โปรแกรมจำลองสถานการณ์เสมือนจริงย่านแว่นตาสามมิติเพื่อการจัดการความปลอดภัย

Virtual 3D Simulation with Head Mounted Display

นางสาวภาวินี วงษ์วิทิต

MS. PHAWINEE WONGVITIT

นางสาวมธวี รุ่งแสงจันทร์

MS. MATHAWEE ROONGSANGCHAN

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต

สาขาวิชาวิศวกรรมอุตสาหการ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2556

โปรแกรมจำลองสถานการณ์เสมือนจริงผ่านแว่นตาสามมิติเพื่อการจัดการความปลอดภัย

Virtual 3D Simulation with Head Mounted Display

นางสาวภาวินี วงษ์วิฑิต

MS. PHAWINEE WONGVITIT

นางสาวเมธาวี รุ่งแสงจันทร์

MS. MATHAWEE ROONGSANGCHAN

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมอุตสาหการ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2556

Virtual 3D Simulation with Head Mounted Display

MS. PHAWINEE WONGVITIT

MS. MATHAWEE ROONGSANGCHAN

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

BACHELOR OF ENGINEERING IN INDUSTRIAL ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

ACADEMIC YEAR 2013

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองปริญญานิพนธ์

หัวข้อปริญญานิพนธ์	โปรแกรมจำลองสถานการณ์เสมือนจริงผ่านแว่นตาสามมิติเพื่อการจัดการความปลอดภัย Virtual 3D Simulation With Head Mounted Display
นักศึกษา	นางสาวภาวินี วงษ์วิฑิต รหัสประจำตัว 53011256 นางสาวเมธาวี รุ่งแสงจันทร์ รหัสประจำตัว 53011311
หลักสูตร	วิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมอุตสาหการ

อาจารย์ผู้ควบคุมปริญญานิพนธ์

(ดร.พลชัย โชติปรายนกุล)

หัวข้อปริญญานิพนธ์	โปรแกรมจำลองสถานการณ์เสมือนจริงผ่านแว่นตาสามมิติเพื่อการจัดการความปลอดภัย
นักศึกษา	นางสาวภาวินี วงษ์วิฑิต นางสาวเมธาวี รุ่งแสงจันทร์
หลักสูตร	วิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมอุตสาหการ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา	2556
อาจารย์ผู้ควบคุมปริญญานิพนธ์	ดร.พลชัย โชติปรายนกุล

บทคัดย่อ

การออกแบบการเคลื่อนที่ของคนผ่านโปรแกรมเพื่อจำลองการเคลื่อนที่นั้น เป็นสิ่งจำเป็นในการออกแบบและวางผังโรงงานก่อนทำการก่อสร้างสถานประกอบการหรือใช้ในการปรับปรุงสถานประกอบการเพื่อเพิ่มระดับความปลอดภัยในภายหลัง เป็นการป้องกันการสูญเสียที่อาจเกิดขึ้นทั้งทางตรง เช่น อุบัติเหตุที่ทำให้เกิดความสูญเสียต่อชีวิต และทางอ้อม เช่น การสูญเสียทรัพย์สินเงินทอง, เวลา นอกจากนี้ยังส่งผลกระทบถึงภาพลักษณ์ของสถานประกอบการนั้นๆด้วย ซึ่งทางผู้จัดทำวิทยานิพนธ์นี้ ได้ทำการเขียนโปรแกรมเพื่อจำลองสถานการณ์ของการเคลื่อนที่ของคนภายในสภาพแวดล้อมที่กำหนด ผ่านโปรแกรม MATLAB เพื่อศึกษาลักษณะการเคลื่อนที่ของคนไปยังเป้าหมายที่กำหนด เพื่อหาขีดความสามารถในการรองรับคนภายในบริเวณดังกล่าว ด้วยเหตุนี้การประยุกต์ใช้ซอฟแวร์ในการจำลองสถานการณ์ล่วงหน้า สามารถทำให้ทราบถึงระดับความปลอดภัยในปัจจุบัน และช่วยลดความสูญเสียต่างๆที่อาจเกิดขึ้นตามมาได้ โปรแกรมที่สร้างขึ้นจะเป็นเครื่องมือในการสนับสนุนการตัดสินใจในการจัดการความปลอดภัยอย่างมีประสิทธิภาพ ซึ่งจะเป็นประโยชน์อย่างยิ่งต่อผู้ประกอบการในการบริหารจัดการความปลอดภัยภายในสถานประกอบการได้เป็นอย่างดี

Thesis Tile	Virtual 3D Simulation with Head Mounted Display
Student	Ms. Phawinee Wongvitit
	Ms. Mathawee Roongsangchan
Degree	Bachelor of engineering in Industrial Engineering
	King Mongkut's Institute of Technology Ladkrabang
Academic Year	2013
Thesis Advisor	Dr. Pholchai Chotiprayanakul

ABSTRACT

Simulation of people's movement in virtual industrial environment is an essential technique in plant designing. It will save us from unnecessary expense in plant constructing investment. Safe workplace, which is a good image for the company, must has been considered since designing. For this simulation, several safety issues can be tested. Analyzing of the simulation result will convey improvement of plant design in order to prevent accidents and minimize its' direct and indirect loss. In this thesis, we programmed a simulation software on MATLAB. The simulation demonstrate people move or evacuate from a specific area to achieve their targets which are exit-points. Thus, this simulation software can predict an emergency event in advance and support our decision-making solution in safety problems. Also this will be a useful tool for safety management for any enterprises.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จได้ด้วยคำแนะนำและคำปรึกษาจาก ดร.พลชัย โชติปรายนกุล ซึ่งเป็นอาจารย์ผู้ให้คำปรึกษาควบคุมปริญญานิพนธ์ ทางกลุ่มผู้วิจัยขอขอบพระคุณเป็นอย่างสูงในความอนุเคราะห์จากอาจารย์ สำหรับการศึกษาปริญญานิพนธ์ฉบับนี้ รวมทั้งความรู้ คำแนะนำ การช่วยเหลือและความเอาใจใส่ในทุกๆ ด้านของปริญญานิพนธ์ตลอดระยะเวลาที่ผ่านมา ขอขอบพระคุณอาจารย์ภายในภาควิชาวิศวกรรมอุตสาหการทุกท่านที่ได้สั่งสอนอบรมแก่พวกเราตลอดมา สุดท้ายขอขอบคุณเพื่อนที่คอยให้คำปรึกษาในการทำปริญญานิพนธ์ฉบับนี้

นางสาวภาวินี วงษ์วิฑิต

นางสาวเมธาวี รุ่งแสงจันทร์

สารบัญ

สารบัญ (ต่อ)

สารบัญรูป

หน้า

สารบัญรูป (ต่อ)

หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ความปลอดภัยเป็นเรื่องที่สำคัญต่อภาคอุตสาหกรรมและภาคครัวเรือน อุบัติเหตุอาจก่อให้เกิดความสูญเสียต่อชีวิตและทรัพย์สิน ในปัจจุบันเทคโนโลยีทางด้านคอมพิวเตอร์ได้พัฒนาไปอย่างมากและการใช้เครื่องมือทางด้านคอมพิวเตอร์กราฟิกมาช่วยในการจำลองสถานการณ์หรือเหตุการณ์ฉุกเฉินมีความเสมือนจริงและรองรับการทำงานแบบเวลาจริงมากขึ้น การจำลองสถานการณ์หรืออุบัติภัยที่ก่อให้เกิดความเสียหายไว้ล่วงหน้าจะเป็นการเรียนรู้ถึงสภาพการณ์และสามารถคาดการสถานการณ์สำหรับปรับสิ่งแวดล้อมให้เหมาะสมให้พร้อมรับมือกับสถานการณ์ฉุกเฉินนั้น ๆ ได้อย่างมีประสิทธิภาพและช่วยลดความสูญเสียลงได้หากเกิดเหตุการณ์นั้นจริง ทางกลุ่มผู้จัดทำโครงการจึงต้องการเขียนโปรแกรมคอมพิวเตอร์ที่ใช้ในการจำลองสถานการณ์เสมือนจริงออกมาในลักษณะของรูปภาพสามมิติ เพื่อใช้เป็นในการศึกษาแนวทางการลดความสูญเสีย ความเสียหายหลังเกิดอุบัติเหตุขึ้น และโปรแกรมจะสามารถระบุถึงพื้นที่บริเวณที่ควรแก้ไขหากมีความแออัดในระหว่างการจำลองสถานะการณ์การอพยพ

1.2 วัตถุประสงค์

1. เพื่อสร้างโปรแกรมจำลองสภาพแวดล้อมเสมือนจริงตามที่กำหนด
2. เพื่อจำลองสถานการณ์ในสภาพแวดล้อมเสมือนจริงที่สร้างขึ้น
3. เพื่อใช้แบบจำลองในการวางแผนการจัดการด้านความปลอดภัย

1.3 ขอบเขตของปริญญานิพนธ์

เขียนโปรแกรมคอมพิวเตอร์ซึ่งสามารถจำลองสภาพสถานการณ์จริงในลักษณะของภาพสามมิติ ซึ่งเป็นการเขียนโปรแกรมจำลองการเคลื่อนที่ไปมาของวัตถุสมมติเพื่อใช้แทนคนจำนวน 50 คน โดยมีการจำกัดขอบเขตการเคลื่อนที่เสมือนเคลื่อนที่อยู่ในห้องทดลองจริงผ่านซอฟแวร์ MATLAB โดยผู้สังเกตุการณ์จะสามารถมองเห็นภาพจำลองและสามารถมองเห็นถึงมุมมองต่าง ๆ ภายในแบบจำลองขึ้นผ่านจอภาพหรือแว่นตาสามมิติ โดยซอฟแวร์ที่สร้างจะทำงานบนระบบปฏิบัติการวินโดว์ โดยเป็นการเขียนโปรแกรมบน Visual C++ และ ใช้โปรแกรม Open Graphics Library (OpenGL) ในการแสดงภาพสามมิติ

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1. เพื่อช่วยในการคาดการณ์การเคลื่อนไหวของคนอย่างสุ่มในขณะที่เกิดอุบัติเหตุต่างๆ ทำให้สามารถคาดการณ์และควบคุมสถานการณ์ที่อาจเกิดขึ้นในอนาคตได้

2. ใช้สนับสนุนระบบการตัดสินใจในการจัดวางผังโรงงานและประตูฉุกเฉินในกรณีเกิดอุบัติเหตุต่างๆได้

3. เพื่อใช้เป็นตัวอย่างในการนำเอาหลักการของการเขียนโปรแกรมคอมพิวเตอร์มาช่วยในการจัดการทางด้านความปลอดภัยในภาคอุตสาหกรรม

4. เพื่อใช้เป็นตัวช่วยในการวิเคราะห์เชิงสถิติเพื่อหาเวลาเฉลี่ยในการหนีออกจากตัวอาคารในกรณีเกิดอุบัติเหตุต่างๆได้

5. เพื่อเป็นการลดต้นทุนในการวางแผนผังโรงงานก่อนทำการก่อสร้างจริง

6. เพื่อใช้เป็นสื่อประกอบการเรียนการสอนของวิชาที่เกี่ยวข้อง

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

บทนำ

การสร้างโปรแกรมจำลองสถานการณ์เสมือนจริงผ่านแว่นตาสามมิติเพื่อการจัดการความปลอดภัย เป็นการศึกษาซึ่งเกี่ยวข้องกับการเขียนโปรแกรมจำลองเหตุการณ์เสมือนจริงผ่านโปรแกรมคอมพิวเตอร์ เพื่อนำไปสู่การจัดการความปลอดภัย การจัดวางผังโรงงาน การจัดวางเครื่องจักรภายในโรงงาน การเลือกขนาดของประตูโรงงานที่เหมาะสม และการวางระบบดับเพลิงหรืออุปกรณ์ดับเพลิง ผลจากการใช้โปรแกรมคอมพิวเตอร์ดังกล่าวส่งผลให้การจัดเตรียมสถานที่ทำงานมีความเหมาะสมในด้านความปลอดภัยมากขึ้น และสามารถลดความเสี่ยงและความสูญเสียลงได้หากเกิดเหตุการณ์นั้นจริง การทำปริญญานิพนธ์ฉบับนี้ได้ทำการศึกษาทฤษฎีที่เกี่ยวข้อง ดังกล่าวในหัวข้อต่อไป

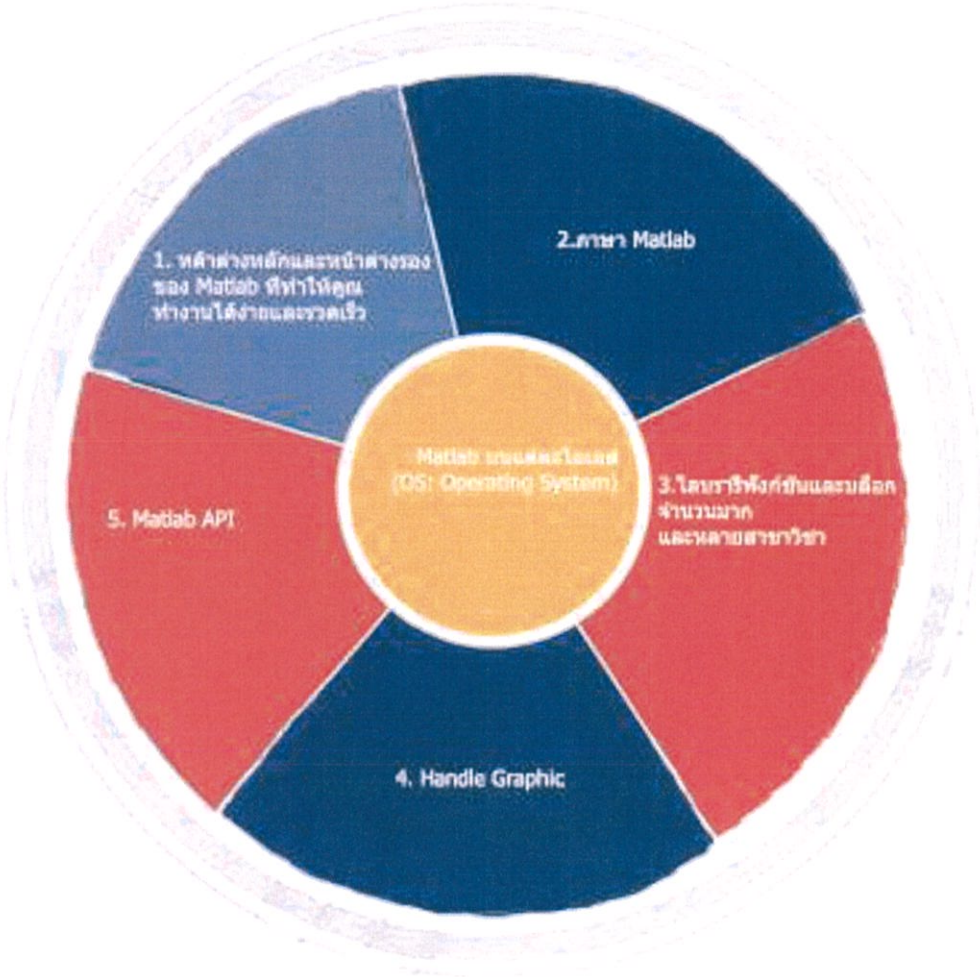
2.1 MATLAB [1]

โปรแกรม MATLAB (MATrix LABoratory) เป็นโปรแกรมของ The Mathworks, Inc. MATLAB เป็นภาษาคอมพิวเตอร์ระดับสูงที่ใช้สำหรับคำนวณเชิงตัวเลข (Numerical Computing) สำหรับใช้ในการทำงานคำนวนทางคณิตศาสตร์ที่ซับซ้อนและมีกระบวนการติดต่อกับผู้ใช้งานแสดงผลกราฟิกในทั้งแบบสองและสามมิติ และการเขียนแอพพลิเคชั่น ทำให้สามารถคำนวณผลลัพธ์ พัฒนาอัลกอลิทึ่ม สร้างแบบจำลอง และแอพพลิเคชั่นได้ง่าย รวดเร็ว

ภายในตัว MATLAB ประกอบด้วย ภาษาคอมพิวเตอร์แบบสคริปท์ ทูลบ็อกซ์(Toolbox) ฟังก์ชันทางคณิตศาสตร์พื้นฐาน ที่ทำให้ MATLAB มีความสามารถในการวิเคราะห์งานที่หลากหลาย และให้คำตอบได้อย่างรวดเร็วกว่าโปรแกรมแบบอื่นอย่างเช่นโปรแกรมตารางคำนวณ (Spreadsheet) และง่ายต่อการโปรแกรมมากกว่าภาษาคอมพิวเตอร์อื่น เช่น C, C++, Fortran, Java เป็นต้น ความสามารถดังกล่าวทำให้ MATLAB สามารถนำไปประยุกต์ใช้งานได้หลายสาขามาก ทั้ง การประมวลผลสัญญาณ (Signal Processing) การสื่อสาร (Communication) การประมวลผลภาพและวิดีโอ (Image and Video Processing) ระบบควบคุม (Control System) การวัดและควบคุม (Instruments and Control) การคำนวณทางเศรษฐศาสตร์ (Economic) การคำนวณทางชีววิทยา (Biology) และอื่นๆ

2.1.1 ส่วนประกอบสำคัญของ MATLAB [1]

MATLAB ได้ออกแบบมาเพื่อสนับสนุนการทำงานของผู้ใช้อยู่ 5 ส่วน เพื่อช่วยในการวิเคราะห์ข้อมูล แสดงผลข้อมูล เชื่อมต่อกับสิ่งต่างๆ ภายนอกได้อย่างรวดเร็วและมีประสิทธิภาพสูงสุด ดังแสดงในรูปที่ 2.1



รูปที่ 2.1 ส่วนประกอบภายในของ MATLAB [2]

1. หน้าต่างหลักและหน้าต่างรองของ MATLAB หรือที่เรียกกันว่า MATLAB Desktop Environment ช่วยให้ผู้ใช้ทำงานได้ง่ายและรวดเร็วมากยิ่งขึ้น ในส่วนนี้ประกอบด้วย ชุดเครื่องมือที่ช่วยให้สามารถใช้งานฟังก์ชันและไฟล์ต่าง ๆ ด้วยการปฏิสัมพันธ์แบบรูปภาพ (GUI) ประกอบด้วย หน้าต่างย่อยของ Editor, Command History, Command Windows และ Workspace

2. ภาษา MATLAB ใช้สำหรับเขียนแอพพลิเคชั่นหรือฟังก์ชันไว้ใช้งานโดยเฉพาะ โดยใช้ MATLAB Editor เขียนในรูปฟังก์ชัน m-file พร้อมเครื่องมือตรวจสอบดีบัคโปรแกรม

3. ไลบรารีฟังก์ชันและบล็อกไดอะแกรม MATLAB จัดสร้างไว้ครอบคลุมหลายสาขาวิชา และรวบรวมฟังก์ชัน m-file หรือ mdl ของ Simulink เป็นไฟล์ย่อยๆ ไว้ โดยแต่ละไฟล์จะเป็นไฟล์ที่สร้างขึ้นมาเพื่อใช้กำหนดลักษณะในการ

คำนวณอัลกอลิทึม (Algorithms) แบบต่างๆ เริ่มจากโอเปอร์เรเตอร์ การบวกลบคูณหาร ฟังก์ชันตรีโกณมิติพื้นฐาน ไปจนถึงฟังก์ชันที่มีความซับซ้อนมีขั้นตอนในการคำนวณมาก ๆ เช่นการหาอินเวิร์สของเมตริกซ์ การหาค่าสมการเชิงซ้อน หรือ การหารูทโลคัส เป็นต้น โดยส่วนนี้ประกอบด้วยฟังก์ชันพื้นฐาน เช่น บวก ลบ คูณ หาร sine, cost, log, x^2 และฟังก์ชันเฉพาะสาขาวิชา ซึ่งเรียกว่าทูลบ็อกซ์ (Toolbox) เช่น Control System, Bioinformatics ,Signal Processing, Fuzzy Logic, Aerospace, Image Processing, Econometrics และอื่นๆ

