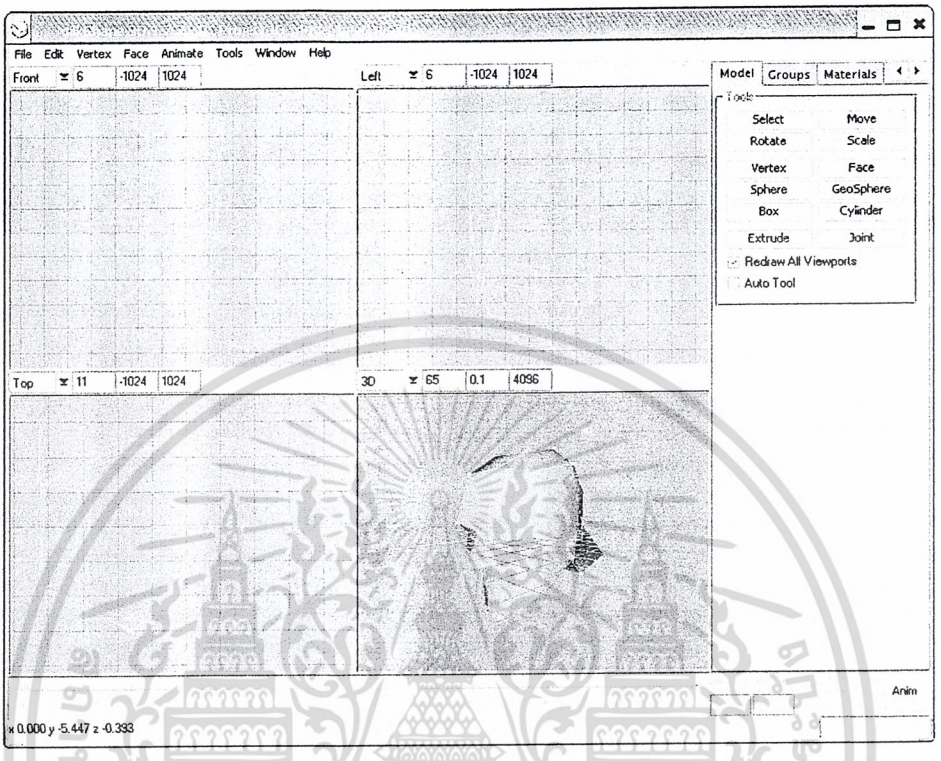
ตัวละครฝ่ายผู้เล่นมีจุดประสงค์คือการแยกชิ้นส่วนของร่างกายออกเป็นส่วนๆ คือ ส่วนหัว ส่วนลำตัว ส่วนไหล่ซ้าย-ขวา ส่วนแขนซ้าย-ขวา และส่วนล่าง

10.5.2.1 สร้างโมเดลตัวละครและพื้นผิวของตัวละคร



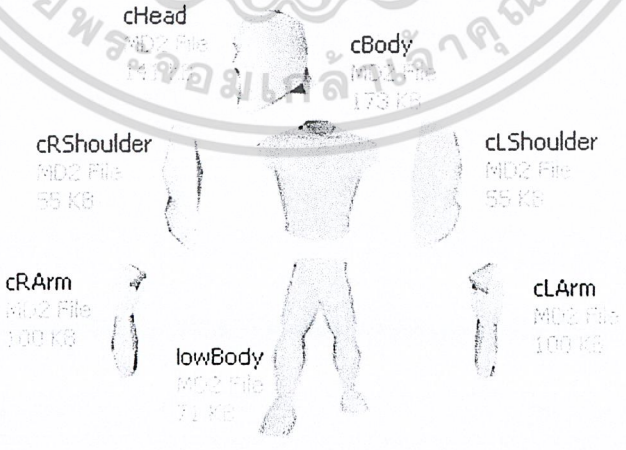
รูปที่ 10.27 โครงร่างของตัวละครผู้เล่นและการแปะพื้นผิวให้ตัวละคร

10.5.2.2 แยกชิ้นส่วนต่างๆของร่างกาย



รูปที่ 10.28 ส่วนหัวของตัวละครฝ่ายผู้เล่น

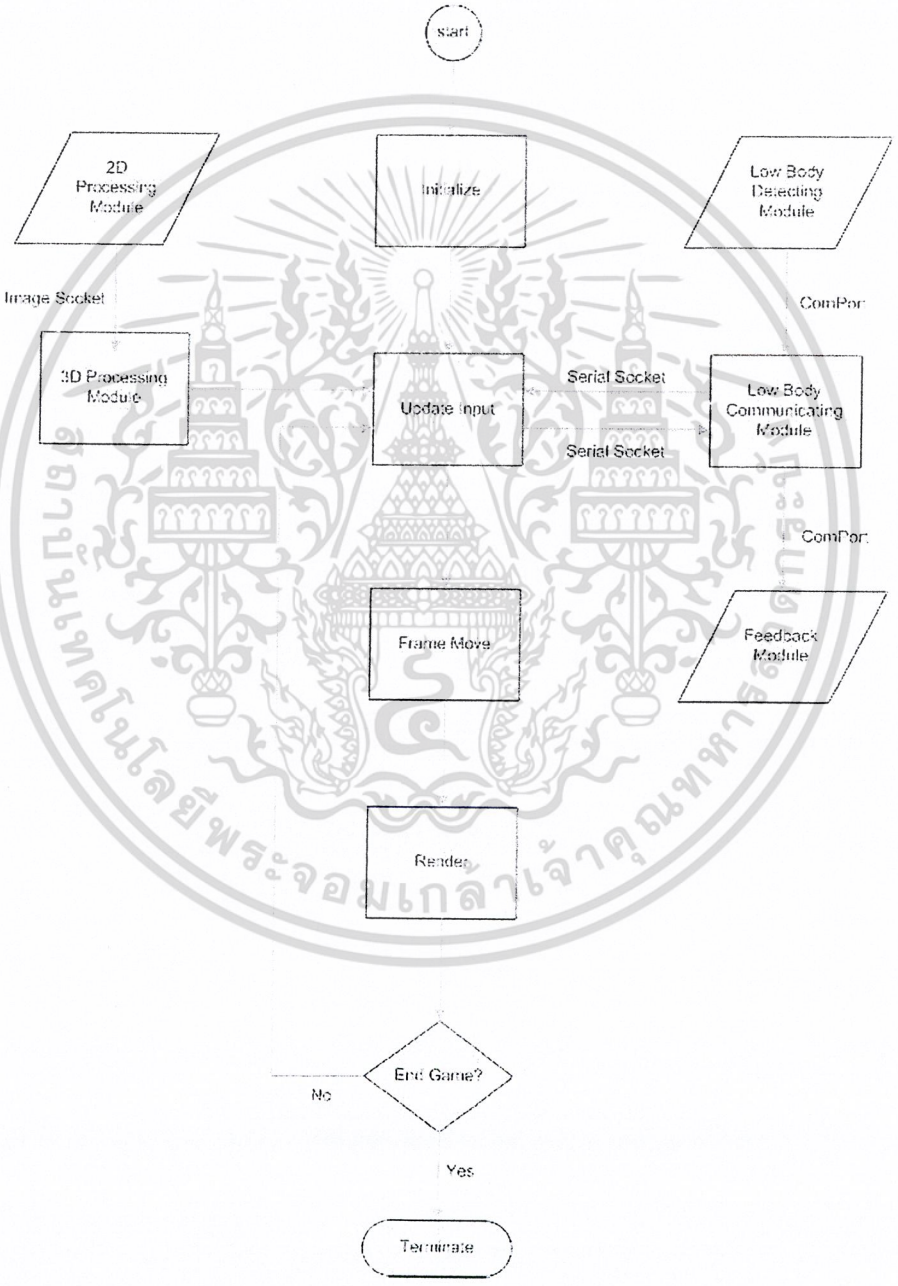
10.5.2.3 เอ็กซ์พอร์ตตัวละครออก เป็นไฟล์ชิ้นส่วนต่างๆ เพื่อนำไปใช้ใน Game Engine