4. Handle Graphics เป็นไลบรารีฟังก์ชันใน MATLAB สำหรับแสดงผลข้อมูล เป็นกราฟิก รูปภาพ เสียง วิดีโอ พร้อมด้วย ไลบรารี Guide เป็นเครื่องมือสำหรับสร้างแอพพลิเคชั่นแบบ GUI (Graphic User Interface) โดยเลียนแบบการเขียนโปรแกรมจาก Visual Basic ทำให้สามารถนำอัลกอไปพัฒนาแอพพลิเคชั่นบน MATLAB เพื่อใช้งานได้อย่างรวดเร็ว

5. MATLAB API (API: Application Program Interface) เป็นส่วนติดต่อกับภาษาคอมพิวเตอร์อื่นๆ โปรแกรมภายนอก นามสกุลไฟล์ต่างๆ และสามารถพัฒนาให้เชื่อมต่อกับฮาร์ดแวร์ได้ง่าย

2.1.2 คำสั่งของ MATLAB [1]

คำสั่งของ MATLAB แบ่งออกได้เป็น 9 ด้านคือ

1. การคำนวณเชิงตัวเลข (Numerical Computing)

การใช้คำนวณพื้นฐานโดยทั่วไป

- ตัวแปรต่างๆ และรูปแบบการสร้างในเวิร์กสเปซ สำหรับใช้ในการคำนวณ
- การกระทำทางคณิตศาสตร์ (Operation) เช่น การบวก ลบ คูณ หาร ยกกำลังสอง หารสอง และอื่นๆ
- การจัดเก็บค่าตัวแปรเป็นไฟล์ และการโหลดไฟล์ที่จัดเก็บไว้เข้ามาใช้งาน
- การย้อนทำในคำสั่งเดิมๆ

2. การคำนวณพีชคณิต
   - หลักสมการพีชคณิตเบื้องต้น
   - วิธีกำหนดค่าตัวแปร MATLAB ในสมการ
3. การคำนวณค่าอนุพันธ์และอินทิกัล- หลักการหาค่าอนุพันธ์โดยสรุป - วิธีคำนวณค่าอนุพันธ์โดย MATLAB
   - หลักการหาค่าอินทิกัล
   - วิธีคำนวณค่าอินทิกัลโดย MATLAB
4. การคำนวณตรีโกณมิติ
   - ฟังก์ชัน MATLAB สำหรับการคำนวณทางตรีโกณมิติ
   - ตัวอย่างการใช้งาน
5. การคำนวณด้านเมตริกซ์
   - เวกเตอร์ (วิธีสร้างตัวแปรเวกเตอร์หลักและแถวใน MATLAB และการใช้งานเวกเตอร์ทั่วไป)
   - เมตริกซ์ (การสร้างตัวแปรเมตริกซ์ต่างๆ และการคำนวณเมตริกซ์ในแบบต่างๆ)
6. การคำนวณเฉพาะในแต่ละสาขาวิชา หรือที่เรียกว่าการคำนวณเชิงเทคนิค (Technical Computing)
   - การหาค่าออฟติไมต์ระบบ มีการนำไปใช้งานหลายสาขาทางวิศวกรรม

7. การเขียนโปรแกรม เพื่อสร้างแอพพลิเคชั่น

- การเขียนสคริปต์
- การเขียนฟังก์ชัน

8. การพล็อตภาพ

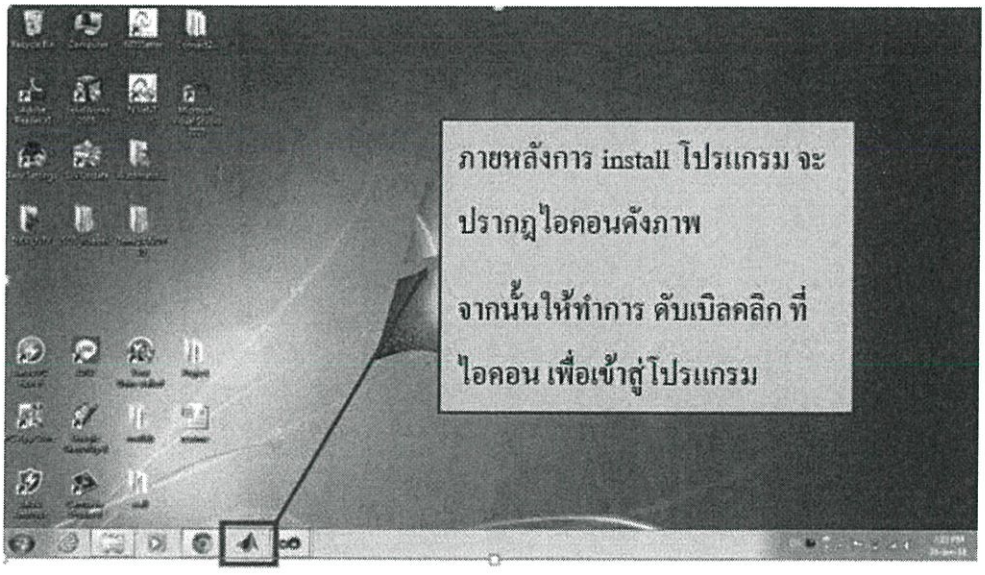
- การพล็อตเบื้องต้น
- การพล็อตภาพสองมิติ (2D)
- การพล็อตภาพสามมิติ (3D)

9. การแสดงผลไฟล์อื่นๆ และการรับข้อมูลจากอุปกรณ์ภายนอก

- การแสดงผลจากไฟล์อื่นๆ เช่นไฟล์ โน้ตแพด (Notepad) เอ็กซ์เซล (Excel) ไฟล์รูปภาพ ไฟล์เสียง ไฟล์วีดีโอ
- การรับข้อมูลจากอุปกรณ์ต่างๆ ทั้งอุปกรณ์พื้นฐาน เช่น กล้องเว็บแคม ไมโครโฟน และอุปกรณ์ของบริษัทพันธมิตรกับ MATLAB

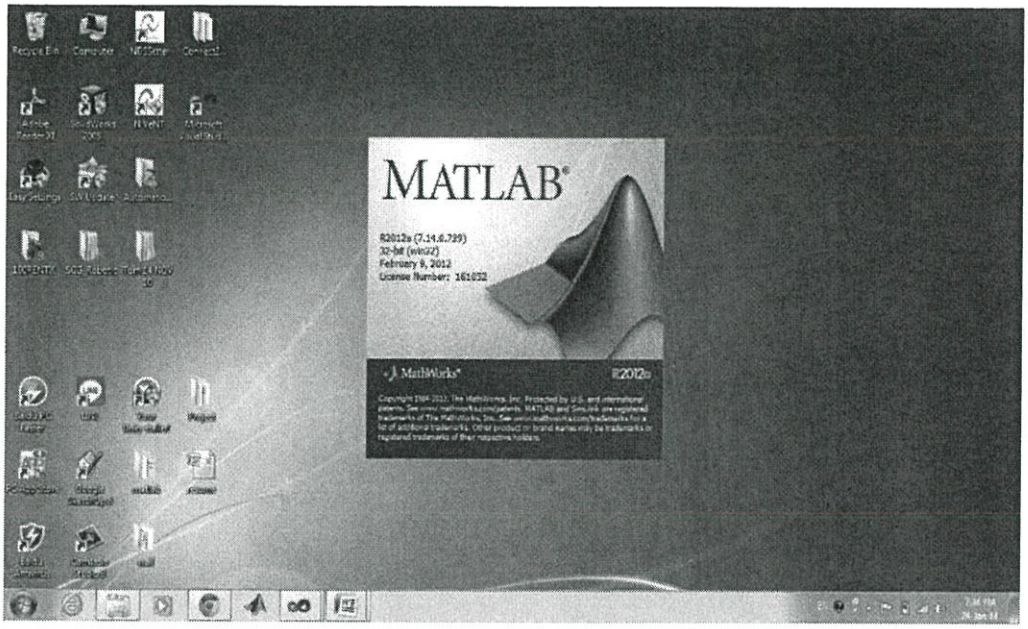
2.1.3 เริ่มต้นใช้งานโปรแกรม MATLAB

1. ภายหลังการติดตั้งโปรแกรมลงบนคอมพิวเตอร์จะปรากฏไอคอนของโปรแกรม MATLAB ดังแสดงในรูปที่ 2.2



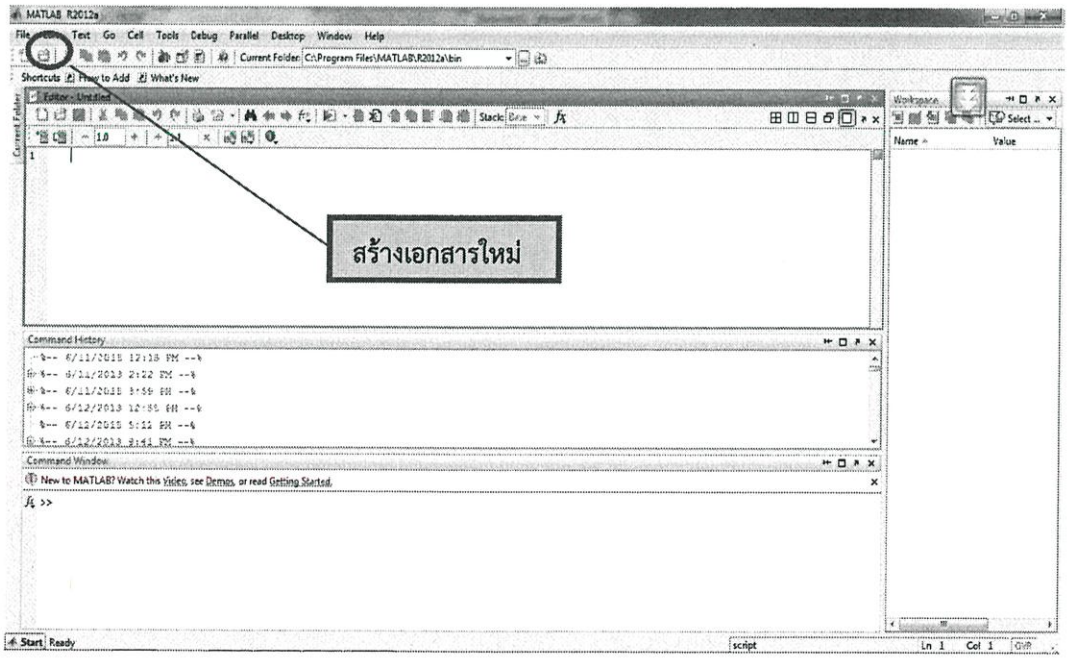
รูปที่ 2.2 วิธีการเข้าสู่โปรแกรม 1

2. หลังจากทำการ ดับเบิลคลิก ที่ไอคอนแล้ว จะปรากฏหน้าจอดังแสดงในรูปที่ เพื่อทำการ Boot เข้าสู่โปรแกรม เวลาในการ Boot เข้าสู่โปรแกรม อาจใช้เวลาชั่วครู่ ดังแสดงในรูปที่ 2.3



รูปที่ 2.3 วิธีการเข้าสู่โปรแกรม 2

3. หลังจากทำการ Boot โปรแกรมเรียบร้อยแล้วจะปรากฏหน้าจอดังแสดงในรูปที่ ให้ทำการสร้างเอกสารใหม่ จะปรากฏหน้าจอดังแสดงในรูปที่ 2.4



รูปที่ 2.4 หน้าต่างการทำงานของ MATLAB

เมื่อเข้าสู่โปรแกรม MATLAB จะปรากฏหน้าต่างดังรูป ซึ่งมีหน้าต่างย่อยคือ Workspace, Editor, Command History และ Command Windows

- Workspace หน้าต่างสำหรับแสดงตัวแปรที่กำลังใช้งานอยู่- Editor หน้าต่างสำหรับป้อนชุดคำสั่ง

– Command Windows หน้าต่างสำหรับแสดงผลลัพธ์- Command History หน้าต่างแสดงชุดคำสั่งที่ใช้งานไปแล้วทั้งหมดในอดีต

การใช้งานโปรแกรม MATLAB ทำโดยการป้อนชุดคำสั่งใน Command Windows เมื่อต้องการความช่วยเหลือให้พิมพ์ help [ชื่อคำสั่ง] เพื่อดูวิธีการใช้งานคำสั่งต่างๆ

2.1.4 การกำหนดชื่อตัวแปร [1]

ตัวแปรใน MATLAB ต้องเริ่มต้นชื่อด้วยตัวอักษรเสมอ อาจตามด้วยตัวอักษร ตัวเลข หรือเครื่องหมายขีดเส้นใต้ (_) ชื่อตัวแปรกำหนดให้มีความยาวไม่เกิน 63 ตัวอักษร ตัวอย่างการเลือกชื่อตัวแปรที่ถูกต้อง เช่น index, year, growth_rate, a1, a2 เป็นต้น และตัวอย่างชื่อตัวแปรที่ผิด เช่น growth rate (ห้ามใช้เว้นวรรค), 2r (ห้ามขึ้นต้นด้วยตัวเลข)

2.1.5 เมตริกซ์ [3]

เมตริกซ์ หรือ เมทริกซ์ (Matrix) คือ ตารางสี่เหลี่ยมที่แต่ละช่องบรรจุจำนวนหรือโครงสร้างทางคณิตศาสตร์ที่สามารถนำมาบวกและคูณกับตัวเลขได้ สามารถใช้เมตริกซ์แทนระบบสมการเชิงเส้น การแปลงเชิงเส้น และใช้เก็บข้อมูลที่ขึ้นกับตัวแปรต้นสองตัว เราสามารถบวก คูณ และแยกเมตริกซ์ออกเป็นผลคูณของเมตริกซ์ได้หลายรูปแบบ เมตริกซ์เป็นแนวความคิดที่มีความสำคัญยิ่งของพีชคณิตเชิงเส้น โดยทฤษฎีเมตริกซ์เป็นสาขาหนึ่งของพีชคณิตเชิงเส้นที่เน้นการศึกษาเมตริกซ์ ซึ่งจะนำเสนอทฤษฎีต่างๆที่ใช้สำหรับการเขียนโปรแกรม ดังต่อไปนี้

<u>การบวกเมตริกซ์</u>

บทนิยาม ให้ $A = [a_{ij}]_{m \times n}$ และ $B = [b_{ij}]_{m \times n}$ จะได้ว่า $A + B = [c_{ij}]_{m \times n}$ โดยที่ $c_{ij} = a_{ij} + b_{ij}$

จากบทนิยามข้างต้น สังเกตว่า การนำเมตริกซ์ 2 เมตริกซ์มาบวกกัน จะเกิดขึ้นได้ต้องมีเงื่อนไข 2 ประการ คือ

- เมตริกซ์ A และ B ต้องมีมิติเท่ากัน

- สมาชิกของผลลัพธ์นั้น เกิดจากการนำสมาชิกในเมตริกซ์ A และสมาชิกในเมตริกซ์ B มาบวกกัน แต่ต้องเป็นสมาชิกที่อยู่ในตำแหน่งเดียวกันทั้งหมด

ตัวอย่าง ให้ $A = \begin{bmatrix} -1 & 0 & 2 \\ 4 & 2 & 6 \end{bmatrix}$, $B = \begin{bmatrix} 7 & -2 & 5 \\ 2 & 1 & 3 \end{bmatrix}$

$$A + B = \begin{bmatrix} -1+7 & 0+(-2) & 2+5 \\ 4+2 & 2+1 & 6+3 \end{bmatrix} = \begin{bmatrix} 6 & -2 & 7 \\ 6 & 3 & 9 \end{bmatrix}$$

<u>การลบเมตริกซ์</u>

บทนิยาม ให้ $A = [a_{ij}]_{m \times n}$ และ $B = [b_{ij}]_{m \times n}$ จะได้ว่า $A - B = [c_{ij}]_{m \times n}$ โดยที่ $c_{ij} = a_{ij} - b_{ij}$

จากบทนิยามข้างต้น สังเกตว่า การนำเมตริกซ์ 2 เมตริกซ์มาลบกัน จะเกิดขึ้นได้ต้องมีเงื่อนไข 2 ประการ คือ

- เมตริกซ์ที่จะนำมาลบกันต้องมีมิติเท่ากัน

- นำสมาชิกที่อยู่ตำแหน่งเดียวกันมาลบกัน

ตัวอย่าง ให้ $A = \begin{bmatrix} 1 & 5 & 7 \\ 2 & 9 & 6 \end{bmatrix}$, $B = \begin{bmatrix} 0 & 1 & 3 \\ 6 & 2 & 4 \end{bmatrix}$

$$A - B = \begin{bmatrix} 1-0 & 5-1 & 7-3 \\ 2-6 & 9-2 & 6-4 \end{bmatrix} = \begin{bmatrix} 1 & 4 & 4 \\ -4 & 7 & 2 \end{bmatrix}$$

การคูณเมตริกซ์ด้วยจำนวนจริง

บทนิยาม ให้ $A = [a_{ij}]_{m \times n}$ และ k เป็นจำนวนจริงใดๆ จะได้ว่า $kA = [ka_{ij}]_{m \times n}$

จากบทนิยามข้างต้น สังเกตว่า การนำจำนวนจริงใดๆคูณกับเมตริกซ์ สามารถนำเข้าไปคูณกับสมาชิกแต่ละหลักได้เลย

การคูณเมตริกซ์ด้วยเมตริกซ์

ถ้า A และ B เป็นเมตริกซ์ 2 เมตริกซ์ใด ๆ การนำเมตริกซ์ A มาคูณกับเมตริกซ์ B จะเกิดผลขึ้นกรณีใดกรณีหนึ่งใน 2 กรณีต่อไปนี้

- ไม่สามารถหาผลคูณได้ เนื่องจากมิติของเมตริกซ์ A และเมตริกซ์ B มีขนาดไม่เท่ากัน
- สามารถหาผลคูณได้ โดยมีเงื่อนไขต่อไปนี้

1. มิติของเมตริกซ์ที่นำมาหาผลคูณ ถ้า A เป็นเมตริกซ์ m x p และ B เป็นเมตริกซ์ q x n ผลคูณ AB จะเกิดขึ้นได้เมื่อ p = q และ AB จะมีมิติ m x n

2. ลักษณะสมาชิกของเมตริกซ์ที่เป็นผลคูณ หลักการหาสมาชิกโดยทั่วๆ ไป สามารถหาได้ดังนี้ "สมาชิกของผลคูณของเมตริกซ์ในแถวที่ i หลักที่ j จะเกิดสมาชิกในแถวที่ i ของเมตริกซ์ที่อยู่หน้า คูณกับ สมาชิกในหลักที่ j ของเมตริกซ์หลังเป็นคู่ ๆ แล้วนำมาบวกกัน"

ตัวอย่าง ให้ $A = \begin{bmatrix} 1 & 2 \\ -1 & 0 \\ 3 & 2 \end{bmatrix}$, $B = \begin{bmatrix} 1 & 5 & 2 \\ -2 & 0 & 1 \end{bmatrix}$

$$AB = \begin{bmatrix} 1 & 2 \\ -1 & 0 \\ 3 & 2 \end{bmatrix} \begin{bmatrix} 1 & 5 & 2 \\ -2 & 0 & 1 \end{bmatrix}$$

ใช้หลักการ แถวคูณหลัก

$$= \begin{bmatrix} (1)(1)+(2)(-2) & (1)(5)+(2)(0) & (1)(2)+(2)(1) \\ (-1)(1)+(0)(-2) & (-1)(5)+(0)(0) & (-1)(2)+(0)(1) \\ (3)(1)+(2)(-2) & (3)(5)+(2)(0) & (3)(2)+(2)(1) \end{bmatrix}$$

$$= \begin{bmatrix} -3 & 5 & 4 \\ -1 & -5 & -2 \\ -1 & 15 & 8 \end{bmatrix}$$

การทรานสโพสของเมตริกซ์ (Transpose of Matrix)

ทรานสโพสของเมตริกซ์ A คือ เมตริกซ์ที่เกิดจากการเอาสมาชิกทั้งหมดใน แถวที่ 1 ของเมตริกซ์ A มาเขียนเป็นสมาชิกในหลักที่ 1 และเอาสมาชิกทั้งหมดในแถวที่ 2 ของเมตริกซ์ A มาเขียนเป็นสมาชิกในหลักที่ 2 และทำเช่นนี้ไปเรื่อยๆจนหมด ทรานสโพสของเมตริกซ์ A เขียนแทนด้วย A^t (หรือ A^T หรือ A')

บทนิยาม ให้ A = $[a_{ij}]_{m \times n}$ แล้ว ทรานสโพสของเมตริกซ์ A จะได้ว่า A' = $[a_{ji}]_{n \times m}$

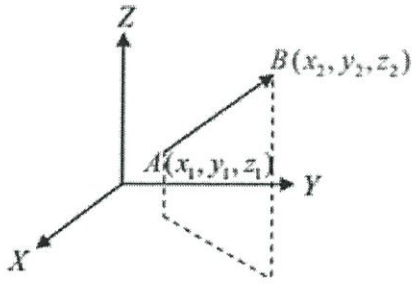
ตัวอย่าง $\begin{bmatrix} 1 & 2 & 3 \\ 6 & 5 & 7 \end{bmatrix}^t = \begin{bmatrix} 1 & 6 \\ 2 & 5 \\ 3 & 7 \end{bmatrix}$

2.1.6 เวกเตอร์สามมิติ

เวกเตอร์ คือ ปริมาณอย่างหนึ่งที่มีทั้งขนาดและทิศทาง ซึ่งเวกเตอร์สามมิติจะประกอบไปด้วย 3 แกน คือ แกน X แกน Y และแกน Z บ่งบอกถึงตำแหน่งต่างๆในรูปแบบสามมิติได้ โดยหลักการของเวกเตอร์สามมิติ ประกอบไปด้วย การเขียนเวกเตอร์ในระบบพิกัดฉาก, การหาขนาดของเวกเตอร์, นิเสธของเวกเตอร์ ($\vec{u}$ & $-\vec{u}$) และเวกเตอร์ศูนย์ ($\vec{0}$), การบวกและการลบเวกเตอร์, การคูณหรือการหารเวกเตอร์, เวกเตอร์ 1 หน่วย เป็นต้น

<u>การเขียนเวกเตอร์ในระบบพิกัดฉาก [4]</u>

การเขียนเวกเตอร์ในระบบพิกัดฉาก สามารถแสดงได้ดังรูปที่ 2.5



รูปที่ 2.5 เวกเตอร์ในระบบพิกัดฉาก

$\overrightarrow{AB} = (x_2 - x_1)\vec{i} + (y_2 - y_1)\vec{j} + (z_2 - z_1)\vec{k}$ เขียนสัญลักษณ์อีกแบบได้เป็น

$$\overrightarrow{AB} = \begin{bmatrix} x_2 - x_1 \\ y_2 - y_1 \\ z_2 - z_1 \end{bmatrix}$$

<u>การหาขนาดของเวกเตอร์ [5]</u>

ให้ $\vec{u} = a\vec{i} + b\vec{j} + c\vec{k}$

จะได้ว่า $|\vec{u}| = \sqrt{a^2 + b^2 + c^2}$

<u>นิเสธของเวกเตอร์ ($\vec{u}$ & $-\vec{u}$) และเวกเตอร์ศูนย์ ($\vec{0}$)</u> [5]

ให้ $\vec{u} = a\vec{i} + b\vec{j} + c\vec{k}$

นิเสธของ $\vec{u}$ คือ $-\vec{u} = -a\vec{i} - b\vec{j} - c\vec{k}$

เวกเตอร์ศูนย์ ($\vec{0}$) $\vec{0} = 0\vec{i} + 0\vec{j} + 0\vec{k}$

<u>การบวกและลบเวกเตอร์</u> [5]

ให้ $\vec{u}$ = a$\vec{i}$ + b$\vec{j}$ + c$\vec{k}$ และ $\vec{v}$ = d$\vec{i}$ + e$\vec{j}$ + f$\vec{k}$

จะได้ว่า $\vec{u} + \vec{v}$ = (a + d) $\vec{i}$ + (b + e) $\vec{j}$ + (c + f) $\vec{k}$

$\vec{u} - \vec{v}$ = (a - d) $\vec{i}$ + (b - e) $\vec{j}$ + (c - f) $\vec{k}$

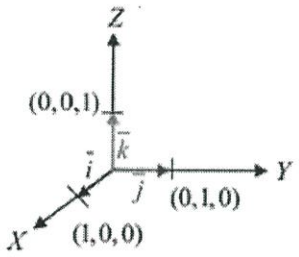
<u>การคูณเวกเตอร์ด้วยสเกลาร์ (k * $\vec{u}$)</u> [5]

ให้ $\vec{u}$ = a$\vec{i}$ + b$\vec{j}$ + c$\vec{k}$

จะได้ว่า k$\vec{u}$ = ka$\vec{i}$ + kb$\vec{j}$ + kc$\vec{k}$

<u>เวกเตอร์ 1 หน่วย (Unit Vector)</u> [5]

การขียนเวกเตอร์ 1 หน่วย เขียนได้ ดังแสดงในรูปที่ 2.6



รูปที่ 2.6 เวกเตอร์ 1 หน่วย

i เป็นเวกเตอร์ 1 หน่วย ที่มีทิศทางตามแกน X

j เป็นเวกเตอร์ 1 หน่วย ที่มีทิศทางตามแกน Y

k เป็นเวกเตอร์ 1 หน่วย ที่มีทิศทางตามแกน Z

ให้ $\vec{u}$ = a$\vec{i}$ + b$\vec{j}$ + c$\vec{k}$ จะได้ว่า $\frac{\vec{u}}{|\vec{u}|} = \frac{a\vec{i} + b\vec{j} + c\vec{k}}{\sqrt{a^2+b^2+c^2}}$ เป็นเวกเตอร์ 1 หน่วย ของ $\vec{u}$ คือมีขนาด 1 หน่วย และมีทิศทางเดียวกับ $\vec{u}$

<u>ผลคูณเชิงสเกลาร์ (Scalar Product หรือ Dot Product)</u> [5]

ผลคูณเชิงสเกลาร์ คือ ผลคูณของเวกเตอร์ที่ได้ค่าออกมาเป็นสเกลาร์ ($\vec{u} \cdot \vec{v}$) ให้ $\vec{u}$ = a$\vec{i}$ + b$\vec{j}$ + c$\vec{k}$ และ $\vec{v}$ = d$\vec{i}$ + e$\vec{j}$ + f$\vec{k}$ จะได้ว่า $\vec{u} \cdot \vec{v}$ = (a · d) + (b · e) + (c · f)

ผลคูณเชิงเวกเตอร์ (Vector Product หรือ Cross Product) [5]

ผลคูณเชิงเวกเตอร์ คือ ผลคูณของเวกเตอร์ซึ่งได้ค่าออกมาเป็นเวกเตอร์

ให้ $\vec{u} = a\vec{i} + b\vec{j} + c\vec{k}$ และ $\vec{v} = d\vec{i} + e\vec{j} + f\vec{k}$ จะได้ว่า

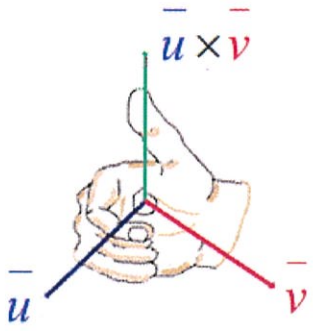
$$\bar{u} \times \bar{v} = \begin{vmatrix} \bar{i} & \bar{j} & \bar{k} \\ a & b & c \\ d & e & f \end{vmatrix} = \begin{vmatrix} \bar{i} & \bar{j} & \bar{k} & \bar{i} & \bar{j} \\ a & b & c & a & b \\ d & e & f & d & e \end{vmatrix}$$

$$-(dbk + eci + faj) = -dbk - eci - faj$$

$$bfi + cdj + aek$$

$$\vec{u} \times \vec{v} = (bf - ec)\,\vec{i} + (cd - fa)\,\vec{j} + (ae - db)\,\vec{k}$$

ทิศทางของผลคูณเชิงเวกเตอร์ หาได้โดยใช้ กฎมือขวา ดังแสดงในรูปที่ 2.7



รูปที่ 2.7 ทิศทางของผลคูณเชิงเวกเตอร์โดยใช้กฎมือขวา [6]

2.1.7 ตัวดำเนินการทางคณิตศาสตร์ใน MATLAB [1]

+, -, *, /	ใช้สำหรับการบวก ลบ คูณ หาร
.*, ./	ใช้สำหรับการคูณ และการหารเมตริกซ์ และเวกเตอร์ แบบ Element by Element
‘	ใช้สำหรับการ Transpose Matrix
^	ใช้สำหรับการยกกำลัง
:	ใช้สำหรับการกำหนดช่วงตัวเลข, สร้างเวกเตอร์
.	จุดทศนิยม
( )	ใช้สำหรับกำหนด Subscripts
=	ใช้สำหรับกำหนดค่า
[ ]	ใช้สำหรับสร้างเวกเตอร์แบะเมตริกซ์
...	ใช้สำหรับกระทำคำสั่งยังบรรทัดต่อไป
;	ใช้สำหรับการขึ้นบรรทัดใหม่ หรือ ปิดท้ายสมาการ
,	ใช้สำหรับแยก Element ภายในเมตริกซ์ และ Subscripts

2.1.8 ฟังก์ชันคณิตศาสตร์พื้นฐานและค่าตัวแปรเฉพาะ MATLAB

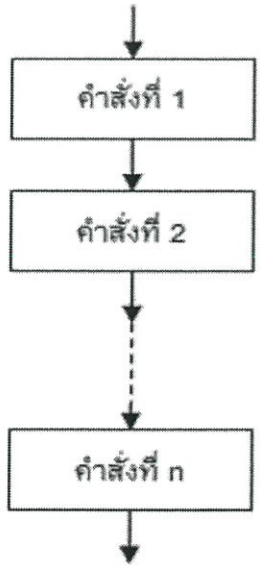
Tic/Toc	ใช้สำหรับการจับเวลาโปรแกรมตั้งแต่เริ่มต้นจบกระทั่งสิ้นสุด
Ans	ใช้สำหรับการเก็บผลจากการคำนวณค่าปัจจุบันใดๆ ที่ไม่ได้ทำการกำหนดชื่อตัวแปรของผลลัพธ์
zeros (n, m)	ใช้สำหรับการสร้างเมตริกซ์ศูนย์ขนาด nxm
ones (n, m)	ใช้สำหรับการสร้างเมตริกซ์หนึ่งขนาด nxm
Nan	ย่อมาจาก not a number คือ ค่าคงที่ที่ไม่ได้มีการกำหนดอาจเกิดจากการหารระหว่างค่าศูนย์กับค่าศูนย์ (0/0)

2.1.9 ฟังก์ชันการสร้างแบบมีเงื่อนไข MATLAB [7]

ก่อนการเขียนโปรแกรม ผู้พัฒนาโปรแกรมจะต้องเลือกภาษาคอมพิวเตอร์ที่จะนำมาใช้ช่วยงานโดยพิจารณาจากปัจจัยต่างๆ ในการทำงาน เช่น ลักษณะของปัญหา ความถนัดของผู้เขียนโปรแกรม สภาพแวดล้อมในการทำงานของระบบคอมพิวเตอร์ เป็นต้น เนื่องจากในปัจจุบันมีภาษาคอมพิวเตอร์ให้เลือกใช้ได้หลายภาษา เช่น ภาษาปาสคาล ภาษาซี ภาษาจาวา ภาษาเดลฟาย เป็นต้น ถึงแม้แต่ละภาษาจะมีรูปแบบและหลักการในการสร้างงานที่แตกต่างกัน แต่ทุกภาษาจะต้องมีโครงสร้างควบคุมหลักทั้ง 3 แบบ ได้แก่ โครงสร้างแบบลำดับ (Sequential Structure) โครงสร้างแบบมี

1. โครงสร้างแบบลำดับ (Sequential structure) [7]

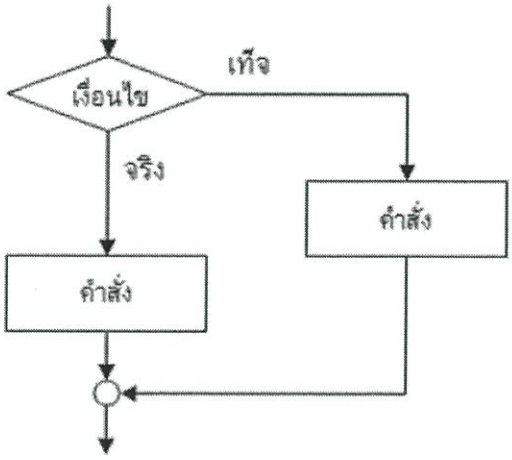
โครงสร้างแบบลำดับ คือ โครงสร้างแสดงขั้นตอนการทำงานที่เป็นไปตามลำดับก่อนหลัง และแต่ละขั้นตอนจะถูกประมวลผลเพียงครั้งเดียวท่านั้น สามารถแสดงการทำงานของโครงสร้างนี้ โดยใช้ผังงานได้ดังแสดงในรูปที่ 2.8



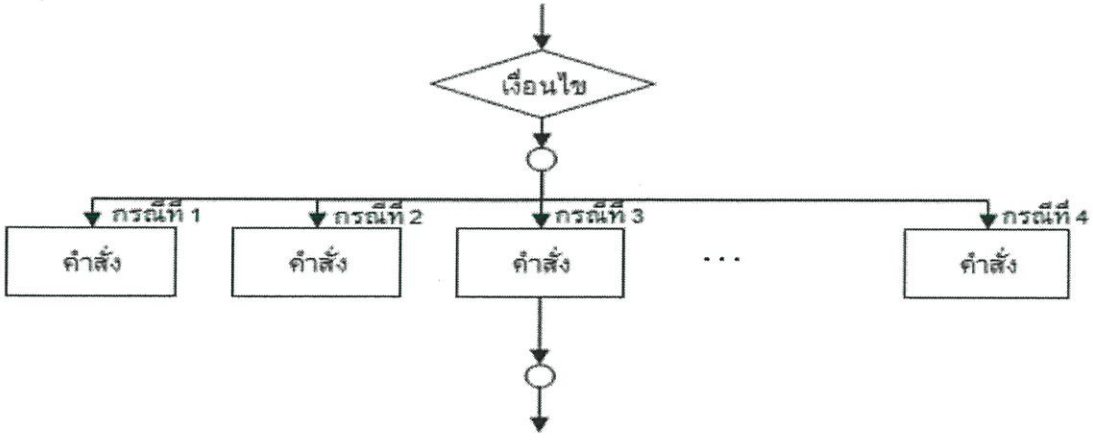
รูปที่ 2.8 โครงสร้างแบบลำดับ [7]

2. โครงสร้างแบบมีทางเลือก (Selection Structure) [7]

คือ โครงสร้างที่มีเงื่อนไข ขั้นตอนการทำงานบางขั้นตอนต้องมีการตัดสินใจเพื่อเลือกวิธีการประมวลผลขั้นต่อไป และจะมีบางขั้นตอนที่ไม่ได้รับการประมวลผล การตัดสินใจอาจมีทางเลือก 2 ทางหรือมากกว่าได้ โครงสร้างที่มีทางเลือกเพียง 2 ทาง เรียกชื่อว่าโครงสร้างแบบ if...then...else และโครงสร้างที่มีทางเลือกมากกว่า 2 ทาง เรียกชื่อว่าโครงสร้างแบบ Case ซึ่งสามารถแสดงการทำงานของโครงสร้างนี้โดยใช้ผังงานได้ดังแสดงในรูปที่ 2.9 และ รูปที่ 2.10



รูปที่ 2.9 โครงสร้างแบบมีทางเลือก if...then...else [7]



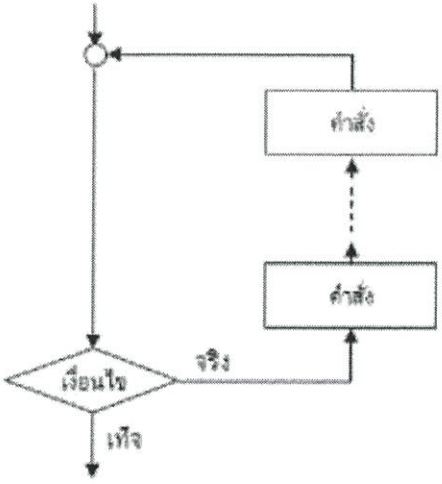
รูปที่ 2.10 โครงสร้างแบบมีทางเลือก (Case) [7]

3. โครงสร้างแบบทำซ้ำ (Repetition Structure) [7]

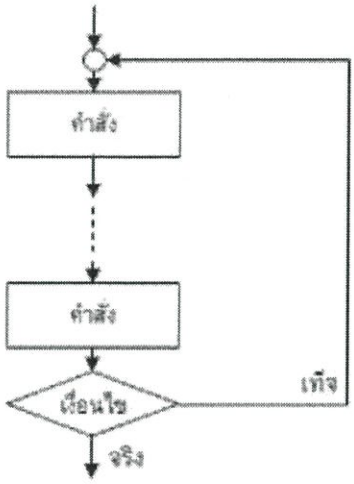
คือ โครงสร้างที่ขั้นตอนการทำงานบางขั้นตอนได้รับการประมวลผลมากกว่า 1 ครั้ง ทั้งนี้ขึ้นอยู่กับเงื่อนไขบางประการ โครงสร้างแบบทำซ้ำนี้ต้องมีการตัดสินใจในการทำงานซ้ำ และลักษณะการทำงานของโครงสร้างแบบนี้มี 2 ลักษณะ ได้แก่

- แบบที่มีการตรวจสอบเงื่อนไขในการทำซ้ำทุกครั้งก่อนดำเนินการกิจกรรมใดๆ ถ้าเงื่อนไขเป็นจริงจะทำงานซ้ำไปเรื่อยๆ และหยุดเมื่อเงื่อนไขเป็นเท็จ เรียกการทำงานลักษณะนี้ว่า การทำซ้ำแบบ do while
- แบบที่ทำกิจกรรมซ้ำเรื่อยๆ จนกระทั่งเงื่อนไขที่กำหนดเป็นจริงแล้วจึงหยุดการทำงาน โดยแต่ละครั้งที่เสร็จสิ้นการดำเนินการแต่ละรอบจะต้องมีการตรวจสอบเงื่อนไข เรียกการทำซ้ำลักษณะนี้ว่า การทำซ้ำแบบ do until