รูปที่ 10.29 ไฟล์ชิ้นส่วนต่างๆที่ได้จากการเอ็กซ์พอร์ต

10.6 Game Engine

ในโครงงานนี้จะใช้โครงสร้างแอพพลิเคชันของ DirectX SDK เวอร์ชัน 9.0 ซึ่งเขียนโปรแกรมบน Microsoft Visual Studio.NET ด้วยภาษา C++ แบบ วินโดว์32แอพพลิเคชัน (Window32 Application) ในส่วนนี้จะเป็นส่วนที่ประมวลผลเกมทั้งหมด การรับอินพุทจากส่วน Motion Capture Module (ค่าของตำแหน่งร่างกายของผู้เล่น) การรับอินพุทจากส่วน Low Body Detecting Module (ค่าของการเคลื่อนที่ของผู้เล่น) รวมถึงการส่งค่าไปให้ส่วน Feedback Module เพื่อสร้างสิ่งตอบสนองกลับไปที่ร่างกายผู้เล่น



รูปที่ 10.30 โฟลวชาร์ตของเกมเอนจิ้น

10.6.1 กำหนดค่าเริ่มต้น (Initialize)

เป็นส่วนที่กำหนดค่าเริ่มต้นทั้งหมดในเกมเอนจิ้น แบ่งออกเป็นส่วนย่อยๆ คือ

10.6.1.1 สร้างอุปกรณ์ (Device) ที่ใช้ติดต่อกับส่วนการแสดงภาพ ใช้ LPDIRECT3DDEVICE9

```
LPDIRECT3DDEVICE9 m_pd3dDevice;
hr = m_pD3D->CreateDevice( m_d3dSettings.AdapterOrdinal(),
                           pDeviceInfo->DevType, m_hWndFocus,
                           behaviorFlags, &m_d3dpp,
                           &m_pd3dDevice );
```

ตารางที่ 10.1 แสดงการเตรียมอุปกรณ์ในการแสดงภาพ

10.6.1.2 สร้างอุปกรณ์วาดภาพ 2 มิติ โดยใช้อุปกรณ์ในการวาดภาพ 2 มิติที่เรียกว่า Sprite

```
ID3DXSprite* m_pSprite;
hr = D3DXCreateSprite( m_pd3dDevice, &m_pSprite);
```

ตารางที่ 10.2 แสดงการเตรียมอุปกรณ์ในการวาดภาพ 2 มิติ

10.6.1.3 เตรียมโมเดลสกุล .x โดยอิมพอร์ตมาจากไฟล์สกุล .x ซึ่งรองรับโดยคลาส CXObject ซึ่งมีการส่งพารามิเตอร์หลักๆ คือ อุปกรณ์แสดงภาพ ตำแหน่งของไฟล์สกุล .x (file) ทิศทางเริ่มต้น (turn) ขนาดย่อขยาย (scale) ตำแหน่งเริ่มต้น (x, y, z) และค่าการวาดเส้นล้อมรอบ (bounding box)

```
CXObject* m_eArrow;
m_eArrow = new CXObject(m_pd3dDevice);
hr = m_eArrow->Setup(char* file, float turn, float
scale, float x, float y, float z, bool bound=false);
```

ตารางที่ 10.3 การอิมพอร์ตไฟล์สกุล .x มาใช้ในแอพพลิเคชัน

10.6.1.3 เตรียมโมเดลตัวละคร โดยอิมพอร์ตมาจากไฟล์สกุล .md2 ซึ่งรองรับโดยคลาส CMD2Model ซึ่งมีการส่งพารามิเตอร์หลักๆ คือ อุปกรณ์แสดงภาพ ตำแหน่งของไฟล์สกุล .md2 (char*) ทิศทางเริ่มต้น (initTurn) ขนาดย่อขยาย (initScale) ตำแหน่งเริ่มต้น (initX, initY, initZ) รวมถึงการกำหนดแอนิเมชันผ่านโครงสร้าง SAnimation

```
CMD2Model* g_enemy;
g_enemy = new CMD2Model(IDirect3DDevice9* device, float initX, float
initY, float initZ, float initScale, float initTurn);
g_enemy->Load (char* );

SAnimation a;
a.add = 0.45f;
CIniFile file;
file.SetFileName ("graphic\\models\\enemy.ini");
file.GetStringValue (a.name, "STAND", "name");
file.GetIntValue    (a.start,"STAND", "start");
file.GetIntValue    (a.end,"STAND"  , "end");
a.cur = (float) a.start;
g_enemy->m_anim.push_back (a);
```

ตารางที่ 10.4 การอิมพอร์ตไฟล์สกุล .md2 มาใช้ในแอพพลิเคชัน

10.6.1.4 เตรียมพื้นผิวที่จะนำมาใช้กับโมเดล MD2 ซึ่งรองรับโดยคลาส CTextureManager โดยส่งพารามิเตอร์ที่เป็นตำแหน่งของไฟล์พื้นผิว (char*) เข้าไป

```
CTextureManager* g_enemyText;
g_enemyText->LoadTexture(char*);
g_enemyText = new CTextureManager(m_pd3dDevice);
```

ตารางที่ 10.5 การเตรียมพื้นผิวที่ใช้กับโมเดล MD2

10.6.1.5 กำหนดค่าของกล้อง มีพารามิเตอร์หลักๆ คือ

- D3DXVECTOR3 vEyePt บอกตำแหน่งของกล้อง
- D3DXVECTOR3 vLookatPt บอกทิศทางการมองของกล้อง
- D3DXVECTOR3 vUpVec บอกทิศทางการตั้งกล้อง

```
D3DXMATRIX matView;
D3DXVECTOR3 vEyePt = m_camera->GetEyePt();
D3DXVECTOR3 vLookatPt = m_camera->GetLookatPt();
```

```
D3DXVECTOR3 vUpVec = m_camera->GetUpVec();
vEyePt[0] = charAtX;
vEyePt[1] = charAtZ;
vLookatPt[0] = -1.0f;
vLookatPt[1] = 0.0f;
vLookatPt[2] = 0.0f;
vUpVec [0] = 0.0f; // ตั้งกล้องในทิศแกน Y
vUpVec [1] = 1.0f;
vUpVec [2] = 0.0f;
m_camera->SetViewParams(vEyePt, vLookatPt, vUpVec);
D3DXMatrixLookAtLH( &matView, &vEyePt, &vLookatPt, &vUpVec );
m_pd3dDevice->SetTransform( D3DTS_VIEW, &matView );
```

ตารางที่ 10.6 การกำหนดค่ากล้อง

10.6.1.6 สร้างแหล่งกำเนิดแสง โดยใช้เป็นชนิด D3DLIGHT_DIRECTIONAL

```
D3DLIGHT9 light;
D3DUtil_InitLight( light, D3DLIGHT_DIRECTIONAL, -2.0f, -5.0f, 0.0f );
m_pd3dDevice->SetLight( 0, &light );
m_pd3dDevice->LightEnable( 0, TRUE );
m_pd3dDevice->SetRenderState( D3DRS_LIGHTING, TRUE );
```

ตารางที่ 10.7 การสร้างแหล่งกำเนิดแสง

10.6.1.7 สร้างวินโดว์ซ็อกเก็ต ใช้วินโดว์ซ็อกเก็ตแบบดาต้าแกรมในการติดต่อกับ 2D Processing Module, Low Body Detecting Module และ Feedback Module

```
SOCKET virtualSock;
SOCKADDR_IN serverAddr;
virtualSock = socket(AF_INET, SOCK_DGRAM , 0);
serverAddr.sin_family = AF_INET;
serverAddr.sin_port = SERVER_PORT;
serverAddr.sin_addr.s_addr = INADDR_ANY;
```

```
memset(&(serverAddr.sin_zero), 0, 8);
bind(virtualSock, (LPSOCKADDR)&serverAddr, addr_len);
```

ตารางที่ 10.8 ซ็อกเก็ตสำหรับติดต่อสื่อสารกับโมดูลอื่น

10.6.2 อัพเดทอินพุท (Update Input)

เป็นส่วนของการรับอินพุทเพื่อนำไปประมวลผลในเกม โดยรับอินพุทจาก 2 ส่วน

10.6.2.1 3D Processing Module

รับอินพุทมาจากส่วน 2D Processing Module ผ่านทางฟังก์ชัน recvfrom( )ของซ็อกเก็ต แล้วนำข้อความ (recvStr) ที่ได้มาวิเคราะห์หาตำแหน่งของร่างกายผู้เล่น ณ เวลานั้นๆ ซึ่งมีรูปแบบของโปรโตคอลดังนี้

ภาพด้านซ้าย-ขวา(L/R)	เวลา	ตำแหน่งของส่วนต่างๆในระนาบ 2 มิติจำนวน 7 จุด

ตารางที่ 10.9 โปรโตคอลของการรับข้อมูลจาก 2D Processing Module

```
recvfrom(virtualSock, recvStr, 255, 0,
(LPSOCKADDR)&imageAddr,&addr_len);

pointTmp[0] = recvStr[i++];
pointTmp[1] = recvStr[i++];
leftImageInput[leftSlot].sec = atoi(pointTmp);

pointTmp[0] = recvStr[i++];
pointTmp[1] = ' ';
pointTmp[2] = ' ';
leftImageInput[leftSlot].msec = atoi(pointTmp);

pointTmp[0] = recvStr[i++];
pointTmp[1] = recvStr[i++];
pointTmp[2] = recvStr[i++];
leftImageInput[leftSlot].pRHand[0] = atoi(pointTmp);
```

```
pointTmp[0] = recvStr[i++];
pointTmp[1] = recvStr[i++];
pointTmp[2] = recvStr[i++];
leftImageInput[leftSlot].pRHand[1] = atoi(pointTmp);
...
Calculate3DPoint('l');
```

ตารางที่ 10.10 การวิเคราะห์หาข้อมูลจากข้อความที่ได้จาก 2D Processing Module

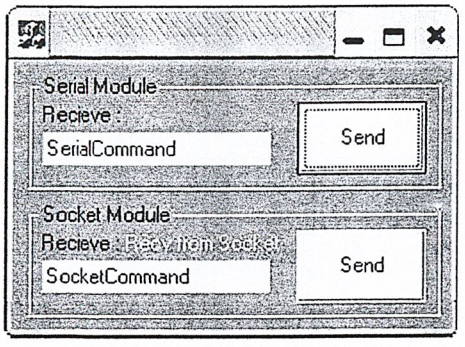
หลังจากวิเคราะห์หาข้อมูลได้แล้วจะนำไปตรวจสอบว่า ณ เวลานั้นๆ ข้อมูลได้ครบจากทั้ง 2 กล้องหรือไม่ ถ้าไม่ครบก็รอรับต่อไป ถ้าครบแล้วจะนำข้อมูลจากทั้ง 2 กล้องไปคำนวณหาตำแหน่งของร่างกายในพิกัด 3 มิติของเวิร์ล ซึ่งรองรับโดยคลาส Stereopsis

```
Stereopsis opsis;
opsis.Create3DPosition(leftImageInput[leftUse].pHead[0],leftImageInput[leftUse].pHead[1],
        rightImageInput[rightUse].pHead[0],rightImageInput[rightUse].pHead[1]);
currentImageInput.pHead[0] = opsis.getX();
currentImageInput.pHead[1] = opsis.getY();
currentImageInput.pHead[2] = opsis.getZ();
...
```

ตารางที่ 10.11 แปลงค่าอินพุทที่ได้เป็นพิกัด 3 มิติของเวิร์ล

10.6.2.2 Low Body Communication Module

เป็นแอพพลิเคชันที่สร้างด้วยภาษา Visual Basic.NET ทำหน้าที่รับอินพุทมาจากส่วน Low Body Capturing Module แล้วนำข้อมูลมาส่งต่อให้ Game Engine ผ่านวินโดว์ซ็อกเก็ตแบบดาต้าแกรม เพื่อลดภาระการติดต่อกับฮาร์ดแวร์ (I/O) ของเกมเอนจิ้น ซึ่งตัวแอพพลิเคชันมียูเซอร์อินเตอร์เฟส และโปรโตคอล ดังนี้



รูปที่ 10.31 Low Body Communication Module

อินพุทจาก Low Body Module (S)	ทิศทางการเคลื่อนที่ของผู้เล่น

ตารางที่ 10.12 โปรโตคอลของการรับข้อมูลจาก 2D Processing Module

Game Engine จะได้รับข้อมูลผ่านทางฟังก์ชัน recvfrom( ) ของซ็อกเก็ต แล้วนำข้อความ (recvStr) ที่ได้มาวิเคราะห์หาข้อมูลการเคลื่อนที่ของผู้เล่น

```
switch(recvStr[1])
{
        case 'F' :
                m_walkFB = 2.0f;
                break;
        case 'B' :
                m_walkFB = -2.0f;
                break;
        case 'L' :
                m_walkLR = 3.0f;
                break;
        case 'R' :
                m_walkLR = -3.0f;
                break;
}
```

ตารางที่ 10.13 การวิเคราะห์หาข้อมูลการเคลื่อนที่ของผู้เล่น

10.6.3 เฟรมมูฟ (Frame Move)

เป็นส่วนหลักของ Game Engine ที่ใช้คำนวณอินพุทและเงื่อนไขต่างๆ ของเกม

10.6.3.1 การเคลื่อนที่ของผู้เล่น

ผู้เล่นสามารถเคลื่อนที่ได้ 4 ทิศทาง คือ เดินหน้า ถอยหลัง และ เคลื่อนที่ซ้ายขวาเป็นวงกลมรอบตัวศัตรู

```
charAtX += m_fElapsedTime * m_UserInput.fAxisWalk * charLookatX;
charAtZ += m_fElapsedTime * m_UserInput.fAxisWalk * charLookatZ;
float distanceX = enemyAtX-charAtX;
float distanceZ = enemyAtZ-charAtZ;
m_distance = sqrt(distanceX*distanceX + distanceZ*distanceZ);
charLookatX = distanceX/m_distance;
charLookatZ = distanceZ/m_distance;
```