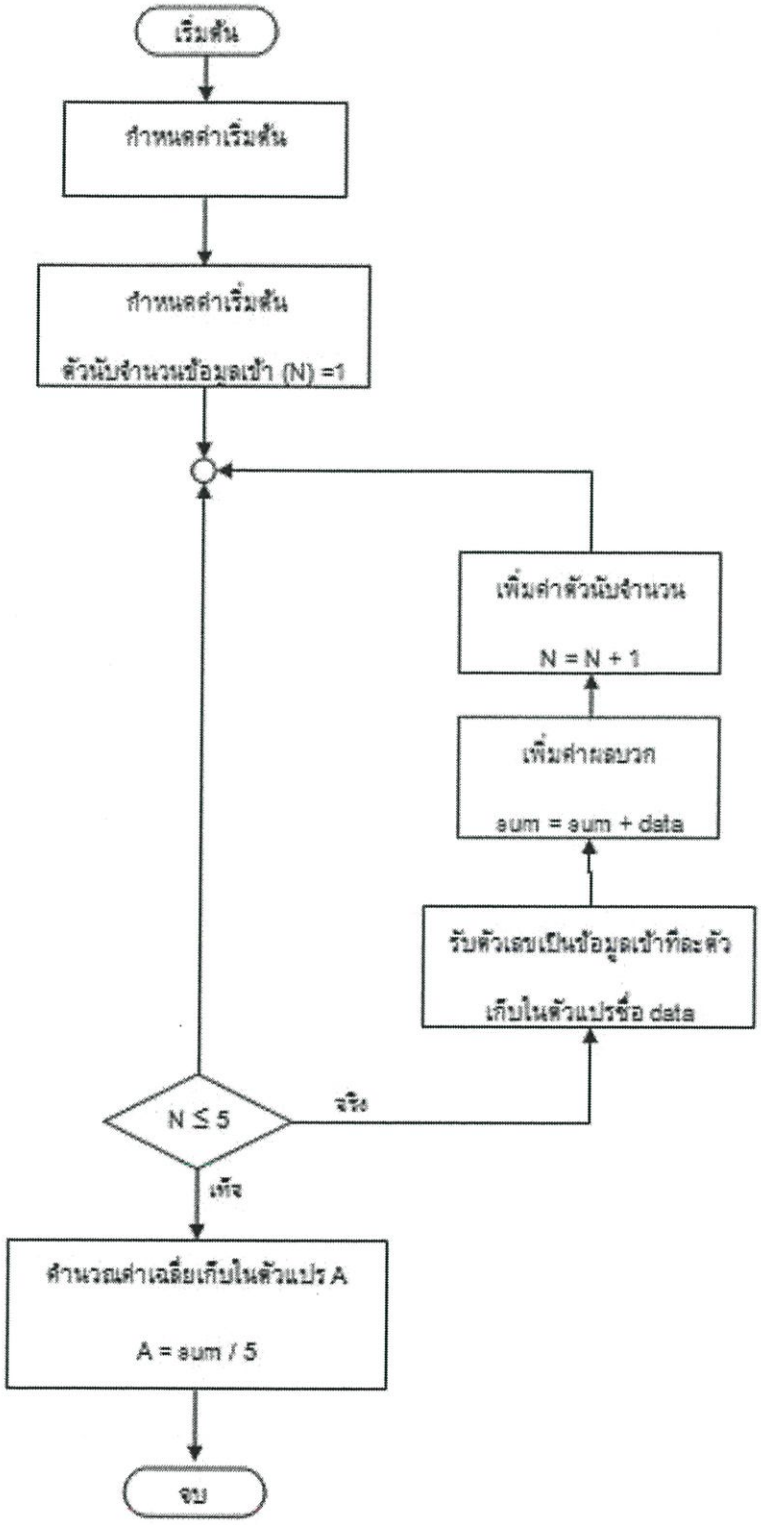
ผังงานแสดงขั้นตอนการทำงานของโครงสร้างแบบทำซ้ำทั้งสองแบบ ดังแสดงในรูปที่ 2.11 รูปที่ 2.12และรูปที่ 2.13



รูปที่ 2.11 การทำงานแบบทำซ้ำของ do while [7]



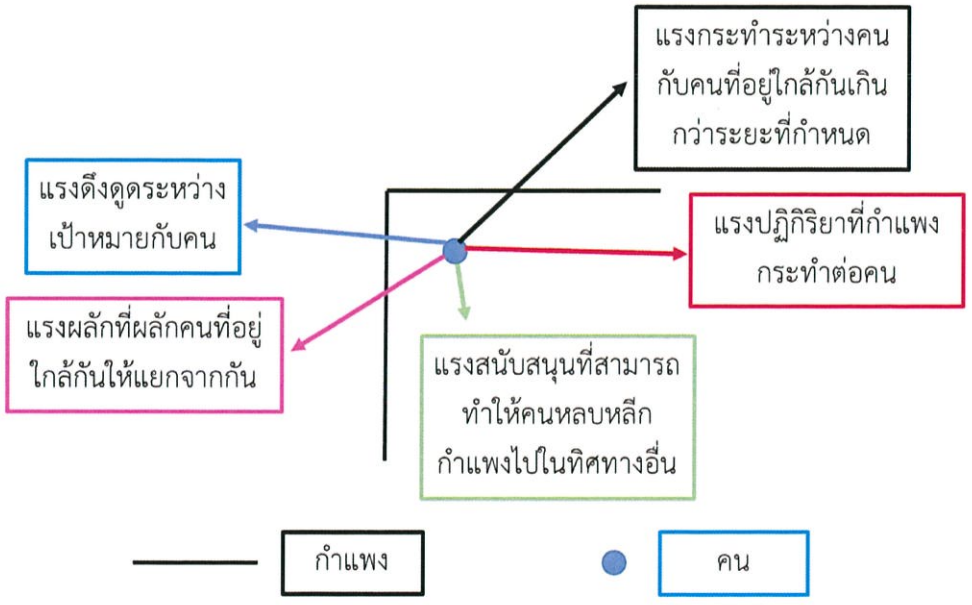
รูปที่ 2.12 การทำงานแบบทำซ้ำของ do until [7]



รูปที่ 2.13 ตัวอย่างแผนผังโครงสร้าง [7]

2.2 การเคลื่อนที่ของวัตถุ [8]

การเคลื่อนที่ของวัตถุเป็นการเคลื่อนที่แบบจลน์ (Dynamics) โดยเป็นไปตามกฎการเคลื่อนที่ของนิวตัน (Newton's Low of Motion) ที่วัตถุจะเคลื่อนที่ไปจากการที่มีแรงมากระทำกับวัตถุนั้นๆ โดยในโครงงานนี้ได้สมมติให้ใช้วงกลมหนึ่งวงแทนคนหนึ่งคน โดยคนที่จะเดินไปหาจุดเป้าหมาย (ประตูทางออก) จากการสมมติ "แรงดึงดูด" (Attractive Force) ระหว่างเป้าหมายกับคน ซึ่งจะทำให้คนเริ่มเคลื่อนที่เข้าหาเป้าหมาย แต่อุปสรรคที่กีดขวางอย่างกำแพง เครื่องจักร ประตู สิ่งของต่างๆ รวมทั้งคนอื่นที่สมมติขึ้นด้วยนั้น จะออกแรงผลักให้คนที่พิจารณาอยู่นั้นเคลื่อนที่เบี่ยงเบนออกไปไม่ให้ชนกำแพง หรืออุปสรรคอื่นๆ "แรงผลัก" (Repulsive Force) เหล่านี้เองจะเป็นตัวทำให้การเคลื่อนที่เป็นไปในรูปแบบสุ่ม ขึ้นกับตำแหน่งของคน อุปสรรค และเป้าหมายในเวลานั้นๆ และแรงผลักจะมีผลต่อคนก็ต่อเมื่อคนๆ นั้นเดินเข้าใกล้อุปสรรคในระยะที่กำหนด (โดยในโครงงานนี้ใช้ระยะ 30 ซม.) แรงผลัก สามารถแบ่งได้เป็นแรงผลักจากสิ่งแวดล้อมคงตัว อย่างผนัง กำแพง เครื่องจักร และแรงผลักจากสิ่งแวดล้อมที่เคลื่อนไหวได้ อย่างคนอื่นๆ โดยในโครงงานนี้ได้สมมติให้ ภายใต้แบบจำลองมีคนเดิน 50 คนในระบบดังกล่าวในหัวข้อต่อไป ดังแสดงในรูปที่ 2.14



รูปที่ 2.14 แรงที่เกิดขึ้นในระบบ

โดยการเคลื่อนที่ทั้งหมดที่กล่าวมาได้ทำอยู่ในรูปของโปรแกรมของ MATLAB ที่ใช้เมตริกซ์ (Matrix) เวคเตอร์ (Vector) และการจำลองการเคลื่อนที่แบบจลน์ที่ใช้วิธีประมวลผลทางตัวเลขในคอมพิวเตอร์ (Computer Numerical) ในการคำนวณแผนการและเส้นทางเดินของคนทั้ง 50 คน ที่สมมติขึ้นในแบบจำลอง เส้นแรงที่เกิดขึ้นจะอธิบายในบทที่ 4

2.2.1 การเคลื่อนที่แบบจลน์ [8]

พลศาสตร์ (Dynamics) เป็นการศึกษาวัตถุที่เกิดการเคลื่อนที่ภายใต้ความเร่ง โดยส่วนใหญ่แบ่งเป็น 2 ส่วน คือ พลศาสตร์แบบเชิงเส้น และพลศาสตร์แบบเชิงมุม พลศาสตร์แบบเชิงเส้น เป็นการศึกษาการเคลื่อนที่ของวัตถุในแนวเส้นตรง รวมถึงสมบัติด้านแรง มวล ความเฉื่อย (Inertia) การกระจัด (Displacement) ความเร็ว (Velocity) ความเร่ง (Acceleration) และโมเมนตัมเชิงเส้น (Linear Momentum) ส่วนพลศาสตร์แบบเชิงมุมศึกษาการเคลื่อนที่ของวัตถุใน

แนวเส้นโค้ง รวมถึงสมบัติด้านทอร์ก โมเมนต์ความเฉื่อย ความเฉื่อยการหมุน (Rotational Inertia) การกระจัดเชิงมุม (Angular Displacement) ความเร็วเชิงมุม (Angular Velocity) ความเร่งเชิงมุม (Angular Acceleration) และโมเมนตัมเชิงมุม (Angular Momentum) ซึ่งในการทำปริญญานิพนธ์ฉบับนี้ ศึกษาในส่วนของพลศาสตร์แบบเชิงเส้นเพียงอย่างเดียว

การเคลื่อนที่ตามแนวเส้นตรง เป็นการศึกษาเพื่อพิจารณาตำแหน่ง ความเร็ว และความเร่งของวัตถุในขณะเวลาที่พิจารณา โดย

- ตำแหน่ง (Position) หมายถึง ปริมาณพีชคณิตที่บ่งบอกระยะทางจากจุดอ้างอิง เป็นปริมาณ Scalar

- การขจัด (Displacement) หมายถึง ปริมาณเวกเตอร์ที่ใช้แทนการเปลี่ยนแปลงตำแหน่งของวัตถุ ใช้สัญลักษณ์ $\vec{x}$ เป็นเวกเตอร์กำหนดตำแหน่งของวัตถุ และ $\Delta\vec{x}$ เป็นการขจัดหรือการเปลี่ยนแปลงขนาดและทิศทางของ $\vec{x}$

- ความเร็ว (Velocity) หมายถึง อัตราการขจัดต่อเวลา ซึ่งเป็นเวกเตอร์ที่บ่งบอกถึงอัตราการเปลี่ยนแปลงตำแหน่งของวัตถุเทียบต่อเวลา ซึ่งความเร็วสามารถแบ่งพิจารณาได้เป็นลักษณะต่าง ๆ ดังนี้

- ความเร็วเฉลี่ย (Average velocity) จะเป็นการนำการขจัดที่เกิดขึ้นมาหารด้วยช่วงเวลาที่เกิดการขจัดนั้น นั่นคือ

$$\vec{v}_{avg} = \frac{\Delta\vec{x}}{\Delta t}$$

- ความเร็วที่เวลาใดๆ เป็นการหาอนุพันธ์ของการขจัดเทียบต่อเวลา ในขณะเวลาที่กำลังพิจารณา นั่นคือ

$$\vec{v} = \frac{d\vec{x}}{dt}$$

- Speed หมายถึง ขนาดของความเร็ว หรือ $v = |\vec{v}| = \frac{ds}{dt}$

- Average Speed หมายถึง ระยะทางการเคลื่อนที่ทั้งหมดหารด้วยระยะเวลาที่ใช้ ซึ่งต่างจาก $\vec{v}_{avg}$ เพราะระยะทางการเคลื่อนที่กับการขจัดเป็นปริมาณที่แตกต่างกัน ความแตกต่างของ Total distance และ Displacement

$$\text{Total Distance (s)} = L_1 + L_2$$

$$\text{Displacement } (\Delta\vec{x}) = \sqrt{{L_1}^2 + {L_2}^2}$$

- ความเร่ง (Acceleration) หมายถึง การเปลี่ยนแปลงความเร็วเทียบกับเวลา โดยความเร่งเป็นปริมาณเวกเตอร์ ซึ่ง

$$\vec{v} = \frac{d\vec{v}}{dt} = \frac{d^2\vec{x}}{dt^2}$$

ขนาดของความเร่งจะหาได้จาก

$$a = \frac{dv}{dt} = \frac{d^2s}{dt^2}$$

2.2.2 กฎการเคลื่อนที่ของนิวตัน [9]

กฎข้อที่ 1 วัตถุจะอยู่ในสภาพคงที่หรือเคลื่อนที่อย่างสม่ำเสมอ ถ้าไม่มีแรงจากภายนอกมากระทำต่อวัตถุนั้น

กฎข้อที่ 2 เมื่อมีแรงลัพธ์ที่ไม่เป็นศูนย์มากระทำต่อวัตถุ จะทำให้วัตถุเคลื่อนที่ด้วยความเร่งในทิศเดียวกับทิศของแรงลัพธ์และขนาดของความเร่งนี้จะแปรผันตรงกับขนาดของแรงลัพธ์ และแปรผกผันกับมวลของวัตถุนั้น

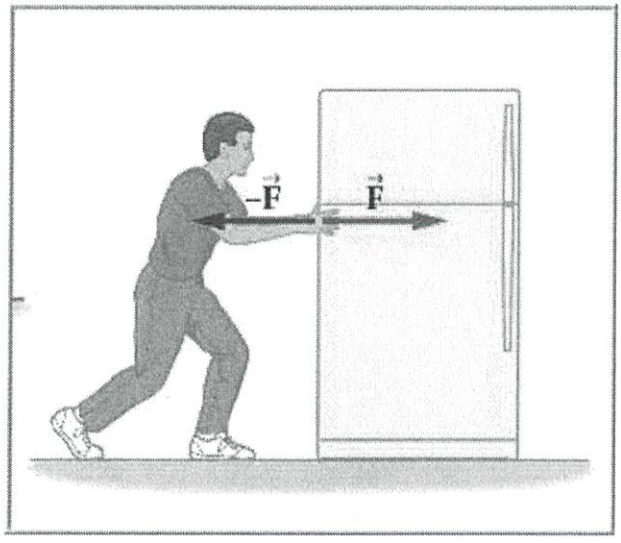
กฎข้อที่ 3 ทุกแรงกิริยาย่อมมีแรงปฏิกิริยาขนาดเท่ากันกระทำในทิศตรงกันข้ามเสมอ

2.2.3 ทฤษฎีแรงกิริยาและแรงปฏิกิริยา

จากกฎข้อที่สามของนิวตัน การเคลื่อนที่ของวัตถุเกิดได้ 2 ลักษณะ ดังนี้

1. การเคลื่อนที่ด้วยแรงกิริยา หมายถึง แรงที่กระทำต่อวัตถุเพื่อเปลี่ยนแปลงสภาพเดิมของวัตถุ และวัตถุเคลื่อนที่ด้วยแรงกระทำต่อวัตถุโดยตรง หรือวัตถุเคลื่อนที่ไปในทิศทางเดียวกับทิศของแรงกระทำต่อวัตถุ เช่น การเคลื่อนที่ของลูกธนู การขว้างก้อนหิน การเคลื่อนที่ของลูกกระสุน เป็นต้น

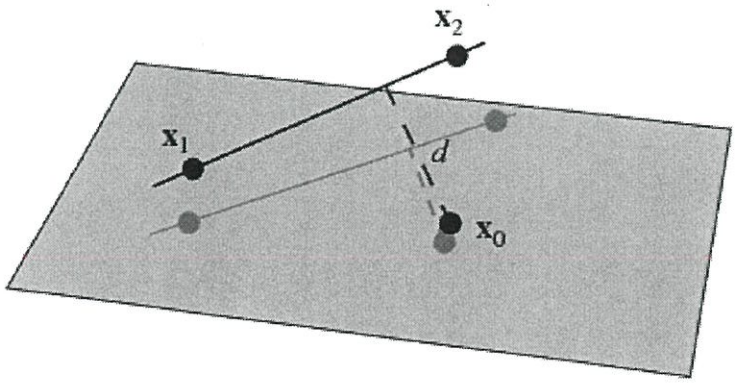
2. การเคลื่อนที่ด้วยแรงปฏิกิริยา หมายถึง แรงที่วัตถุกระทำตอบโต้แรงกิริยาในทิศทางตรงข้ามเป็นการเคลื่อนที่ในลักษณะที่แรงขับดันไปข้างหลัง แล้วมีแรงปฏิกิริยาดันวัตถุให้เคลื่อนที่ไปข้างหน้า เช่น การเคลื่อนที่ของจรวด บั้งไฟ เครื่องบินไอพ่น เรือหางยาว เป็นต้น



รูปที่ 2.15 การเคลื่อนที่ด้วยแรงกิริยาและแรงปฏิกิริยา [10]

2.2.4 ระยะระหว่างจุดและเส้นตรง (Distance between a Point and a Line) [11]

การหาระยะระหว่างจุดกับเส้นเป็นการหาระยะเผื่อเพื่อป้องกันการชนกำแพง ดังแสดงวิธีการหาในรูปที่ 2.16



รูปที่ 2.16 ระยะห่างระหว่างจุดและเส้นตรง [11]

กำหนดเส้นตรงให้เป็น 3 แกน โดยมีจุดต้นอยู่ที่ $x_1 = (x_1, y_1, z_1)$ และ $x_2 = (x_2, y_2, z_2)$ ดังนั้นจะได้เวกเตอร์ดังนี้

$$v = \begin{bmatrix} x_1 + (x_2 - x_1)t \\ y_1 + (y_2 - y_1)t \\ z_1 + (z_2 - z_1)t \end{bmatrix}$$

ซึ่งจุดที่กำหนดไว้ จะอยู่ในตำแหน่ง $x_0 = (x_0, y_0, z_0)$ และจะสามารถหาระยะห่างระหว่างจุดกับเส้นยกกำลัง $2(d^2)$ ได้จาก

$$d^2 = [(x_1 - x_0)+(x_2 - x_1)t]^2 + [(y_1 - y_0)+(y_2 - y_1)t]^2 + [(z_1 - z_0)+(z_2 - z_1)t]^2$$

เพื่อหาระยะทางที่สั้นที่สุด จะทำการ $d(d^2)/dt = 0$ และหาค่า t ออกมา

$$t = - \frac{(x_1 - x_0)\cdot(x_2 - x_1)}{|x_2 - x_1|^2}$$

และเมื่อนำค่าที่แทนลงไปในสมการที่ 2 จะได้ว่า

$$\begin{aligned} d^2 &= (x_1 - x_0)^2 + (y_1 - y_0)^2 + (z_1 - z_0)^2 + 2t\,[(x_2 - x_1)(x_1 - x_0) + (y_2 - y_1) \\ &\quad (y_1 - y_0) + (z_2 - z_1)(z_1 - z_0)] + T^2\,[(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2] \\ &= |x_1 - x_0|^2 - 2\,\frac{[(x_1 - x_0)\cdot(x_2 - x_1)]^2}{|x_2 - x_1|^2} + \frac{[(x_1 - x_0)\cdot(x_2 - x_1)]^2}{|x_2 - x_1|^2} \\ &= \frac{|x_1 - x_0|^2 |x_2 - x_1|^2 [(x_1 - x_0)\cdot(x_2 - x_1)]^2}{|x_2 - x_1|^2} \end{aligned}$$

จาก $(AxB)^2 = A^2B^2 - (A\cdot B)^2$

$$d^2 = \frac{|(x_2 - x_1) * (x_1 - x_0)|^2}{|x_2 - x_1|^2}$$

ทำการถอดรากที่ 2 ออก จะได้คำตอบดังนี้

$$\begin{aligned} d &= \frac{|(x_2 - x_1) * (x_1 - x_0)|}{|x_2 - x_1|} \\ &= \frac{|(x_0 - x_1) * (x_0 - x_2)|}{|x_2 - x_1|} \end{aligned}$$

2.3 ทฤษฎีการสร้างภาพกราฟิก [12]

การสร้างภาพสามมิติบนโปรแกรม Microsoft Visual Studio C++ ด้วยฟังก์ชัน OpenGL (Open Graphics Library) ซึ่งจะช่วยให้การสร้างภาพกราฟิกเป็นเรื่องง่าย และสามารถประยุกต์การใช้งานกราฟิกในขั้นสูงได้อย่างเหมาะสม OpenGL เป็นเครื่องมือที่มีประสิทธิภาพสูงในการเร่งความเร็วแอพพลิเคชั่นสามมิติและเกมต่างๆในปัจจุบัน สามารถใช้ข้อมูลจำนวนมากและสร้างเอ็ฟเฟ็กต์สามมิติในแบบเรียลไทม์ได้อย่างมีประสิทธิภาพ การเพิ่มการสนับสนุน