ตารางที่ 10.14 การคำนวณตำแหน่งของผู้เล่น

10.6.3.2 การเคลื่อนไหวของผู้เล่น

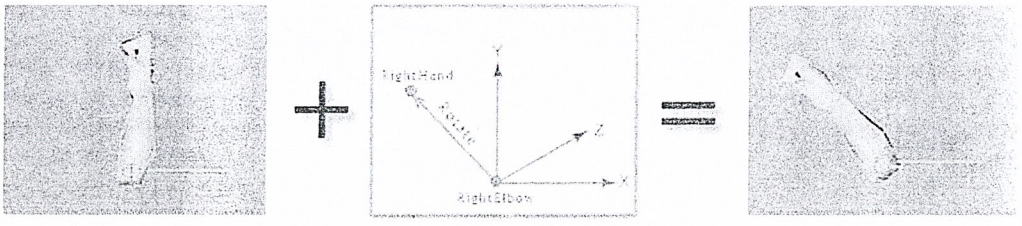
ผู้เล่นสามารถเคลื่อนที่ได้อิสระภายในขอบเขตของกล้อง โดยการเคลื่อนที่ของตัวละครมีความราบรื่นจากฟังก์ชันที่คอยคำนวณตำแหน่งของร่างกาย ณ ช่วงเวลาที่ข้อมูลขาดหายไป

```
If( abs(oldImageInput.pHead[0]currentImageInput.pHead[0])
<3*abs(difImageInput.pHead[0]) )
        oldImageInput.pHead[0] = currentImageInput.pHead[0];
else oldImageInput.pHead[0] += difImageInput.pHead[0];
if( abs(oldImageInput.pHead[1]currentImageInput.pHead[1])
<3*abs(difImageInput.pHead[1]) )
        oldImageInput.pHead[1] = currentImageInput.pHead[1];
else oldImageInput.pHead[1] += difImageInput.pHead[1];
if( abs(oldImageInput.pHead[2]currentImageInput.pHead[2])
<3*abs(difImageInput.pHead[2]) )
        oldImageInput.pHead[2] = currentImageInput.pHead[2];
else oldImageInput.pHead[2] += difImageInput.pHead[2];
```

ตารางที่ 10.15 การคำนวณการเคลื่อนไหวของผู้เล่นในส่วนของข้อมูลที่ขาดหาย

10.6.3.3 การวางชิ้นส่วนร่างกายของตัวละครฝ่ายผู้เล่นในพิกัด 3 มิติ ของเวิร์ล

การหมุนและการวางชิ้นส่วนของร่างกายตัวละครผู้เล่นจะวางตามเวกเตอร์ลัพธ์ที่ได้จากการคำนวณตำแหน่งข้อต่อของผู้เล่น



รูปที่ 10.32 การวางชิ้นส่วนร่างกายของผู้เล่น

```
m_cRotate.rRElbow[0] = oldImageInput.pRElbow[0] - oldImageInput.pRHand[0];
m_cRotate.rRElbow[1] = oldImageInput.pRElbow[1] - oldImageInput.pRHand[1];
m_cRotate.rRElbow[2] = oldImageInput.pRElbow[2] - oldImageInput.pRHand[2];
D3DXVec3Normalize(&m_cRotate.rRElbow, &m_cRotate.rRElbow);
```

ตารางที่ 10.16 การคำนวณเวกเตอร์ลัพธ์ในการหมุนชิ้นส่วนร่างกาย

10.6.3.4 การเคลื่อนไหวของตัวละครฝ่ายศัตรู

ศัตรูสามารถเคลื่อนไหวได้เองผ่านฟังก์ชัน AIofEnemy() ซึ่งการเคลื่อนไหวที่ได้ จะนำไปอัพเดทแอนิเมชันของตัวละครฝ่ายศัตรู

```
AIofEnemy();
switch(enemyActMode){
case 0: enemyAct = "stand";
        enemyActTime = (rand()%5+1) * 250 / combatFactor; break;
case 1: enemyAct = "forward"; enemyActTime = 350; break;
....
case 5: enemyAct = "punchLeft"; enemyActTime = 350; break;
case 6: enemyAct = "punchRight"; enemyActTime = 350; break;
....
```

ตารางที่ 10.17 การแสดงแอนิเมชันของตัวละครฝ่ายศัตรู

10.6.3.5 การต่อสู้

ตรวจสอบการชกจากระยะห่างระหว่างตัวละคร (m_distance) และ ท่าทางของตัวละคร เมื่อมีการชก พลังของตัวละครฝ่ายถูกชกจะลดลง และหน้าจอจะเกิดการสั่น

```
if (enemyActTime == 120)
if ((enemyActMode == 5) || (enemyActMode == 6))
{
        if(m_distance<4.5f)
        {
                charBloodTmp = charBlood;
                if(charBlood>0) charBlood -= 10;
                charHitTime = 180;
                m_pPunchSound->Play();
        }
}
```

ตารางที่ 10.18 การตรวจสอบการต่อสู้

10.6.3.6 มุมมองในเกม

ผู้เล่นสามารถเลือกมุมมองได้ 3 แบบคือ

- FIRST PERSON VIEW มุมมองบุคคลที่หนึ่ง (มองจากตาของตัวละคร)
- THIRD PERSON VIEW มุมมองบุคคลที่สาม (มองจากด้านหลังของตัวละคร)
- FREE VIEW มุมมองอิสระ (สามารถเลื่อนกล้องไปได้อิสระ)

10.6.4 เรนเดอร์ (Render)

เป็นส่วนของการแสดงภาพออกทางจอมอนิเตอร์

10.6.4.1 Render World

ส่วนนี้จะแสดงภาพของเวิร์ลทั้งหมด คือ สกายบ็อก สนามต่อสู้ ตัวละครฝ่ายศัตรู ตัวละครฝ่ายผู้เล่น แสงเงา แผนที่ตัวละคร (เป็นมุมมองแบบ 3 มิติจากมุมสูงโดยใช้สัญลักษณ์แทนตำแหน่งและทิศทางการหันหน้าของตัวละคร)

รูปที่ 10.33 แผนที่ตัวละคร

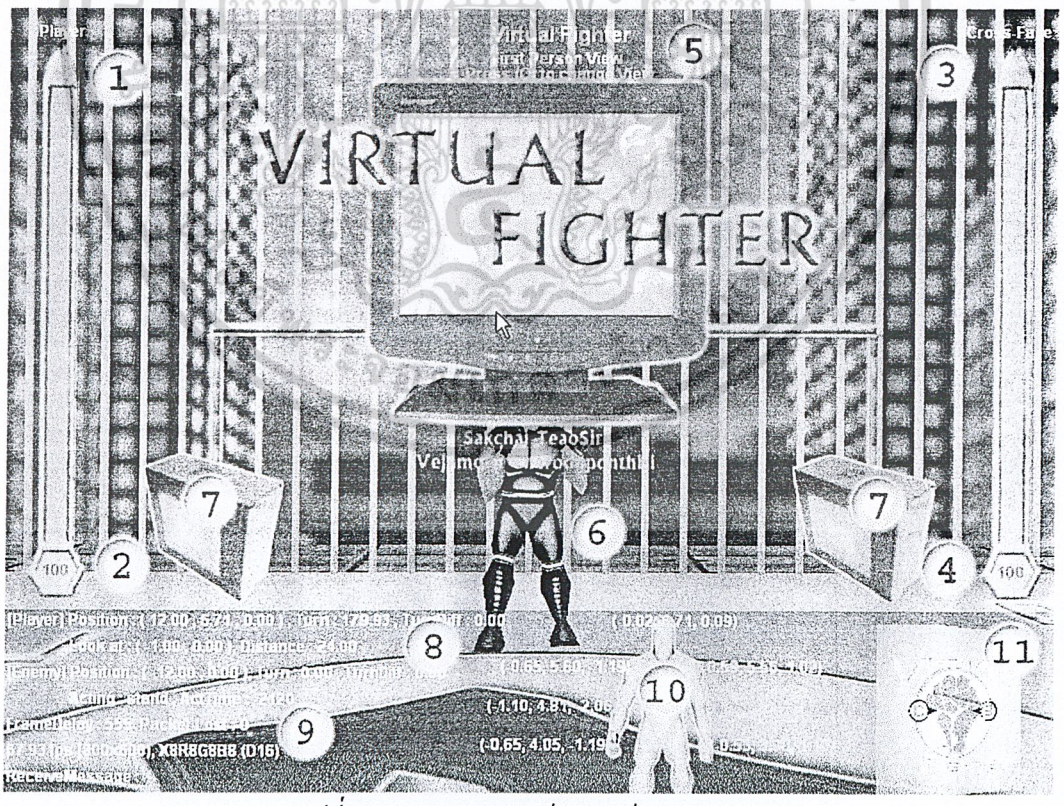
10.6.4.2 Render 2D

ส่วนนี้จะแสดงพลังของตัวละคร รวมทั้งพื้นหลังที่ใช้ในการตกแต่งอินเตอร์เฟสของเกม

10.6.4.3 Render Text

ส่วนนี้จะแสดงค่าอินพุทในพิกัด 3 มิติของชิ้นส่วนต่างๆ ของตัวละครฝ่ายผู้เล่น ตำแหน่งของตัวละครในเวิร์ล ระยะห่าง ข้อความที่ได้รับจากซ็อกเก็ต(ก่อนนำไปวิเคราะห์) โหมดการมอง จำนวนเฟรมที่เครื่องสามารถเรนเดอร์ได้ต่อหนึ่งหน่วยวินาที

10.6.5 ยูสเซอร์อินเตอร์เฟส



รูปที่ 10.34 แสดงยูสเซอร์อินเตอร์เฟสของเกม

- จุดที่ 1 แสดงหลอดพลังของผู้เล่น
- จุดที่ 2 แสดงจำนวนพลังของผู้เล่น
- จุดที่ 3 แสดงหลอดพลังของศัตรู
- จุดที่ 4 แสดงจำนวนพลังของศัตรู
- จุดที่ 5 แสดงมุมมองปัจจุบัน (กด O เพื่อเปลี่ยนมุมมอง)
- จุดที่ 6 แสดงตัวละครศัตรู
- จุดที่ 7 แสดงสถานการณ์เชื่อมต่อจากเครื่องประมวลผลการเคลื่อนไหวของผู้เล่น
- จุดที่ 8 ตำแหน่งของผู้เล่น ตำแหน่งของศัตรู ระยะห่าง ทิศทางการมอง ทิศทางการหัน ท่าทางของศัตรู
- จุดที่ 9 จำนวนแพ็กเกจที่สูญหาย จำนวนเฟรมต่อวินาทีที่เครื่องแสดงภาพได้ ขนาดหน้าจอที่ใช้เล่นเกม ข้อความที่ได้รับผ่านซ็อกเก็ต
- จุดที่ 10 ตำแหน่งของข้อต่อร่างกายผู้เล่นในพิกัด 3 มิติ ที่เป็นอินพุทของเกม
- จุดที่ 11 แผนที่ตัวละคร แสดงตำแหน่งของผู้เล่นและศัตรูในแบบ 3 มิติจากมุมสูง

บทที่ 11
การทดสอบ

11.1 การทดสอบการตรวจจับการเคลื่อนไหวของผู้เล่น

การทดสอบการประมวลผลหาตำแหน่งของผู้เล่นในขั้นแรกจะทดสอบกับไฟล์วีดีโอที่ทำการจับภาพมาได้ก่อน แล้วจึงทดสอบด้วยการประมวลผลหาตำแน่งมาร์กเกอร์แบบเรียลไทม์ โดยจะทำการทดสอบแต่ละส่วนของโปรแกรม ดังนี้

11.1.1 การทดสอบการกำหนดค่าช่วงสีของมาร์กเกอร์

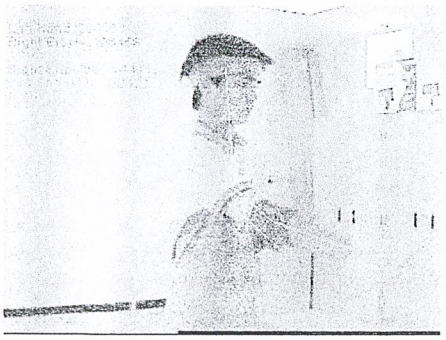
ถ้ากำหนดค่าช่วงสีของมาร์กเกอร์มีขนาดกว้างจนเกินไปจะทำให้ช่วงสีของมาร์กเกอร์เหลื่อมล้ำกัน ซึ่งโปรแกรมอาจเข้าใจผิดว่ามาร์กเกอร์อยู่ที่ตำแหน่งเดียวกันบนภาพ

รูปที่ 11.1 การหาตำแหน่งที่ผิดพลาดของมาร์กเกอร์

แก้ไขโดยใช้มาร์กเกอร์ที่มีค่าเฉลี่ยของสีบนมาร์กเกอร์ แตกต่างกันมากขึ้น และกำหนดความกว้างของช่วงสีให้ลดลง ปัญหาค่าสีที่เหลื่อมล้ำของค่าสีจะลดลง

จากการทดลองพบว่าถ้ากำหนดความกว้างของช่วงค่า H ขยายจากค่าเฉลี่ยไปทางบวก และลบข้างละ 4 องศา และ ช่วงค่า S,I ขยายจากค่าเฉลี่ยไปทางบวก และลบข้างละ 10% จะทำให้การหาตำแหน่งมาร์คเกอร์ได้ผลดีที่สุด

11.1.2 การทดสอบการหาตำแหน่งมาร์กเกอร์ โดยที่บางมาร์กเกอร์ถูกบดบัง หรือ หายไปจากภาพ



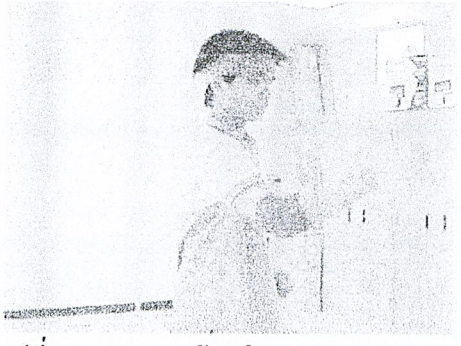
รูปที่ 11.2 มาร์กเกอร์ถูกบัง

รูปที่ 11.3 มาร์กเกอร์หายไปจากภาพ

ภาพที่รับมาจากกล้องวีดีโอบางครั้ง บางมาร์กเกอร์ถูกบดบัง หรือหายไปจากภาพ ทำให้เฟรมนั้นไม่สามารถหาตำแหน่งมาร์กเกอร์นั้นได้แก้ไขโดยให้ใช้ตำแหน่งเดิมของมาร์กเกอร์นั้นในเฟรมที่แล้ว เป็นตำแหน่งของมาร์กเกอร์ในเฟรมปัจจุบัน