อุปกรณ์ใหม่ๆลงไปใน OpenGL ที่ทำได้ง่ายและรวดเร็ว ทำงานได้บนหลายแพล็ตฟอร์ม มีเสถียรภาพในการทำงานสูง และสามารถใช้งานร่วมกับคอมไพเลอร์ได้หลากหลาย จึงเป็นเหตุผลสำคัญที่นิยมนำ OpenGL มาใช้งาน

2.3.1 หลักการประยุกต์ใช้งานของ OpenGL [12]

ในปัจจุบันโปรแกรมต่างๆมีรูปแบบที่หลากหลายมากยิ่งขึ้น แต่ละโปรแกรมได้ออกแบบและตกแต่งลักษณะของโปรแกรมให้มีความน่าสนใจและดึงดูดใจผู้ใช้ โดยในหลายๆโปรแกรมนำภาพกราฟิกมาใช้และได้รับความนิยมอย่างแพร่หลายในหมู่ผู้ใช้ แต่โปรแกรมสำหรับสร้างกราฟิกมีการใช้งานค่อนข้างซับซ้อน ยากต่อการเข้าใจสำหรับผู้เริ่มต้นใช้โปรแกรม ดังนั้นการทำความเข้าใจเกี่ยวกับพื้นฐานของ OpenGL จะช่วยให้การสร้างภาพกราฟิกเป็นเรื่องง่าย และสามารถประยุกต์การใช้งานกราฟิกในขั้นสูงได้อย่างเหมาะสม

OpenGL (Open Graphics Library) เป็นซอฟต์แวร์ไลบรารี (Software Library) ที่ใช้ติดต่อกับฮาร์ดแวร์เพื่อการแสดงภาพกราฟิก โดย OpenGL จะมีคำสั่งสำหรับการวาดภาพพื้นฐานคือจุด เส้น และรูปเหลี่ยมต่างๆ และการแสดงภาพแรสเตอร์ ซึ่งคำสั่งพื้นฐานมีอยู่ประมาณ 120 คำสั่งที่สามารถใช้กำหนดคุณลักษณะและควบคุมการทำงานของแอพพลิเคชันสามมิติ ซึ่งผู้พัฒนาโปรแกรมสามารถใช้ไลบรารี OpenGL ได้โดยไม่มีค่าลิขสิทธิ์ ทำให้มีการนำไลบรารีของ OpenGL ไปใช้งานอย่างแพร่หลายในงานกราฟิก

ภาษาที่สามารถใช้กับ OpenGL มีดังนี้ C / C++ (VC++, Borland C++, C++ Builder, C compiler on UNIX), Delphi, Visual Basic, Java, Perl, Python, Fortran, และ Ada เป็นต้น เนื่องจากโครงสร้างของ OpenGL เป็นอินเตอร์เฟสที่เป็นอิสระจากฮาร์ดแวร์ (Hardware-independent interface) และสามารถใช้ได้กับระบบปฏิบัติการหลายๆแบบไม่ว่าจะเป็น Windows, UNIX, (X Window System), IBM OS / 2 หรือ Apple MAC OS ก็ได้ และด้วยเหตุที่ OpenGL ถูกออกแบบให้ทำงานโดยไม่ยึดติดกับระบบ สามารถทำงานได้บนทุกๆแพล็ตฟอร์ม (Independent Platform) ทำให้สามารถเคลื่อนย้ายโค๊ดที่สร้างเรียบร้อยแล้วไปใช้แพล็ตฟอร์มอื่นได้อย่างสะดวกและไม่ต้องเปลี่ยนแปลงโค้ดโปรแกรมเลย

การที่ OpenGL สามารถใช้ได้กับระบบปฏิบัติการที่หลากหลายนี้เอง ทำให้ OpenGL ไม่มีคำสั่งที่จัดการกับระบบปฏิบัติการเลย อีกทั้งยังไม่มีคำสั่งเพื่อรับอินพุตจากผู้ใช้อีกด้วย หน้าที่ทั้งสองอย่างนี้เป็นของผู้เขียนโปรแกรมที่จะต้องออกแบบและเขียนโค้ดเพื่อให้การทำงานเป็นไปอย่างมีประสิทธิภาพ แต่อย่างไรก็ตาม ยังมียูทิลิตี้ที่ช่วยจัดการงานทั้งสองนี้ หากพัฒนาโปรแกรมบนระบบปฏิบัติการ Windows ยูทิลิตี้ดังกล่าวคือ GLUT (OpenGL Utility Toolkit)

นอกจากนี้ OpenGL ยังไม่มีคำสั่งระดับสูงที่จะใช้วาดวัตถุสามมิติแบบซับซ้อน เช่น รถยนต์ อวัยวะ หรือโมเลกุลต่างๆ ที่ OpenGL เตรียมไว้มีเพียงการสร้างรูปจำลองสามมิติคือ รูปทรงเลขาคณิตอย่างง่าย ได้แก่ จุด เส้น และรูปหลายเหลี่ยม ซึ่งผู้ใช้งานจะต้องนำรูปทรงเหล่านี้มาประกอบกันเพื่อให้เกิดรูปทรงสามมิติที่ซับซ้อน

โครงสร้างของ OpenGL มีลักษณะคล้าย API (Application Programming Interface) เป็นโปรแกรมที่ทำหน้าที่เชื่อมต่อระหว่างแอพพลิเคชันกับฮาร์ดแวร์ โดยโปรแกรมเมอร์ที่เขียนแอพพลิเคชันต่างๆ ไม่จำเป็นต้องทราบการทำงานของฮาร์ดแวร์และไม่จำเป็นต้องส่งคำสั่งไปให้ฮาร์ดแวร์ทำงานโดยตรง ความรู้ทางด้าน OpenGL สามารถนำไปใช้กับซอฟต์แวร์อื่นๆ ได้เช่นกัน ถึงแม้ว่า OpenGL จะง่ายในการเรียนรู้เมื่อเปรียบเทียบกับ API อื่นๆ แต่มีความสามารถสูงมาก เพราะ OpenGL สนับสนุนทั้งภาพ 2 และ 3 มิติ รวมทั้งสนับสนุนเทคนิคการแสดงภาพระดับสูงอีกด้วย การทำงานของระบบกราฟิก ผู้ใช้ไม่จำเป็นต้องทราบเหตุผลว่ามีการทำงานอย่างไร รู้แต่เพียงว่าอินพุต และเอาต์พุต คืออะไรก็เพียงพอแล้ว ในระบบกราฟิกจะมีฟังก์ชันที่ถูกเรียกใช้ โดยมีการรับค่าของอินพุตจากอุปกรณ์ต่างๆ เช่น เมาส์หรือคีย์บอร์ด หรืออุปกรณ์อื่นๆ เช่น แมสเสจจากระบบปฏิบัติการ หรืออินเทอร์รัพต์ก็ได้

ผลลัพธ์ที่ได้จากการทำงานจะแสดงออกไปยังอุปกรณ์แสดงผล เช่น จอภาพ CRT หรือจอ LCD ทั้งนี้เอาต์พุตอาจจะมองอินพุตเป็นฟังก์ชันที่ผู้ใช้เรียกใช้และเอาต์พุตคือ ผลของการทำงานที่แสดงยังจอภาพ ดังรูป

รูปที่ 2.17 ระบบกราฟิกที่เปรียบเสมือนกล่องดำ (Black Box) [12]

โครงสร้างไลบรารีพื้นฐานของ OpenGL (ที่เรียกว่า OpenGL) จะมีฟังก์ชันเก็บอยู่ในไลบรารี GL นั้น ฟังก์ชันเหล่านี้จะขึ้นต้นด้วย gl หลังจากนั้นจะมีชื่อฟังก์ชันที่ขึ้นต้นตัวแรกด้วยตัวอักษรพิมพ์ใหญ่ ข้างล่างนี้เป็นตัวอย่างของชื่อฟังก์ชัน

glBegin, glClear, glCopyPixels, glPolygonMode

ฟังก์ชันเฉพาะอย่างนี้ต้องใช้อาร์กิวเมนต์ 1 ตัวหรือมากกว่า 1 ตัวซึ่งขึ้นอยู่กับฟังก์ชัน อาร์กิวเมนต์อาจจะเป็นสัญลักษณ์เฉพาะ เช่น ค่าคงที่, ชื่อพารามิเตอร์, ค่าของพารามิเตอร์ หรือโหมดของพารามิเตอร์ เป็นต้น ค่าคงที่ทั้งหมดนี้จะขึ้นต้นด้วยตัวอักษรพิมพ์ใหญ่ GL นอกจากนี้ยังมีเครื่องหมายขีดล่าง (Underscore) _ เพื่อคั่นระหว่างคอมโพเนนต์ ตัวอย่างข้างล่างเป็นตัวอย่างชื่อฟังก์ชันที่ใช้เครื่องหมาย_ คั่นระหว่างคอมโพเนนต์

GL_2D, GL_RGB, GL_POLYGON, GL_AMBIENT_AND_DIFFUSE

ฟังก์ชันของ OpenGL อาจจะกำหนดประเภทข้อมูลได้ เช่น ใช้กำหนดประเภทข้อมูลตัวเลขจำนวนเต็ม 32 บิต แต่ขนาดของการกำหนดตัวเลขจำนวนเต็มอาจจะแตกต่างกันไปตามเครื่อง การกำหนดค่าประเภทข้อมูล OpenGL จะใช้ชื่อประเภทข้อมูลที่มีในฟังก์ชัน โดยชื่อประเภทข้อมูลนี้จะขึ้นต้นด้วยตัวพิมพ์ใหญ่ GL ต่อด้วยชื่อประเภทข้อมูลมาตรฐานที่เป็นตัวพิมพ์เล็ก (หรืออาจจะใช้ชื่อธรรมดาก็ได้ เช่น int, float) ดังตัวอย่าง

GLbyte, GLshort, Glint, GLfloat, GLdouble, GLboolean

บางอาร์กิวเมนต์ของฟังก์ชัน OpenGL สามารถกำหนดค่าโดยใช้อาร์เรย์ที่เป็นลิสต์ของค่าข้อมูลก็ได้ มีออปชันในการกำหนดลิสต์ของค่าข้อมูลเป็นพอยเตอร์ของอาร์เรย์ ยิ่งกว่านั้นยังสามารถกำหนดลิสต์ในลักษณะ Explicit ให้เป็นพารามิเตอร์ของอาร์กิวเมนต์ได้อีกด้วย ตัวอย่างทั่วไปในการใช้ออปชันนี้ก็คือการกำหนดค่าโคออร์ดิเนต xyz นั่นเอง นอกจากไลบรารีหลักของ OpenGL แล้ว ยังมีไลบรารีที่เกี่ยวข้องอีกเป็นจำนวนมากเพื่อจัดการกับงานเฉพาะอย่างได้อย่างมีประสิทธิภาพ ไลบรารีที่เกี่ยวข้องกับ OpenGL และเป็นไลบรารีเฉพาะมีดังนี้

- OpenGL Utilities (GLU) เป็นไลบรารีที่ประกอบด้วยรูทีนมากมายในการจัดการมุมมองเพื่อแสดงรูปพื้นฐานและออปเจ็กต์ที่ซับซ้อนที่ประกอบขึ้นจากเส้นและรูปหลายเหลี่ยม, แสดงรูปลูกบาศก์, การเรนเดอร์พื้นผิว (surface-rendering) และงานที่ซับซ้อน

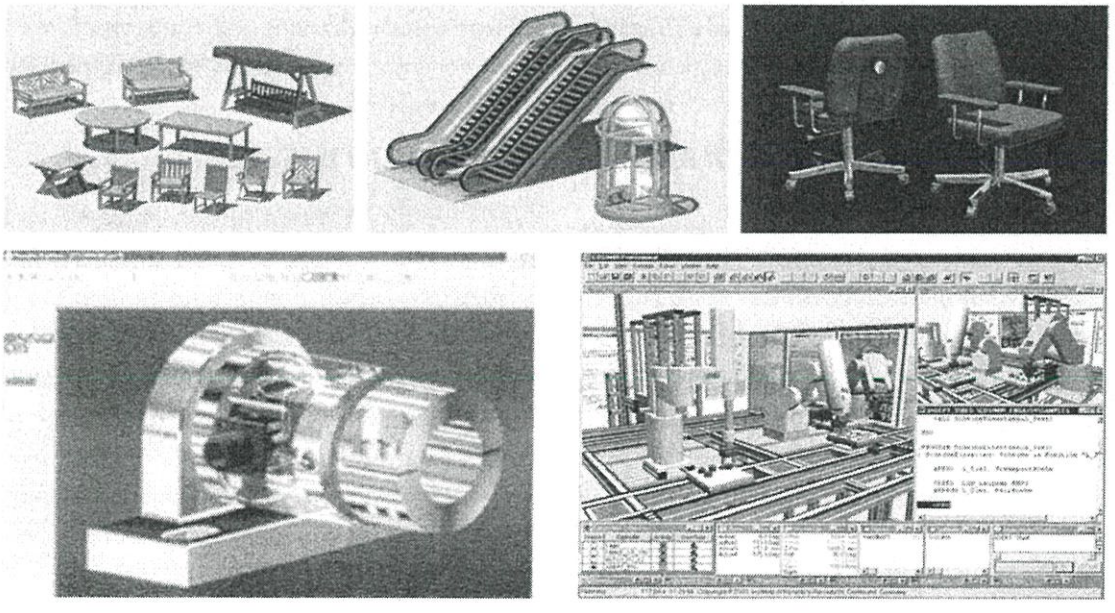
- OpenGL Utilitiy Toolkit (GLUT) คือไลบรารีของระบบกราฟิกที่ช่วยในการติดต่อกับการแสดงผลทางจอภาพ ทั้งนี้เนื่องจากในการใช้งาน OpenGL เพื่อใช้งานด้านกราฟิกสิ่งแรกที่มีความจำเป็นต้องทำคือการกำหนดวินโดว์สำหรับการแสดงผล (Display Window) บนจอภาพ ซึ่งวินโดว์ดังกล่าวนี้คือพื้นที่สี่เหลี่ยมผืนผ้าของจอภาพที่ใช้แสดงกราฟิก คุณไม่สามารถสร้างวินโดว์นี้ได้โดยตรงจากฟังก์ชันของ OpenGL เนื่องจากไลบรารีนี้ประกอบเพียงฟังก์ชันทางด้านกราฟิกที่ไม่ขึ้นกับอุปกรณ์ใดๆ การเขียนโปรแกรมโดยใช้ OpenGL เหมาะสำหรับการพัฒนาโปรแกรมขนาดเล็ก

ถึงขนาดกลาง คำสั่งของ GLUT จะขึ้นต้นด้วยคำว่า glut เสมอ การเรียกใช้จะต้อง Include ไฟล์ Header ที่ชื่อ glut.h ในตอนต้นของโค้ดโปรแกรมเช่นกัน ถ้าคุณใช้ glut.h แล้ว ก็ไม่จำเป็นต้องใช้ gl.h และ glu.h อีก ไลบรารีเหล่านี้ยังประกอบด้วยเมธอดสำหรับการสร้างและแรนเดอร์ Quadric Curves และพื้นผิว

- ไลบรารีอื่นๆ ที่ช่วยในการแสดงผลของระบบต่างๆนอกจาก GLUT ซึ่งเป็นไลบรารีของระบบกราฟิกที่ช่วยในการติดต่อกับการแสดงผลทางจอภาพทั่วไปแล้ว ในบางระบบจะมีไลบรารีเฉพาะอย่างแยกออกจากกัน OpenGL Extension to the X window System (GLX) เป็นไลบรารีสำหรับระบบปฏิบัติการ UNIX ที่ใช้ X Window ในไลบรารีนี้ประกอบด้วยชุดของรูทีนสำหรับการแสดงผลโดยชื่อฟังก์ชันจะขึ้นต้นด้วยคำว่า glX ส่วนในเครื่องแอปเปิ้ลจะใช้ Apple GL (AGL) เป็นอินเทอร์เฟซสำหรับการจัดการวินโดว์ ชื่อฟังก์ชันของไลบรารีนี้ขึ้นต้นด้วย agl

2.3.2 OpenGL กับงานคอมพิวเตอร์กราฟิก [12]

คอมพิวเตอร์กราฟิกถูกนำมาใช้ในการแสดงผลภาพกราฟ แผนภาพของข้อมูล และวัตถุรูปร่างต่างๆให้ปรากฏบนจอภาพคอมพิวเตอร์ได้เป็นอย่างดี สามารถสร้างกราฟได้หลายรูปแบบ เช่น กราฟเส้น กราฟแท่ง และกราฟวงกลม นอกจากนี้ยังสามารถแสดงภาพได้ทั้งในรูปแบบสองมิติ และสามมิติ ทำให้กราฟหรือภาพเหล่านั้นดูดีและน่าสนใจมากยิ่งขึ้น ซึ่งเป็นประโยชน์ต่อผู้บริหารหรือผู้จัดการกิจการเป็นอย่างยิ่ง เนื่องจากสามารถทำความเข้าใจกับข้อมูลได้ง่ายและรวดเร็วกว่าเดิม



รูปที่ 2.18 ภาพตัวอย่างการนำเอา OpenGL มาประยุกต์ใช้กับคอมพิวเตอร์กราฟิก [2]

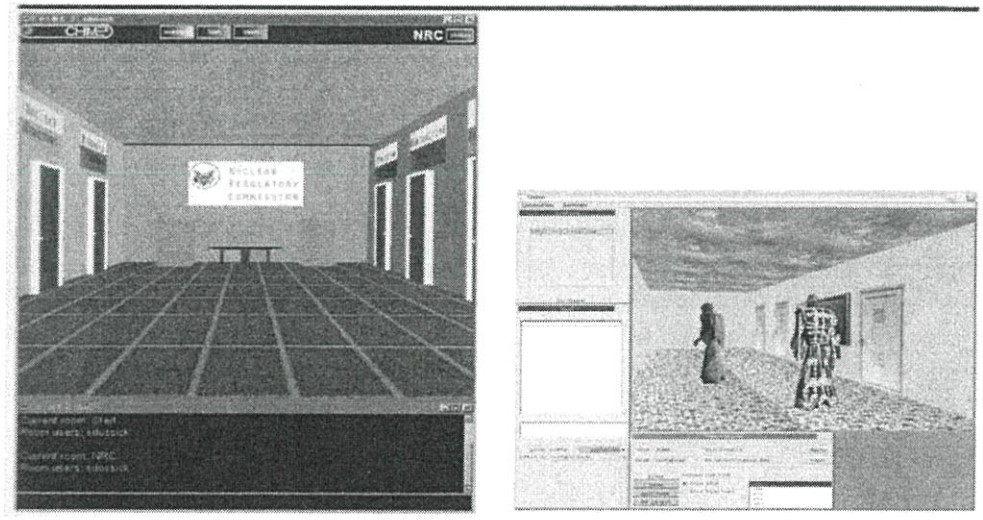
นอกจากนี้คอมพิวเตอร์กราฟิกยังถูกในมาใช้กับการจำลองสถานการณ์ (Simulation) เพื่อหาคำตอบว่าถ้าเกิดสถานการณ์ใดๆแล้วจะเกิดผลอย่างไร เช่น ผู้ผลิตรายหนึ่งมีความประสงค์ที่จะก่อสร้างโรงงานผลิตสินค้าขนาดใหญ่ โดยต้องการให้การผลิตสินค้านั้นสะดวกแก่การปฏิบัติงาน การไหลของวัตถุดิบมีความคล่องตัว ลดระยะเวลาของการเคลื่อนย้ายวัตถุดิบไปตามกระบวนการต่างๆ ใช้พื้นที่ให้เกิดประโยชน์สูงสุด อำนวยความสะดวกและความปลอดภัยภายในโรงงาน ผู้ผลิตรายดังกล่าวจำเป็นต้องแบ่งสัดส่วนภายในโรงงานอย่างไร ควรติดตั้งเครื่องจักรไว้บริเวณไหน ทิศทาง