11.1.3 การตั้งกล้องห่างเกินไป

การตั้งกล้องห่างเกินไป มุมการมองเห็นจะแคบลง ซึ่งอาจทำให้เกิดการบดบังของมาร์กเกอร์ได้



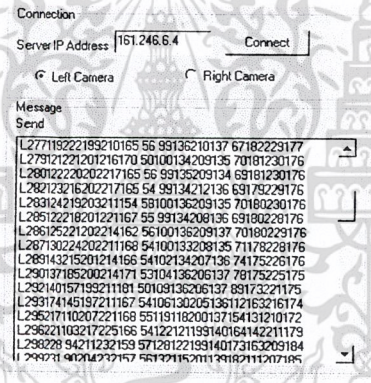
รูปที่ 11.4 การวางกล้องห่างกัน 240 เซนติเมตร

รูปที่ 11.5 การวางกล้องห่างกัน 180 เซนติเมตร

การตั้งกล้องที่เหมาะสมที่สุดในโครงงานนี้ คือการตั้งกล้องห่างกัน 180 เซนติเมตร ดังรูปที่ 10.8

11.1.4 การส่งข้อมูลตำแหน่งมาร์กเกอร์

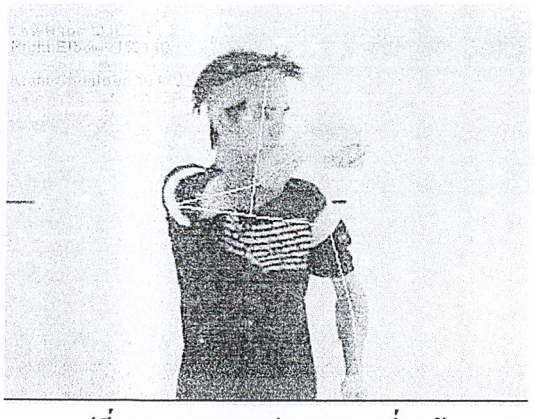
การส่งข้อมูลของมาร์กเกอร์ได้สามารถส่งได้ประมาณ 9-10 ชุดต่อวินาที



รูปที่ 11.6 การส่งข้อมูลการเคลื่อนไหวไปให้ส่วนเกม

11.1.5 การทดสอบหลังจากแก้ปัญหา

การหาตำแหน่งมีความแม่นยำมากยิ่งขึ้น โดยมีการหาตำแหน่งมาร์กเกอร์ที่ถูกต้องประมาณ 90 % บางภาพถ้ามาร์กเกอร์ถูกบดบังหรือ หายไปจากภาพ ก็ให้ใช้ตำแหน่งเดิมของมาร์กเกอร์นั้นในเฟรมที่แล้ว แต่ถ้าการหาตำแหน่งผิดพลาดในมาร์กเกอร์ใดมีการผิดพลาดมาก(เช่น การตรวจเจอว่าตำแหน่งของศอกซ้ายอยู่ที่เดียวกับมือขวาตลอด ซึ่งอาจเกิดจากการเหลื่อมล้ำกันของช่วงสีของมาร์กเกอร์) ก็ให้ไปปรับค่าเฉลี่ยของมาร์กเกอร์นั้นใหม่ และการประมวลผลภาพสามารถทำได้ 14-15 เฟรมต่อวินาที แต่จะทำการส่งข้อมูลแค่ 10 เฟรมต่อวินาที เพื่อง่ายต่อการทำซิงโครไนซ์



รูปที่ 11.7 แสดงการประมวลผลที่ถูกต้อง

11.2 การทดสอบฮาร์ดแวร์

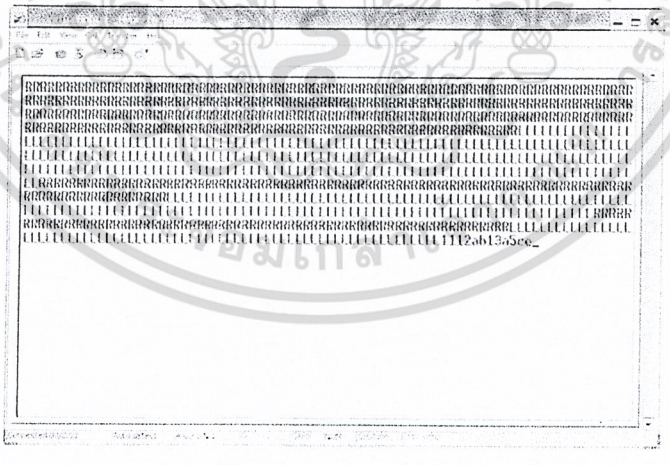
เมื่อสร้างวงจรฮาร์ดแวร์หมดแล้วสามารถทดสอบข้อมูลที่รับส่งระหว่างไมโครคอนโทรลเลอร์และ คอมพิวเตอร์อย่างง่ายด้วย ไฮเปอร์เทอมินอล(Hyper Terminal) ของระบบปฏิบัติการวินโดว์ โดยต้องตั้งค่าดังนี้

Connection Using : COM1

Bits per Second : 9600

Flow Control : Xon/Xoff

ที่เหลือให้ใช้ค่าเดิม

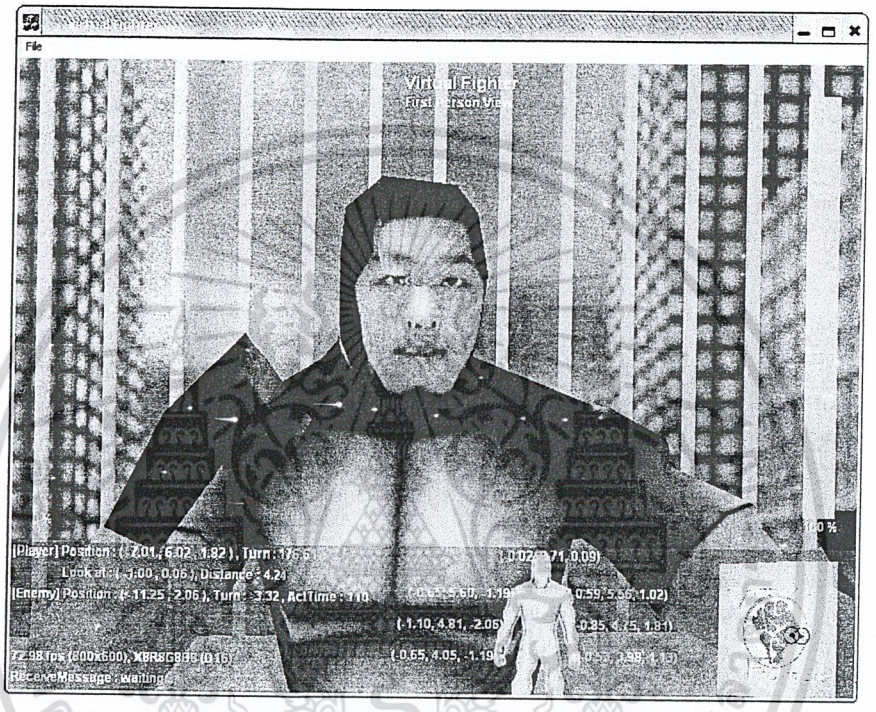


รูปที่ 11.8 การติดต่อระหว่างไมโครคอนโทรลเลอร์และ คอมพิวเตอร์ด้วย ไฮเปอร์เทอมินอล

ซึ่งจากการทดลอง การติดต่อมีการหน่วงเวลาระหว่างฮาร์ดแวร์และเครื่องคอมพิวเตอร์เพิ่มขึ้นเล็กน้อย เมื่อมีการติดต่อหลายอย่างในช่วงเวลาเดียวกัน เช่น มีการเดินหน้า ผู้เล่นโจมตีศัตรู และผู้เล่นโดนโจมตี จะมีการส่งข้อมูล 3 อย่างในเวลาเดียวกัน

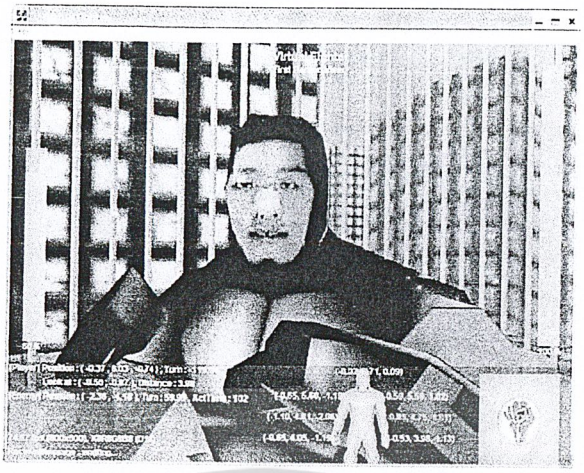
11.3 การทดสอบเกม

11.3.1 ทดสอบการเคลื่อนที่และการแสดงท่าทางของตัวละครฝ่ายศัตรู เช่น การเดิน การชก การหลบหลีก จากการทดสอบพบว่าในระยะที่ห่างจากตัวผู้เล่น ศัตรูจะเดินเพื่อหาจังหวะเข้ามาโจมตี และเมื่อผู้เล่นอยู่ในระยะที่ศัตรูสามารถชกโดน ศัตรูจะแสดงท่าทางการชก และถ้าผู้เล่นมีการชก ศัตรูสามารถยกแขนมาป้องกันได้



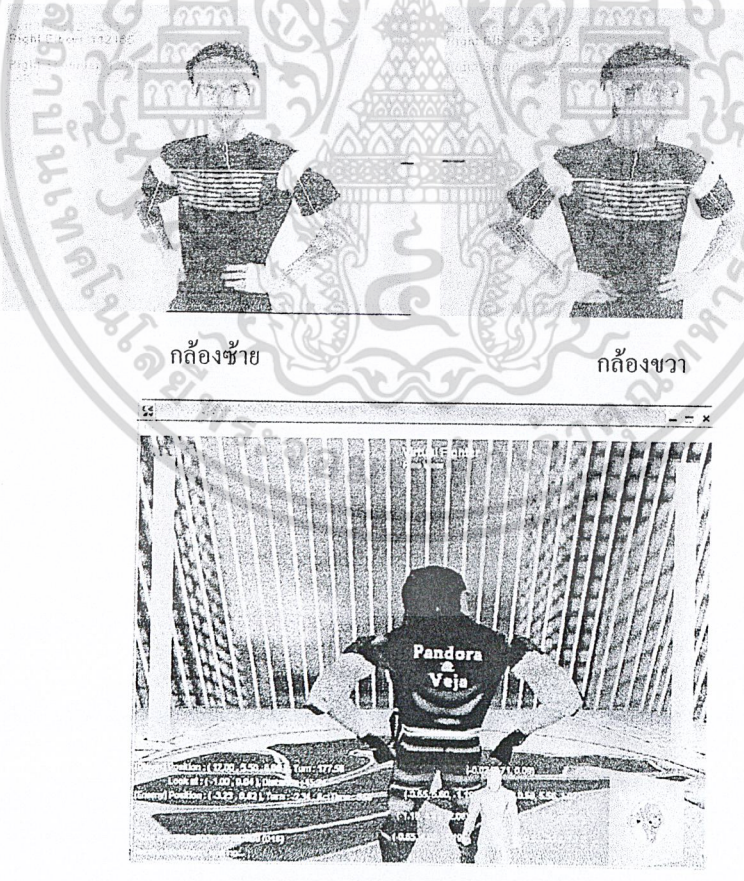
รูปที่ 11.9 แสดงลักษณะการชกของตัวละครฝ่ายศัตรู

11.3.2 ทดสอบการชกและการลดของหลอดพลัง ศัตรูจะชกเมื่อผู้เล่นอยู่ในระยะห่างที่สามารถชกถึง จากการทดสอบพบว่า ถ้าศัตรูชกและผู้เล่นไม่สามารถถอยพ้นจากระยะการชก พลังของผู้เล่นจะลดลงและเกิดการสั่นที่มอเตอร์ในตำแหน่งที่ถูกชก แต่ถ้าผู้เล่นสามารถยกแขนขึ้นมาป้องกันได้ พลังของผู้เล่นจะลดไม่มาก (ประมาณครึ่งหนึ่งของการโดนชกปกติ)



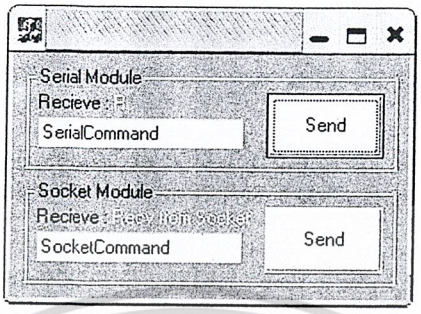
รูปที่ 11.10 แสดงการโดนชกและการลดของหลอดพลัง

11.3.3 ทดสอบการแสดงท่าทางของตัวละคร โดยรับอินพุทจากลักษณะท่าทางของผู้เล่นที่ตรวจจับได้ จากการทดสอบพบว่าตัวละครแสดงท่าทางตามการกระทำผู้เล่น โดยมีระยะเวลาหน่วงประมาณ 0.3 -0.5 วินาที



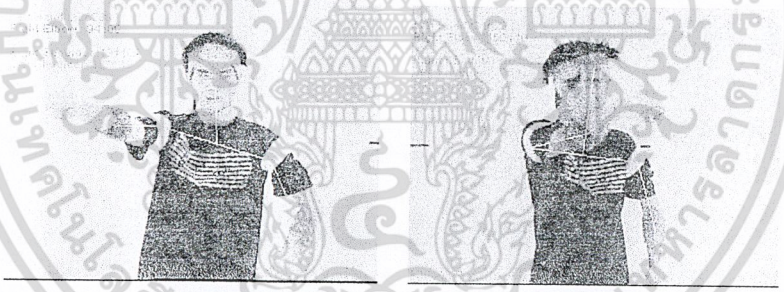
รูปที่ 11.11 การแสดงท่าทางของตัวละคร

11.3.4 ทดสอบการรับอินพุทจากส่วนการตรวจจับการเคลื่อนที่ โดยรับมาจากคอมพอร์ตแล้วส่งต่อให้เกมเอนจิ้น ซึ่งผลที่ได้คือตัวละครฝ่ายผู้เล่นสามารถขยับไปในทิศทางตามที่ผู้เล่นต้องการ