ใด จะไม่เกิดอันตรายต่อผู้ปฏิบัติงาน ดังนั้นการนำการจำลองสถานการณ์ด้วยคอมพิวเตอร์กราฟิกมาใช้จึงช่วยให้ทราบผลล่วงหน้า ประหยัดค่าใช้จ่าย และไม่เกิดอันตราย

ต่อมาได้มีการนำเอาคอมพิวเตอร์กราฟิกเข้ามามีส่วนร่วมกับการพัฒนาระบบสื่อประสมทำให้เครื่องคอมพิวเตอร์เพียงเครื่องเดียวสามารถทำหน้าที่ได้หลายลักษณะ เช่น การทำงานเป็นเครื่องวีดีทัศน์ที่แสดงภาพและตัวอักษรทั้งภาพนิ่งและภาพเคลื่อนไหวหรืออาจจะมีเสียงประกอบ หัวใจของของการผลิตระบบสื่อประสมคือ เทคโนโลยีการแปลงข้อมูลอนาล็อกให้เป็นดิจิตอล ดังนั้นเมื่อข้อมูลต่างๆถูกแปลงให้อยู่ในรูปของดิจิตอลแล้วจึงสามารถนำมาแก้ไขดัดแปลงโดยซอฟต์แวร์ได้อย่างมีข้อจำกัด กระบวนการปรับแต่งข้อมูลดิจิตอลนี้จึงเป็นช่องทางสำคัญสำหรับการสร้างสรรค์รูปแบบใหม่ๆของผลงาน ตัวอักษร กราฟิก ภาพถ่าย เสียง และภาพเคลื่อนไหว

การพัฒนารูปแบบของระบบสื่อประสม ซึ่งบรรจุข้อมูลไว้เป็นจำนวนมากให้สามารถโยงความเกี่ยวเนื่องของข้อมูลทำให้เกิด สื่อหลายมิติ (Hypermedia) นั้นคือ การรวมเอาสื่อกราฟิก เสียง วีดีทัศน์ และระบบที่เกี่ยวข้องกับการจัดเก็บและแสดงผลข้อมูลในรูปแบบอื่นๆไว้ซึ่งเป็นรูปแบบโต้ตอบที่เกี่ยวข้องกับเนื้อหาที่ผู้ใช้กำลังสนใจอยู่ได้อย่างไม่มีข้อจำกัด ปัจจุบันสื่อหลายมิติมีการพัฒนาให้ผู้ใช้มีปฏิสัมพันธ์กับโปรแกรมมากยิ่งขึ้น โดยเฉพาะการนำระบบเสมือนจริงมาใช้กับสื่อประเภทนี้ ทำให้บทบาทของคอมพิวเตอร์จากเดิมที่เป็นเครื่องมือประมวลผลทางคณิตศาสตร์และวิทยาศาสตร์ กลายมาเป็นเครื่องที่ใช้สำหรับการเรียนรู้ และการสร้างสรรค์ผลงาน ที่สนองต่อความงาม ความเพลิดเพลินและความไพเราะ

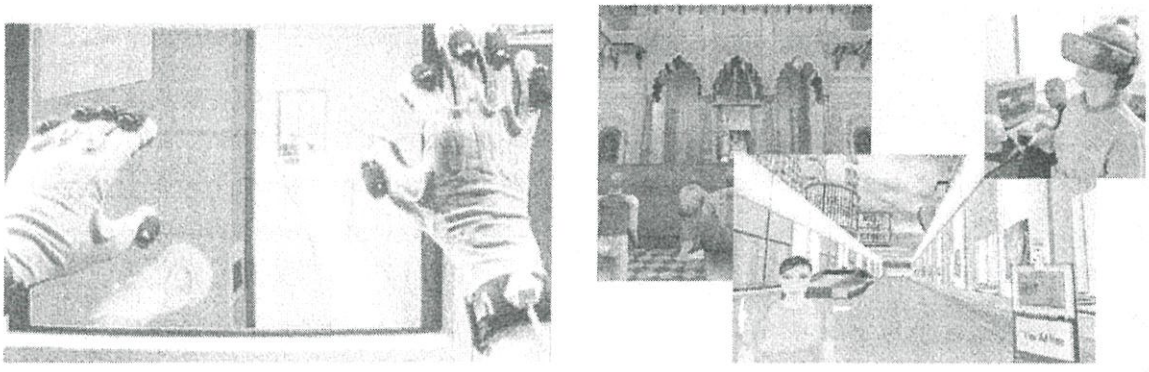
การพัฒนาซอฟต์แวร์สื่อประสม ที่ให้การใช้งานเชิงตอบโต้ได้มากยิ่งขึ้นซึ่งอาจเป็นคอมพิวเตอร์กับผู้ใช้หรือสามารถตอบโต้กันระหว่างผู้ใช้กับผู้ใช้ด้วยกัน โดยอาศัยระบบ GUI (Graphical User Interface) ที่ให้ความเป็นจริงเสมือน สามารถสนองการรับรู้ของมนุษย์ที่มากขึ้นกว่าเดิมโดยการอุปกรณ์รอบข้างที่เอื้อต่อการพฤติกรรมการใช้งานของผู้ใช้ เพื่อให้สื่อประสมกลายเป็นสื่อหลายมิติที่มีความสมบูรณ์แบบมากยิ่งขึ้น



รูปที่ 2.19 ตัวอย่างของสื่อหลายมิติ (Hypermedia) [2]

แต่เดิมผู้ใช้งานคอมพิวเตอร์กราฟิกมักจะจำการทำงานอยู่ที่หน้าจอคอมพิวเตอร์ที่มีลักษณะเป็นกรอบภาพสองมิติ ในขณะที่ความเป็นในการสร้างสถานการณ์จำลองต่างๆ ทำให้ต้องมีการคิดค้นอุปกรณ์จำลองเพื่อตอบสนองความรู้สึกของผู้ใช้ให้เสมือนกับอยู่บนความเป็นจริงมากที่สุด

การพัฒนาระบบเสมือนจริง คือ การใช้คอมพิวเตอร์สร้างสิ่งแวดล้อมใหม่ที่ทำให้ผู้ใช้รู้สึกเหมือนเข้าไปอยู่ในที่ว่างสามมิติที่มองเห็น หรือรับรู้สภาวะสามมิติจากตำแหน่งต่างๆในช่องว่างหรือสามารถเคลื่อนย้ายเปลี่ยนแปลงวัตถุในสิ่งแวดล้อมนั้นได้ แนวความคิดนี้จึงทำให้มีการพัฒนาอุปกรณ์ความจริงเสมือนที่ต้องการตอบสนองต่อพฤติกรรมการสื่อสารและรับรู้ของมนุษย์มากยิ่งขึ้น การนำอุปกรณ์จอภาพสวมศีรษะมาแสดงผลภาพให้ปรากฏบนกระจกหรือจอภาพขนาดเล็กที่ติดตั้งไว้ที่ตาทั้งสองของผู้ใช้ จึงเป็นการพัฒนาระบบเสมือนจริงที่มีความสมบูรณ์มากยิ่งขึ้น และยิ่งให้ความสมบูรณ์ขึ้นไปอีกเมื่อนำมาใช้ร่วมกับอุปกรณ์อื่นที่ให้ความจริงเสมือนเช่น ถุงมือข้อมูล (Data Gloves) ชุดแต่งกายข้อมูล (Data Suite) เป็นต้น ที่สามารถถ่ายทอดการเคลื่อนไหวของร่างกายของผู้ใช้ไปยังคอมพิวเตอร์



รูปที่ 2.20 ถุงมือข้อมูล (Data Gloves) และอุปกรณ์ที่ใช้ในระบบเสมือนจริง [2]

2.3.3. ฟังก์ชันแสดงรูปภาพทางเรขาคณิตพื้นฐาน [12]

OpenGL ได้เตรียมฟังก์ชันพื้นฐานไว้มากมายเพื่อให้คุณสามารถนำไปสร้างรูปภาพทางเรขาคณิตพื้นฐานไม่ว่าจะเป็นจุด (Point), เส้น (Line), รูปสามเหลี่ยม (Triangle), รูปสี่เหลี่นม (Quad), และรูปหลายเหลี่ยม (Polygon) รายละเอียดแต่ละฟังก์ชันมีดังนี้

1. GL_POINTS สำหรับการพล็อตแต่ละจุดตามตำแหน่งที่กำหนด

2. GL_LINES สำหรับการวาดเส้นตรงโดยเป็นการลากเส้นตรงเชื่อมระหว่างจุด 2 จุด ทำให้ได้ส่วนของเส้นตรงที่ไม่เชื่อมต่อกับเส้นอื่น เช่น ถ้ามีจุด p0-p3 จะเป็นการลากเส้นระหว่าง p0-p1 และ p2-p3 เป็นต้น ถ้ามีจุดเป็นจำนวนคี่จะทำให้จุดสุดท้ายถูกละทิ้งไป

3. GL_LINE_STRIP สำหรับการวาดเส้นตรงระหว่างจุด 2 จุดเช่นเดียวกับ GL_LINES แต่ฟังก์ชันนี้จะแตกต่างกับฟังก์ชัน GL_LINES ตรงที่จะมีการลากเส้นเชื่อมกับเส้นตรงอื่นด้วย เช่น ถ้ามีจุด p0-p3 ฟังก์ชันนี้จะทำการวาดเส้นระว่าง p0-p1, p1-p2, p2-p3 เป็นต้น

4. GL_LINE_LOOP สำหรับการวาดเส้นตรงระหว่างจุด 2 จุดเช่นเดียวกับ GL_LINE_STRIP แต่หลังจากลาดเส้นสุดท้ายแล้วจะมีการเชื่อมระหว่างจุดสุดท้ายกลับมาที่จุดเริ่มต้นทำให้เกิดเป็นรูป ปิด ดังนั้นถ้ามีจุด p0-p3 ฟังก์ชันนี้จะทำการวาดเส้นระหว่าง p0-p1, p1-p2, p2-p3 และ p3-p0 เป็นต้น

5. GL_TRIANGLES สำหรับการวาดรูปสามเหลี่ยมโดยใช้จุด 3 จุด ถ้ามีจำนวนจุดไม่ครบ 3 จุด จุดที่เหลือเศษนี้จะถูกละทิ้งไป เช่น ถ้ามีจุด v0-v5 ฟังก์ชันนี้จะสร้างสามเหลี่ยมจาก v0-v1-v2 และ v3-v4-v5 เป็นต้น

6. GL_TRIANGLES_STRIP สำหรับการวาดรูปสามเหลี่ยมเช่นเดียวกับฟังก์ชัน GL_TRIANGLE แต่จะสร้างสามเหลี่ยมต่อเนื่องโดยใช้จุด 3 จุดที่เปลี่ยนกลุ่มไปทำให้ได้เหมือนแผ่นพับที่เป็นพื้นผิว เช่น ถ้ามีจุด v0-v5 ฟังก์ชันนี้จะสร้างรูป 3 เหลี่ยมจากชุดของจุดคือ v0-v1-v2, v2-v1-v3, v2-v3-v4, v4-v3-v5 เป็นต้น

7. GL_TRIANGLE_FAN สำหรับการวาดรูปสามเหลี่ยมในลักษณะของพัด โดยใช้จุดแรกเป็นจุดยอดของสามเหลี่ยม เช่น มีจุด v0-v5 ฟังก์ชันนี้จะสร้างสามเหลี่ยม คือ v0-v1-v2, v0-v2-v3, v0-v3-v4, v-v4-v5 เป็นต้น

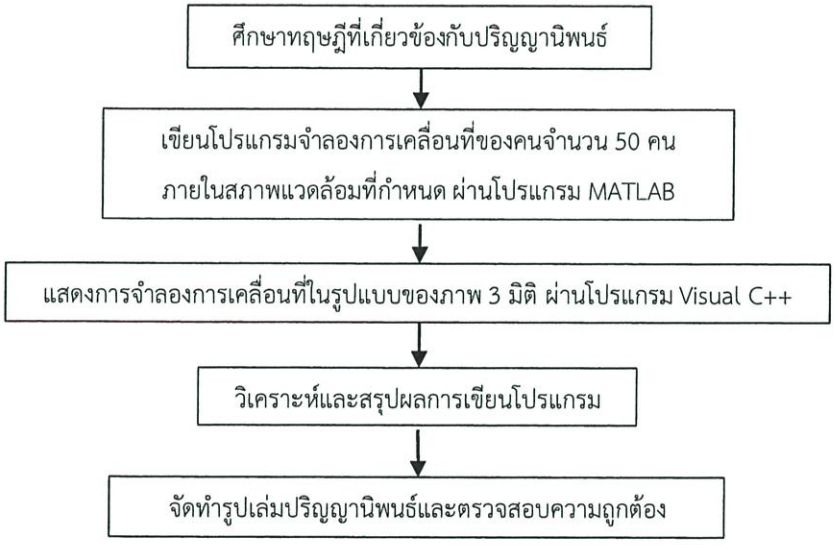
8. GL_QUADS สำหรับวาดรูปสี่เหลี่ยมโดยใช้จุด 4 จุดถ้ามีไม่ครบ 4 จุด จุดที่เหลือเศษจะถูกละทิ้งไป เช่น ถ้ามี v0-v7 ฟังก์ชันจะสร้างสี่เหลี่ยมจาก v0-v1-v2-v3 และ v4-v5-v6-v7 เป็นต้น

9. GL_QUAD_STRIP สำหรับการสาดรูปสี่เหลี่ยมด้วยฟังก์ชัน GL_QUADS แต่ละฟังก์ชันนี้จะวาดรูปสี่เหลี่ยมต่อเนื่องโดยใช้ชุดของจุดที่เปลี่ยนไป เช่น ถ้ามีจุด v0-v7 จะวาดรูปสี่เหลี่ยม v0-v1-v3-v2, v2-v3-v5-v4, v4-v5-v7-v6 เป็นต้น

10. GL_POLYGON สำหรับการวาดรูปสี่เหลี่ยมตามจำนวนจุดที่กำหนด โดยรูปหลายเหลี่ยมนี้จะต้องไม่มีส่วนที่ตัดกัน

บทที่ 3
วิธีการดำเนินการ

การจัดทำวิทยานิพนธ์เรื่องโปรแกรมจำลองสถานการณ์เสมือนจริงผ่านแว่นตาสามมิติเพื่อการจัดการความปลอดภัย ซึ่งการเขียนแบบจำลองสถานการณ์เสมือนจริงนั้นถูกเขียนขึ้นผ่าน โปรแกรม MATLAB บนระบบปฏิบัติการวินโดว์ (Windows) ซึ่งมีขั้นตอนและวิธีการดำเนินงานดังแผนผังในรูปที่ 3.1



รูปที่ 3.1 แผนผังขั้นตอนและวิธีการดำเนินงาน

3.1 ศึกษาทฤษฎีที่เกี่ยวข้องกับปริญญานิพนธ์

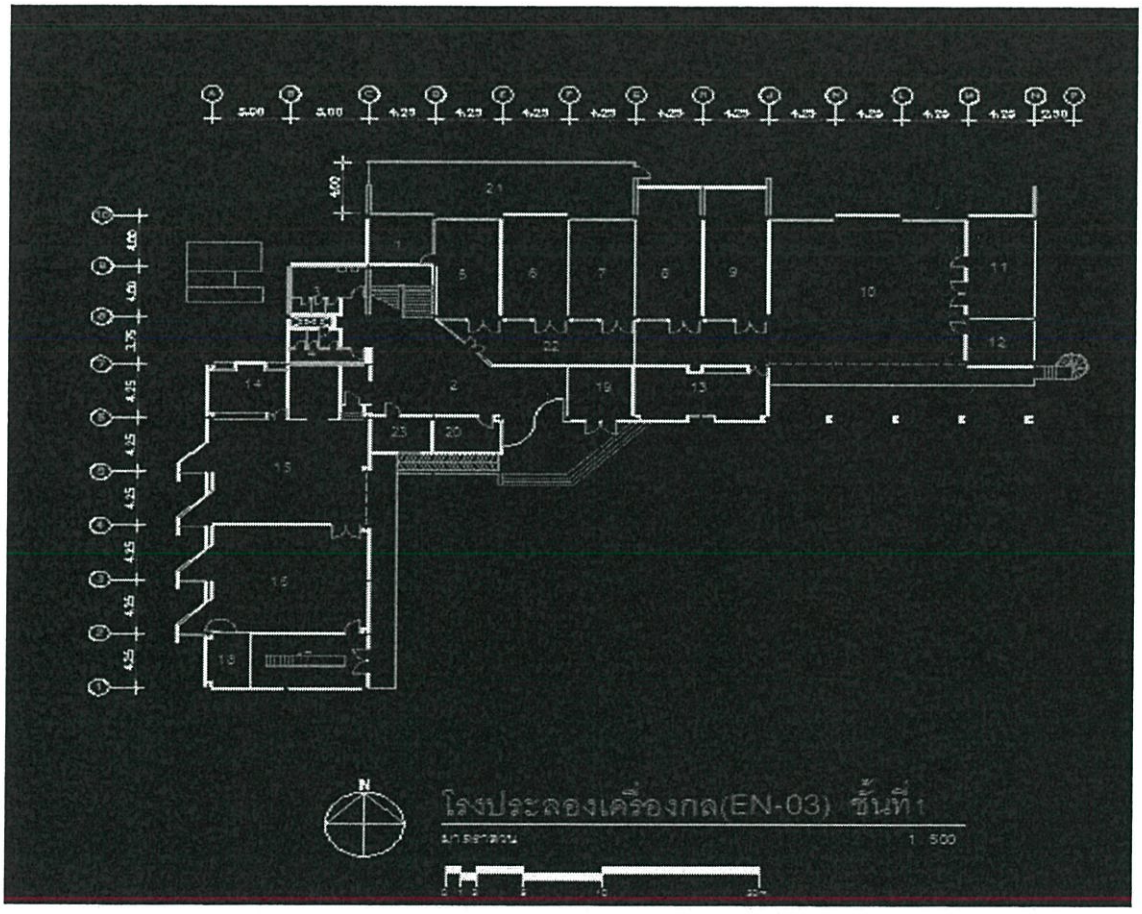
การศึกษาทฤษฎีที่ใช้ในการจำลองสถานการณ์ เป็นการศึกษาทฤษฎีทางคณิตศาสตร์และฟิสิกส์ พร้อมนำมาประยุกต์ใช้ในการเขียนโปรแกรมจำลองสถานการณ์เสมือนจริงผ่านแว่นตาสามมิติ เพื่อการจัดการความปลอดภัย ซึ่งโปรแกรมถูกเขียนขึ้นผ่านโปรแกรม MATLAB โดยการสร้างสมการทางคณิตศาสตร์และฟิสิกส์ เพื่อจำลองสถานที่และจำลองการเคลื่อนที่ของคนจำนวน 50 คน ขณะอยู่ในโรงปฏิบัติการทางวิศวกรรม คณะวิศวกรรมศาสตร์ สาขาวิชาวิศวกรรมอุตสาหการ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ในกรณีเกิดอุบัติเหตุหรืออุบัติภัยขึ้น

3.2 เขียนโปรแกรมจำลองการเคลื่อนที่ของคนภายในสภาพแวดล้อมที่กำหนด

ในการจำลองการเคลื่อนที่ของคนในกระบวนการหลบหนี เพื่อการจัดการความปลอดภัยนั้น ต้องคำนึงถึงปัจจัยต่างๆมากมาย เช่น ขอบเขตของสภาพแวดล้อมที่ต้องการจำลอง จำนวนคนที่พิจารณา แรงกระทำต่างๆที่อาจเกิดขึ้นในกรณีเกิดอุบัติเหตุหรืออุบัติภัยต่างๆ เป็นต้น ซึ่งในการเขียนโปรแกรมจำลองการเคลื่อนที่ของคนภายในสภาพแวดล้อมที่กำหนดนั้น พิจารณาการจำลองการเคลื่อนที่ ภายในโรงปฏิบัติการทางวิศวกรรม คณะวิศวกรรมศาสตร์ สาขาวิชาวิศวกรรมอุตสาหการ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง โดยจำลองการเคลื่อนที่ของคนจำนวน 50 คน ซึ่งมีขั้นตอนการดำเนินการดังต่อไปนี้