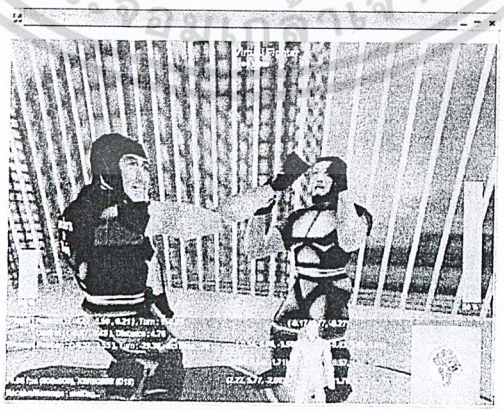


รูปที่ 11.12 รับค่าการเดินไปทางขวา (R)

11.3.5 ทดสอบการชกของผู้เล่น ผู้เล่นแสดงท่าทางการชกใส่ตัวศัตรู จากการทดสอบพบว่า ถ้าศัตรูอยู่ในระยะที่ผู้เล่นสามารถชกถึง พลังของศัตรูจะลดลงพร้อมทั้งแสดงท่าทางการถูกชก แต่ถ้าศัตรูสามารถยกแขนขึ้นมาป้องกันได้ พลังของศัตรูจะลดลงเป็นครึ่งหนึ่งของการโดนชกปกติ โดยเมื่อผู้เล่นชกโดนศัตรูมอเตอร์ที่ส่วนมือข้างที่ชกของผู้เล่นจะเกิดการสั่น

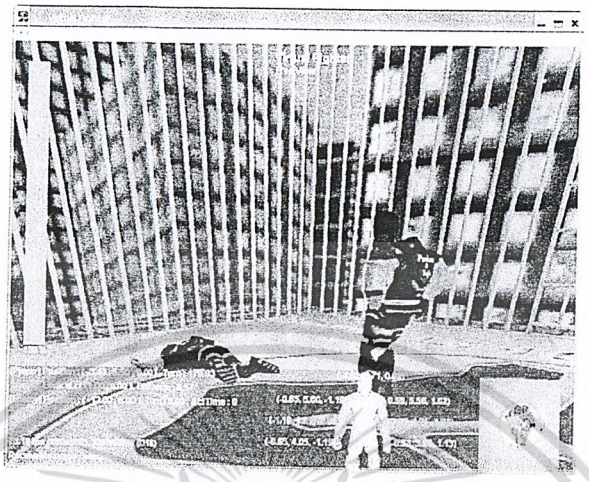


กล้องซ้าย กล้องขวา



รูปที่11.13 ศัตรูโดนผู้เล่นชก

11.3.6 ทดสอบการจบเกม จากการทดสอบพบว่า เมื่อผู้เล่นฝ่ายใดฝ่ายหนึ่งพลังหมดถือเป็นการจบเกม และสามารถเริ่มเกมใหม่ได้โดยการกด R



รูปที่11.14 ผู้เล่นชนะจากการทำให้ศัตรูพลังหมด

บทที่ 12
บทวิจารณ์และสรุป

12.1 ประเมินผล

จากผลลัพธ์ของโครงงานนี้ที่ได้ออกมาเมื่อเทียบกับวัตถุประสงค์ที่ได้กำหนดไว้ในตอนแรกแล้ว ถือว่าโครงงานนี้บรรลุตามวัตถุประสงค์ โดยผลลัพธ์ของโครงงานได้ออกมาในรูปเกมต่อสู้ 3 มิติ ซึ่งเน้นในด้านการนำเอาการตรวจจับการเคลื่อนไหว (Motion Capturing) มาประยุกต์ใช้กับเกมในแบบเรียลไทม์ ผู้เล่นสามารถแสดงท่าทางเพื่อบังคับการเคลื่อนไหวต่างๆของตัวละครในเกม เช่น การชก การป้องกัน การเคลื่อนที่ และผู้เล่นจะได้รับแรงตอบสนองเมื่อมีการชกหรือถูกชก ทำให้การผู้เล่นรู้สึกมีส่วนร่วมกับการเล่นเกมมากขึ้น

12.2 แนวทางการพัฒนาต่อ

จุดประสงค์ของการสร้างเกมที่ผู้พัฒนาเกมทุกคนต้องการนั้นคือ พยายามทำให้ผู้เล่นมีส่วนร่วมกับการเล่นเกมให้มากที่สุด ซึ่งมีหลายแนวทางที่สามารถทำได้ เช่น ภาพสเปเชียลกราฟฟิกที่สวยงาม เสียงประกอบที่สมจริง เนื้อหาของเกมที่น่าติดตาม หรือใช้ส่วนประกอบทางฮาร์ดแวร์อื่นๆ

โครงงานนี้เลือกใช้การตรวจจับการเคลื่อนไหวของผู้เล่น โดยใช้กล้องจำนวน 2 ตัวในการจับภาพและคำนวนตำแหน่งมาร์กเกอร์สีที่ติดอยู่ตามตำแหน่งข้อต่อของร่างกายผู้เล่น ซึ่งเป็นฮาร์ดแวร์ที่ลงทุนน้อย และได้ผลที่น่าพอใจ แต่ด้วยการใช้กล้องเพียง 2 ตัว อาจทำให้เกิดการบดบังตำแหน่งของมาร์กเกอร์ ทำให้ค่าตำแหน่งที่คำนวณได้อาจจะผิดเพี้ยนไป อาจจแก้ไขโดยเพิ่มจำนวนกล้องที่ใช้จับภาพ หรือเปลี่ยนไปใช้ระบบเชิงกลในการตรวจจับการเคลื่อนไหว แต่ก็ต้องยอมรับในเรื่องค่าใช้จ่ายที่อาจสูงตามไปด้วย

ในด้านการพัฒนาเกมนั้น อาจพัฒนาต่อได้โดย เพิ่มท่าทางการเคลื่อนไหวของตัวละครฝ่ายศัตรูให้มีความหลากหลายมากขึ้น เพิ่มระบบปัญญาประดิษฐ์เพื่อทำให้ตัวละครฝ่ายศัตรูสามารถตอบสนองกับผู้เล่นได้อย่างสมจริงมากขึ้น เพิ่มการอาวุธเข้าไปในเกม เช่น อาจใช้ดาบฟันแทนการใช้หมัดชก หรือทำให้เกมสามารถเล่นกับผู้เล่นคนอื่นๆได้ โดยใช้วิธีการติดต่อผ่านเครือข่าย

บรรณานุกรม

[1] Satoshi YONEMOTO, Asuka MATSUMOTO, Daisaku ARITA, Rin-ichiro TANIGUCHI, *A Real-time Motion Capture System with Multiple Camera Fusio,* Department of Intelligent Systems, Kyushu University

[2] C.Wren, A.Azarbayejani, T.Darrell, A.Pentland,"Pfinder: Real-Time Tracking of the Human Body", *IEEE Transactions on Pattern Analysis and Machine Intelligence,1997*

[3] Rafael C. Gonzalez, Richard E. Woods, *Digital Image Processing,* Prentice Hall, 2002

[4] David A. Forsyth, Jean Ponce, *Computer Vision A Modern Approach,* Prentice Hall, 2003

[5] Frank D. Luna, *Introduction to 3D GAME programming with DirectX 9.0,* Wordware Publishing, 2003

[6] Peter Walsh, *Advanced 3D Game Programming with DirectX 9.0,* Wordware Publishing, 2003

[7] Norman I. Badler, Cary B. Phillips, Bonnie L. Webber, *SIMULATING HUMANS: COMPUTER GRAPHICS, ANIMATION, AND CONTROL,* Oxford University Press, 1999