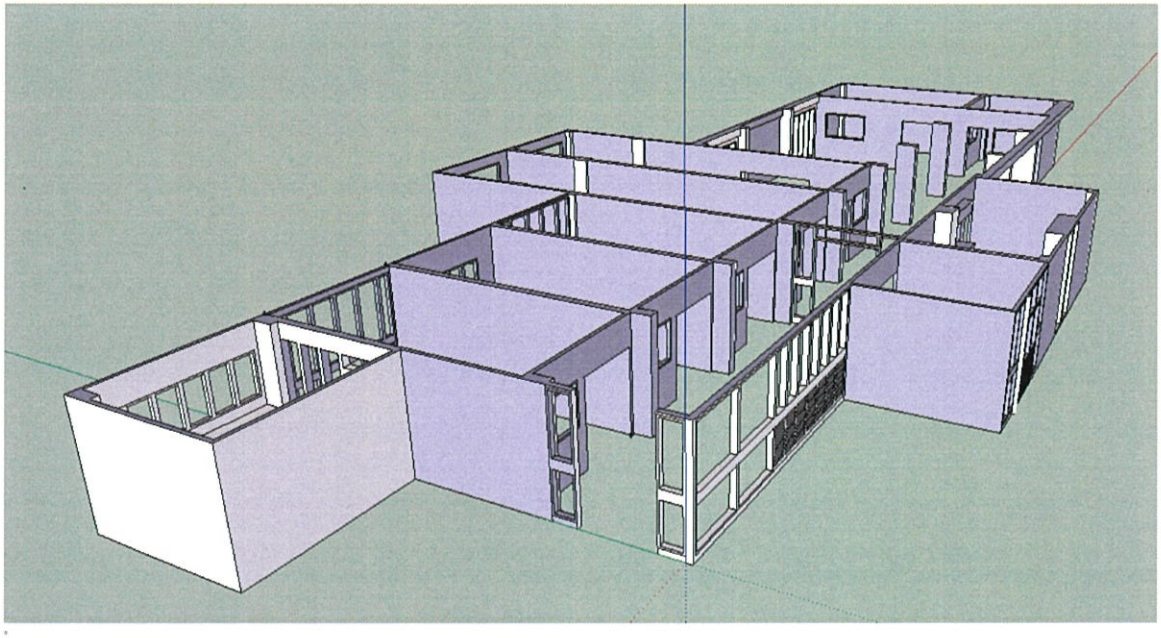
3.2.1 การสร้างขอบเขตของสภาพแวดล้อมเสมือนจริง

ผังโรงงานที่นำมาใช้สำหรับจัดเก็บเป็นฐานข้อมูลอ้างอิงในการสำหรับการจำลองสถานการณ์ในการเคลื่อนที่นั้น คือ โรงปฏิบัติการทางวิศวกรรม คณะวิศวกรรมศาสตร์ สาขาวิศวกรรมอุตสาหการ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ซึ่งมีขนาดและอัตราส่วนของพื้นที่ใช้สอยเป็นค่าตามความเป็นจริงของโรงปฏิบัติการที่ถูกสร้างขึ้น



รูปที่ 3.2 ผังโรงปฏิบัติการทางวิศวกรรม คณะวิศวกรรมศาสตร์ สาขาวิศวกรรมอุตสาหการ

เนื่องจากมีภาพกราฟิกของ Work Shop จากผู้วิจัยก่อนหน้า จึงทำการวัดขนาดด้วยโปรแกรม Google Sketch Up ภายหลังได้ถูกนำมาสร้างในรูปแบบของภาพมุมมองด้านบนด้วยโปรแกรม MATLAB และได้ทำการเขียนโปรแกรมสร้างห้องจำนวน 10 ห้อง รวมถึงพื้นที่ภายใน Work Shop และทางเดิน



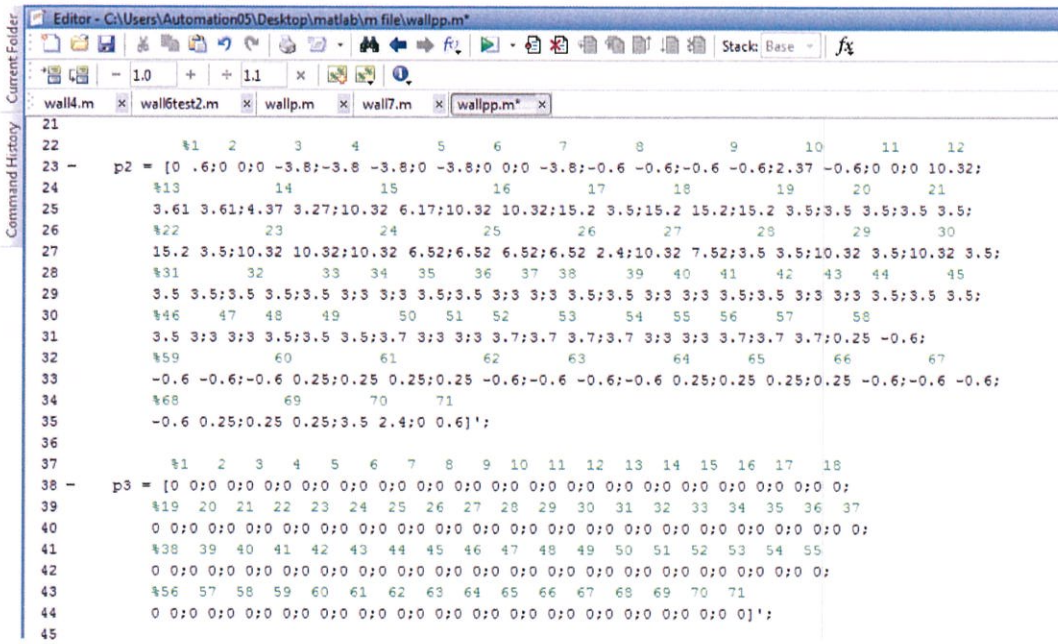
รูปที่ 3.3 การวัดขนาดของ Work Shop

Editor - C:\Users\Automation05\Desktop\matlab\m file\wallpp.m*

wall4.m | wall6test2.m | wallp.m | wall7.m | wallpp.m*

```
clear all;clc;
tic
       %1   2       3       4           5           6           7           8
p1 = [0 0;0 9.45;7.5 7.5;7.5 19.25;11.25 11.25;11.25 18.35;19.25 19.25;19.25 22.8;
     %9          10    11        12        13      14     15          16
     27.15 31;31 31;31 34.75;34.75 34.75;31 34.75;31 31;31 31;34.75 19.25;
     %17           18          19          20          21          22
     19.25 19.25;11.25 19.25;15.25 15.25;17.05 19.25;11.25 13.45;11.25 11.25;
     %23           24     25     26  27    28    29          30      31      32        33
     -4.3 11.25;-4.3 -4.3;-4.3 0;0 0;0 0;0 1.95;3.75 3.75;7.5 7.5;3.75 5.7;7.5 9.45;3.75 3.75;
     %34         35       36       37       38       39          40          41          42
     3.75 4.05;4.05 4.05;7.5 7.5;7.5 7.8;7.8 7.8;11.25 11.25;11.25 11.55;11.55 11.55;15.25 15.25;
     % 43          44          45       46          47          48          49          50
     15.25 15.55;15.55 15.55;15.55 15.25;19.25 19.25;19.25 19.55;19.55 19.55;19.55 19.25;22.5 22.5;
     %51          52       53       54          55          56          57          58
     22.5 22.8;22.8 22.8;22.8 22.5;27.15 27.15;27.15 27.45;27.45 27.45;27.45 27.15;19.25 19.25;
     %59           60          61          62          63       64       65          66
     19.25 19.55;19.55 19.55;19.55 19.25;22.5 22.5;22.5 22.8;22.8 22.8;22.8 22.5;27.15 27.15;
     %67           68          69          70          71
     27.15 27.45;27.45 27.45;27.45 27.15;11.55 11.55;11.55 11.55]';
```

รูปที่ 3.4 โค้ดที่ใช้จำลองตำแหน่งของกำแพงบนแกน X บนโปรแกรม MATLAB



```
Editor - C:\Users\Automation05\Desktop\matlab\m file\wallpp.m*
wall4.m  wall6test2.m  wallp.m  wall7.m  wallpp.m*

            %1   2        3       4             5        6          7           8              9            10          11        12
    p2 = [0 .6;0 0;0 -3.8;-3.8 -3.8;0 -3.8;0 0;0 -3.8;-0.6 -0.6;-0.6 -0.6;2.37 -0.6;0 0;0 10.32;
        %13              14               15                16               17              18                 19             20           21
        3.61 3.61;4.37 3.27;10.32 6.17;10.32 10.32;15.2 3.5;15.2 15.2;15.2 3.5;3.5 3.5;3.5 3.5;
        %22           23              24                25              26             27               28              29           30
        15.2 3.5;10.32 10.32;10.32 6.52;6.52 6.52;6.52 2.4;10.32 7.52;3.5 3.5;10.32 3.5;10.32 3.5;
        %31         32         33    34    35      36    37   38       39    40    41      42    43    44        45
        3.5 3.5;3.5 3.5;3.5 3;3 3;3 3.5;3.5 3;3 3;3 3.5;3.5 3;3 3;3 3.5;3.5 3;3 3;3 3.5;3.5 3.5;
        %46    47    48     49        50     51    52       53      54     55    56      57        58
        3.5 3;3 3;3 3.5;3.5 3.5;3.7 3;3 3;3 3.7;3.7 3.7;3.7 3;3 3;3 3.7;3.7 3.7;0.25 -0.6;
        %59           60               61                62             63                64          65            66           67
        -0.6 -0.6;-0.6 0.25;0.25 0.25;0.25 -0.6;-0.6 -0.6;-0.6 0.25;0.25 0.25;0.25 -0.6;-0.6 -0.6;
        %68              69              70        71
        -0.6 0.25;0.25 0.25;3.5 2.4;0 0.6]';

           %1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16   17   18
    p3 = [0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;
        %19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37
        0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;
        %38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55
        0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;
        %56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71
        0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0]';
```

รูปที่ 3.5 โค้ดที่ใช้จำลองตำแหน่งของกำแพงบนแกน Y และแกน Z บนโปรแกรม MATLAB

3.2.2 การเขียนโปรแกรมเพื่อจำลองการเคลื่อนที่ของคน 50 คน

เป็นการเขียนโปรแกรมจำลองการเคลื่อนที่ของคนจำนวน 50 คน เป็นการเขียนโปรแกรมสร้างสมการทางฟิสิกส์และคณิตศาสตร์ เพื่อใช้ในการบังคับทิศทางและการเคลื่อนที่ของคนให้ไปยังเป้าหมายที่กำหนดไว้ โดยได้ตั้งสมมติฐานไว้ว่า คนทุกคนจะวิ่งด้วยความเร็วเท่ากัน และมีทางออกเพียงทางเดียวเท่านั้น ตัวแปรต่างๆที่เกิดขึ้น มีดังต่อไปนี้

1. Pa คือ ตำแน่งของคนจำนวน 50 คน
2. Pb คือ ตำแหน่งของเป้าหมาย
3. Na คือ จำนวนคน 50 คน
4. Rep คือ แรงปฏิกิริยาของกำแพงที่กระทำต่อคน
5. Fatt คือ แรงดึงดูดของคนกับกำแพง
6. Fx คือ แรงที่ทำให้เกิดการเปลี่ยนทิศทางการเคลื่อนที่ เมื่อแรงดึงดูดกับแรงผลักมีค่าเท่ากัน
7. Ds คือ ระยะห่างระหว่างคนกับกำแพง เป็นค่าต่ำสุดที่คนไม่ควรเข้าใกล้กำแพงเกินไปกว่านั้น
8. Pt คือ ตำแหน่งของกำแพงแต่ละเส้นที่ใกล้กับคนแต่ละคนมากที่สุด

การเขียนโปรแกรมจำลองการเคลื่อนที่นี้มีการแบ่งขั้นตอนในการเขียน ดังนี้

3.2.2.1 โปรแกรมสร้างแรงดึงดูดระหว่างเป้าหมายกับคน 1 คน และแรงกระทำระหว่างกำแพงกับคน 1 คน

ขั้นตอนนี้ คือ การสร้างแรงดึงดูดระหว่างเป้าหมายกับคน 1 คน และแรงกระทำระหว่างกำแพงกับคน 1 คน ซึ่งมีการเขียนโปรแกรมดังนี้

rep คือ แรงปฏิกิริยาที่เกิดขึ้นระหว่างกำแพงกับคน เป็นการหาระยะห่างระห่างกำแพงแต่ละเส้นกับคน 1 คน และทำให้เป็นเวกเตอร์หนึ่งหน่วย ดังสมการต่อไปนี้

rep = (20*(-[pt(1,:)-pa(1) ; pt(2,:)-pa(2) ; pt(3':)-pa(3)])./[ds;ds;ds]); (1)

rep = rep(:,-isnan(rep(1,:).*double(abs(rep(1,:))<1000)) ~0; (2)

สมการที่ 1 แสดงการหาเวกเตอร์ 1 หน่วย ของระยะระหว่างคนกับกำแพงแต่ละเส้น

สมการที่ 2 แสดงการหาค่า rep ที่มีค่าเท่านั้นที่จะนำไปใช้ในการคำนวณในขั้นต่อไป

fatt คือ แรงดึงดูดระหว่างคนกับเป้าหมาย เป็นการหาระยะกระจัดระหว่างคนกับเป้าหมายเพื่อพาคนไปยังเป้าหมายที่กำหนด ดังสมการต่อไปนี้

fatt = 10*(pb-pa); (3)

nfatt = norm(fatt); (4)

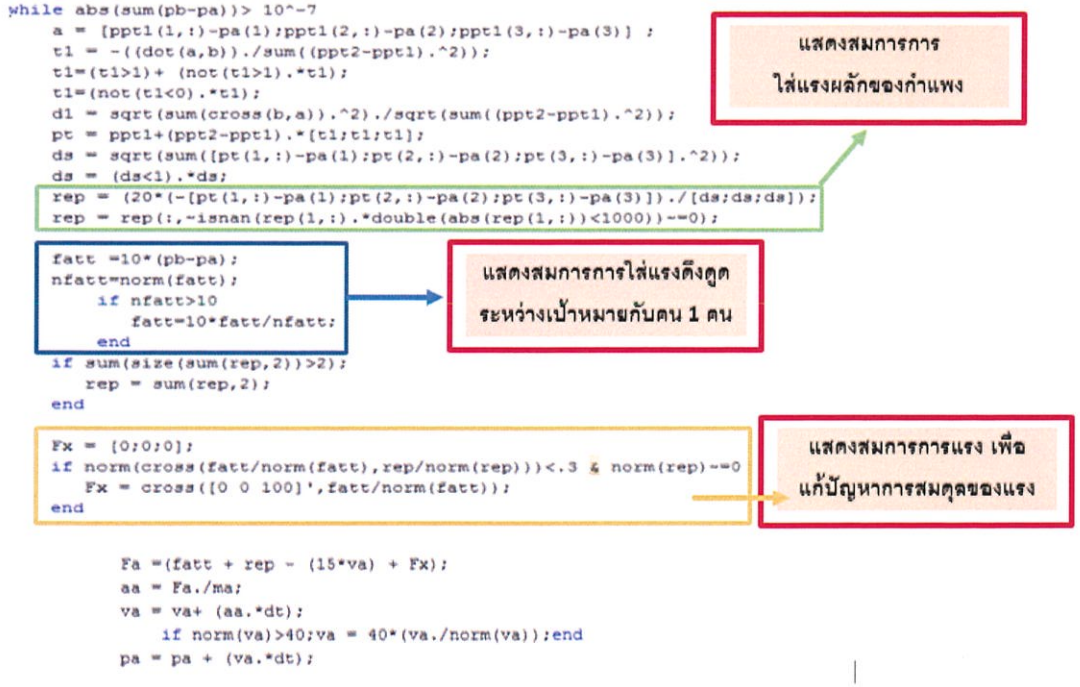
สมการที่ 3 แสดงการคำนวณแรงดึงดูดระหว่างคนกับเป้าหมาย

สมการที่ 4 แสดงการหาขนาดของ fatt

Fx คือ แรงสนับสนุนที่ทำให้เกิดการเปลี่ยนแปลงทิศทางการเคลื่อนที่ของคนไปในทิศทางอื่น

Fx = cross([0 0 100]',fatt/norm(fatt)); (5)

สมการที่ 5 แสดงการหาแรงสนับสนุนเพื่อการเปลี่ยนทิศทางการเคลื่อนที่ของคนเพื่อหลบหลีกสิ่งขีดขวาง ดังแสดงในรูปที่ 3.6



รูปที่ 3.6 โค้ดจำลองการเคลื่อนที่ของคน 1 คนและแรงกระทำที่เกิดขึ้น

จากโค้ดข้างต้น แสดงถึง สมการของแรงกระทำที่ใส่เข้าไปเพื่อให้เกิดการเคลื่อนที่ของคน เพื่อออกไปยังเป้าหมายที่กำหนด โดยเกิดจากการใส่แรงกระทำ 3 แรง คือ

1. สีเขียว แสดงแรงดึงดูดระหว่างคนกับเป้าหมาย
2. สีฟ้า แสดงแรงผลักของกำแพงเพื่อไม่ให้คนเดินทะลุผ่านกำแพง
3. สีเหลือง แรงเบี่ยงเบนทิศทาง เป็นแรงสนับสนุนให้เกิดการเคลื่อนที่ไปในทิศทางอื่น หากเมื่อมีแรงกระทำ 2 แรงข้างต้นที่เท่ากัน

3.2.2.2 <u>ขยายโปรแกรมเพื่อรองรับคนที่เพิ่มขึ้น จำนวน 50 คน</u>

ขั้นตอนนี้ คือ การขยายโปรแกรมเพื่อรองรับคนที่เพิ่มขึ้น จำนวน 50 คน มีหลักการเขียนโปรแกรมเหมือนดังขั้นตอนข้างต้น โดยมีการขยายเมตริกซ์เพื่อรองรับการเพิ่มขึ้นของคน ซึ่งมีการเขียนโปรแกรมดังนี้

rep1 คือ แรงปฏิกิริยาที่เกิดขึ้นระหว่างกำแพงกับคน เป็นการหาระยะห่างระห่างกำแพงแต่ละเส้นกับคนแต่ละคน และทำให้เป็นเวกเตอร์หนึ่งหน่วย ดังสมการต่อไปนี้

```
rep1 = (20*(-[pt(1,:)-pa((i-1)*3+1) ; pt(2,:)-pa((i-1)*3+2) ; pt(3':)-pa(i*3)])./[ds;ds;ds]);        (6)
rep1 = rep1(:,-isnan(rep(1,:).*double(abs(rep1(I,:))<1000)) =~0);                                    (7)
```

สมการที่ 6 แสดงการคำนวณแรงดึงดูดระหว่างคนแต่ละคนกับเป้าหมาย

สมการที่ 7 แสดงการหาค่า rep1 ที่มีค่าเท่านั้นที่จะนำไปใช้ในการคำนวณในขั้นต่อไป ดังแสดงในรูปที่ 3.7

Fat(:,i)t คือ แรงดึงดูดระหว่างคนกับเป้าหมาย เป็นการหาระยะกระจัดระหว่างคนกับเป้าหมายเพื่อพาคนไปยังเป้าหมายที่กำหนด ดังสมการต่อไปนี้

```
Fatt(:,i) = 10*(pb-pa(:,i));                                                                         (8)
nfatt = norm(fatt(:,i));                                                                             (9)
```

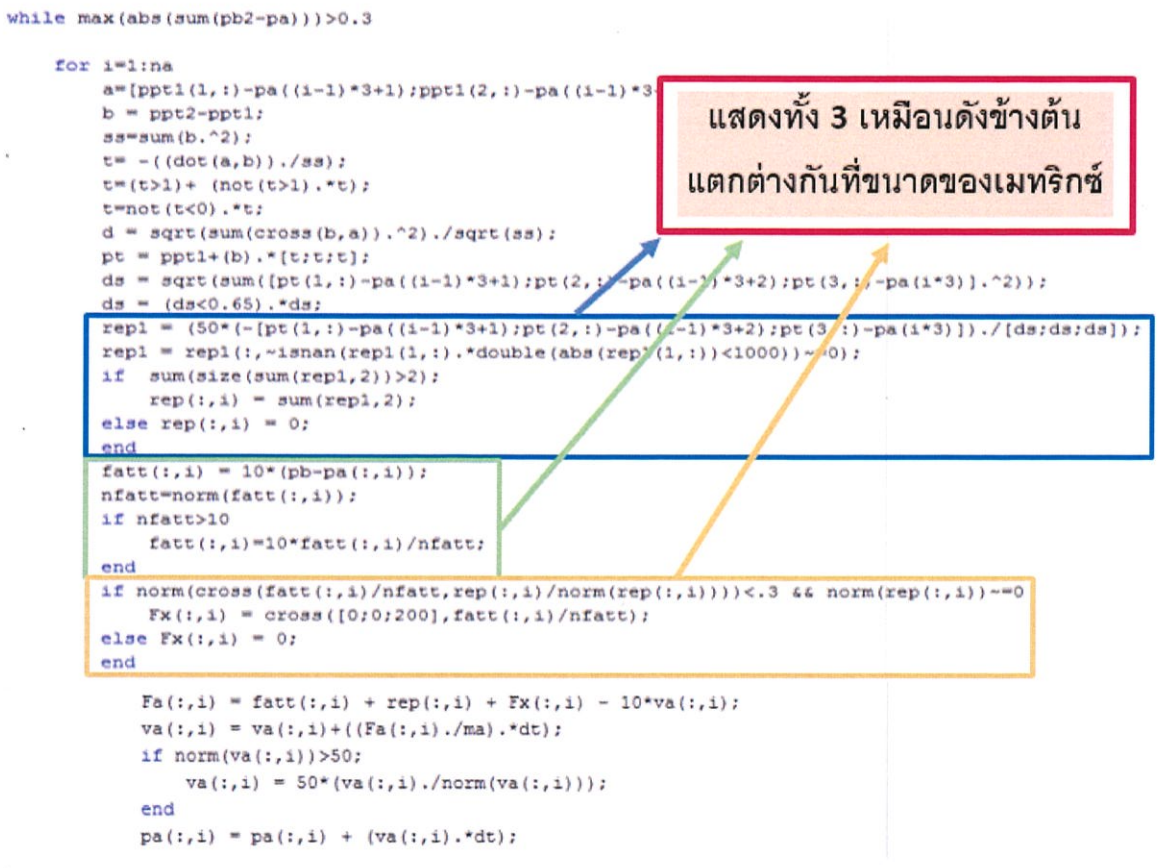
สมการที่ 3 แสดงการคำนวณแรงดึงดูดระหว่างคนแต่ละคนกับเป้าหมาย

สมการที่ 4 แสดงการหาขนาดของ fatt ของคนแต่ละคน

Fx(:,i) คือ แรงสนับสนุนที่ทำให้เกิดการเปลี่ยนแปลงทิศทางการเคลื่อนที่ของคนไปในทิศทางอื่น

```
Fx(:,i) = cross([0 0 100]',fatt(:,i)/norm(fatt));                                                    (10)
```

สมการที่ 5 แสดงการหาแรงสนับสนุนเพื่อการเปลี่ยนทิศทางการเคลื่อนที่ของคนเพื่อหลบหลีกสิ่งขีดขวาง ดังแสดงในรูปที่ 3.7




```
while max(abs(sum(pb2-pa)))>0.3

    for i=1:na
        a=[ppt1(1,:)-pa((i-1)*3+1);ppt1(2,:)-pa((i-1)*3
        b = ppt2-ppt1;
        ss=sum(b.^2);
        t= -((dot(a,b))./ss);
        t=(t>1)+ (not(t>1).*t);
        t=not(t<0).*t;
        d = sqrt(sum(cross(b,a)).^2)./sqrt(ss);
        pt = ppt1+(b).*[t;t;t];
        ds = sqrt(sum([pt(1,:)-pa((i-1)*3+1);pt(2,:)-pa((i-1)*3+2);pt(3,:)-pa(i*3)].^2));
        ds = (ds<0.65).*ds;
        rep1 = (50*(-[pt(1,:)-pa((i-1)*3+1);pt(2,:)-pa((i-1)*3+2);pt(3,:)-pa(i*3)])./[ds;ds;ds]);
        rep1 = rep1(:,~isnan(rep1(1,:).*double(abs(rep1(1,:))<1000))~=0);
        if  sum(size(sum(rep1,2)))>2);
            rep(:,i) = sum(rep1,2);
        else rep(:,i) = 0;
        end
        fatt(:,i) = 10*(pb-pa(:,i));
        nfatt=norm(fatt(:,i));
        if nfatt>10
            fatt(:,i)=10*fatt(:,i)/nfatt;
        end
        if norm(cross(fatt(:,i)/nfatt,rep(:,i)/norm(rep(:,i))))<.3 && norm(rep(:,i))~=0
            Fx(:,i) = cross([0;0;200],fatt(:,i)/nfatt);
        else Fx(:,i) = 0;
        end

            Fa(:,i) = fatt(:,i) + rep(:,i) + Fx(:,i) - 10*va(:,i);
            va(:,i) = va(:,i)+((Fa(:,i)./ma).*dt);
            if norm(va(:,i))>50;
                va(:,i) = 50*(va(:,i)./norm(va(:,i)));
            end
            pa(:,i) = pa(:,i) + (va(:,i).*dt);
```

รูปที่ 3.7 โค้ดจำลองการเคลื่อนที่ของคน 50 คนและแรงกระทำที่เกิดขึ้น

หากเปรียบเทียบสมการดังกล่าวกับสมการในรูปที่ 3.7 จะเห็นว่า มีการขยายรูปแบบการเขียนโปรแกรมให้มีลักษณะของเมตริกซ์ที่เพิ่มขึ้น จาก 1 แกนเป็น 2 แกน และมีการเพิ่ม For Loop เข้ามาในโปรแกรม เพื่อเป็นการอ้างอิงจำนวนตัวแปรที่เพิ่มขึ้น ในที่นี้ เพื่อรองรับคนจำนวน 50 คน ที่มีการเคลื่อนที่ใกล้เคียงกัน มีความเร็วที่เท่ากัน ขึ้นอยู่กับตำแหน่งต่างๆของคนแต่ละคนที่จะทำให้คนแต่ละคนมีทิศทางการเดินที่แตกต่างกัน

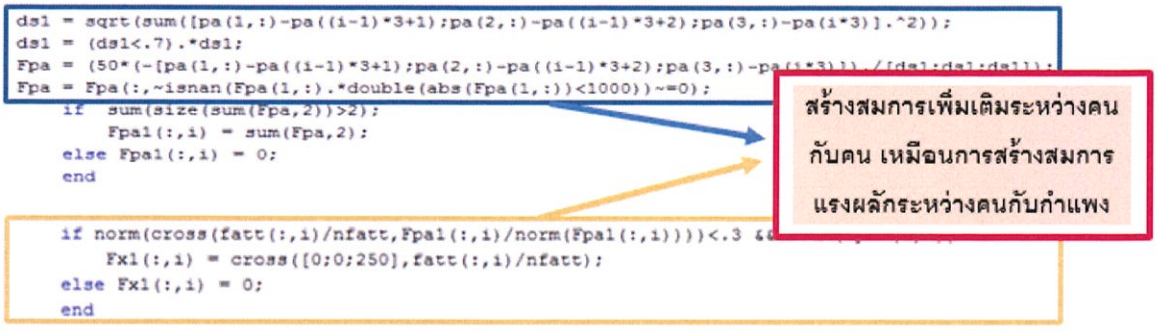
3.2.2.3 โปรแกรมสร้างแรงกระทำระหว่างคนแต่ละคน

ในการสร้างแรงกระทำระหว่างคนแต่ละคน เป็นการแก้ไขและป้องกันการซ้อนทับกันของคนแต่ละคนเพื่อให้เกิดความเสมือนจริงมากขึ้นในการจำลองสถานการณ์ต่างๆ โดยมีหลักการเขียนโปรแกรมดังต่อไปนี้

Ds1 คือ การหาระยะห่างระหว่างกำแพงต่อคนแต่ละคน

Fpa คือ แรงปฏิกิริยาระหว่างคนแต่ละคน

ทั้ง 2 ตัวแปรดังกล่าวเกิดจากการขยายโปรแกรมของขั้นตอนข้างต้น ซึ่งมีการเขียนโปรแกรมดังแสดงในรูปที่ 3.8



รูปที่ 3.8 โค้ดจำลองการเคลื่อนที่ของคน 50 คนและแรงกระทำที่เกิดขึ้น

ขั้นตอนสำหรับการสร้างแรงกระทำระหว่างคนกับคน ทำโดยการเพิ่มสมการแรงผลักเช่นเดียวกับการเสร้างแรงผลักของกำแพง จากรูปที่ 3.8 เพื่อให้เกิดแรงผลักที่สมจริงมากยิ่งขึ้น

3.3 การแสดงภาพกราฟิก

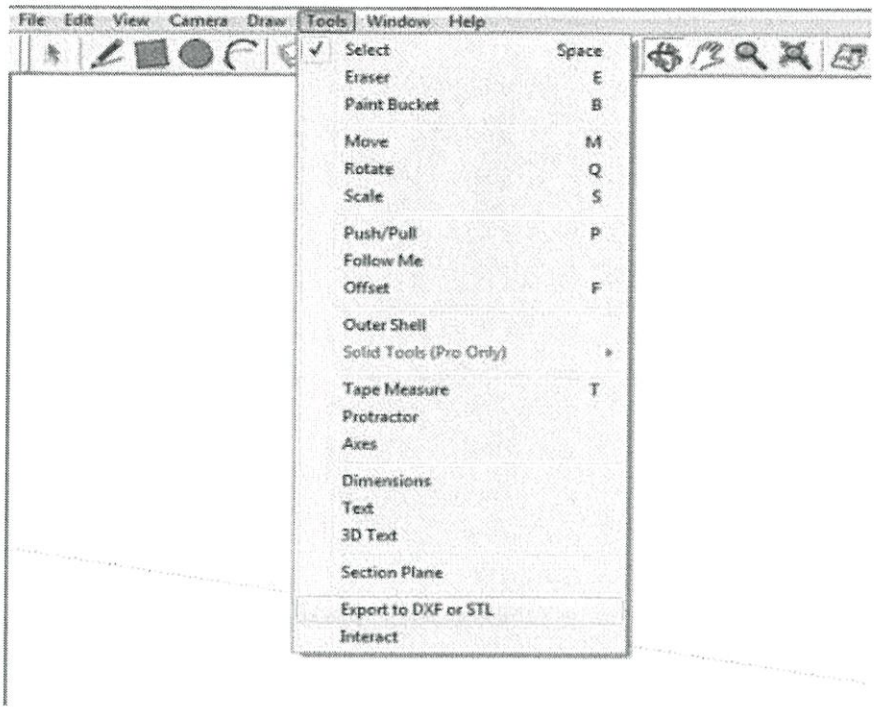
การใช้ฟังก์ชันแสดงรูปภาพทางเรขาคณิตพื้นฐานสร้างรูปภาพสามมิติขึ้นมา โดยภาพคนที่เราสร้างขึ้นเกิดจากการวาดรูปด้วยจุด 3 จุด และฟังก์ชันในไลบรารี OpenGL

3.3.1 การสร้างและแสดงภาพคนสามมิติ

ในการสร้างคนสามมิติ สามารถดาวน์โหลดไฟล์สำเร็จรูปจาก http://sketchup.google.com/3dwarehouse มาทำการแปลงไฟล์เป็นโค้ดภาพสามมิติสกุลไฟล์ STL หรือทำการสร้างคนสามมิติขึ้นมาเองผ่านโปรแกรมสร้างโมเดลสามมิติ เช่น Google Sketch Up, 3D Max, Maya, POSER เป็นต้น ในปริญญานิพนธ์ฉบับนี้ใช้การโหลดโมเดลสามมิติและทำการแปลงไฟล์ผ่านโปรแกรม Google Sketch Up

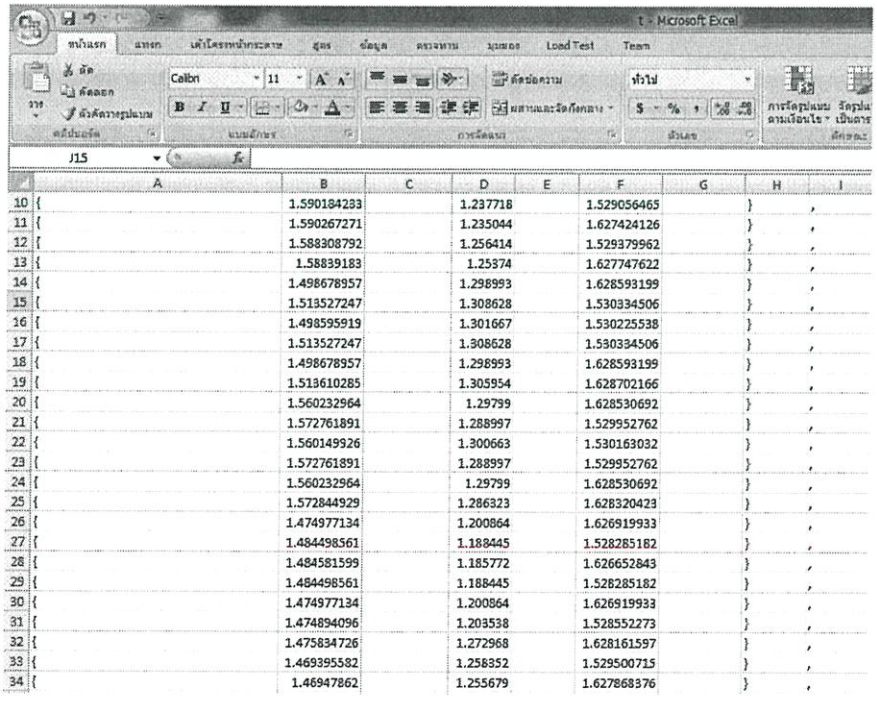
การแปลงไฟล์เป็นโค้ดโดยโดยผ่านคำสั่งบนซอฟแวร์ Google Sketch Up มีขั้นตอนดังนี้

Tools --> Export to DXF or STL --> เลือก Meters --> เลือก STL--> เซฟในโฟลเดอร์ที่เตรียมไว้



รูปที่ 3.9 การแปลงไฟล์ภาพสามมิติเป็นโค้ดสกุลไฟล์ STL

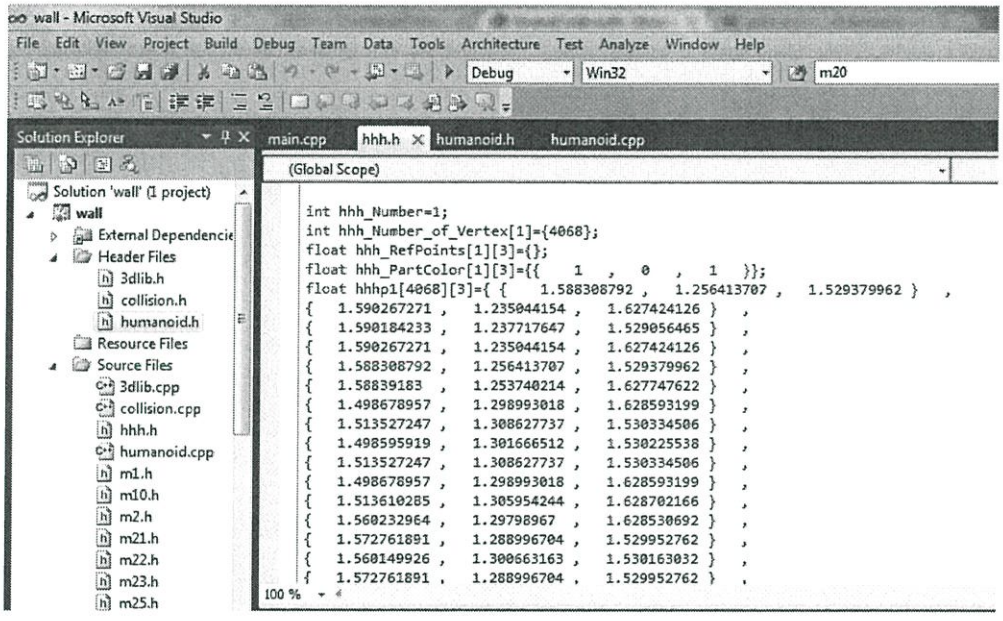
นำโค้ดมาทำการตัดส่วนที่เป็นข้อความที่ไม่เกี่ยวข้องออก ให้เหลือเพียงโค้ดตัวเลข และจัดเรียงตัวเลขเหล่านั้นให้อยู่ในรูปแบบที่พร้อมนำไปใช้งานต่อ ซึ่งโค้ดเหล่านี้จะอยู่ในรูปแบบสามมิติ บนแกน X แกน Y และแกน Z



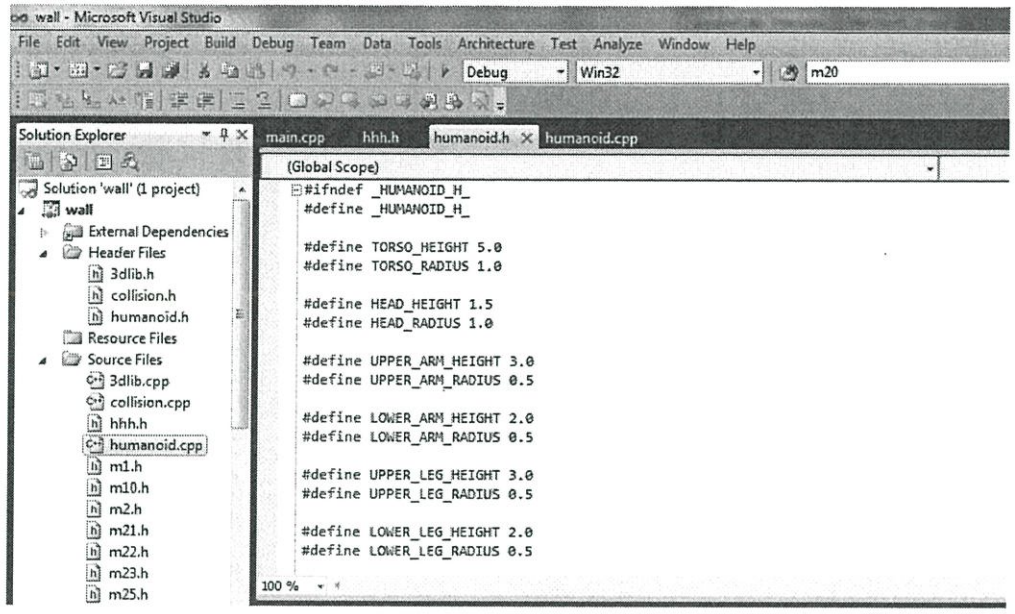
	A	B	C	D	E	F	G	H	I
10	{	1.590184233		1.237718		1.529056465		}	,
11	{	1.590267271		1.235044		1.627424126		}	,
12	{	1.588308792		1.256414		1.529379962		}	,
13	{	1.58839183		1.25374		1.627747622		}	,
14	{	1.498678957		1.298993		1.628593199		}	,
15	{	1.513527247		1.308628		1.530334506		}	,
16	{	1.498595919		1.301667		1.530225538		}	,
17	{	1.513527247		1.308628		1.530334506		}	,
18	{	1.498678957		1.298993		1.628593199		}	,
19	{	1.513610285		1.305954		1.628702166		}	,
20	{	1.560232964		1.29799		1.628530692		}	,
21	{	1.572761891		1.288997		1.529952762		}	,
22	{	1.560149926		1.300663		1.530163032		}	,
23	{	1.572761891		1.288997		1.529952762		}	,
24	{	1.560232964		1.29799		1.628530692		}	,
25	{	1.572844929		1.286323		1.628320423		}	,
26	{	1.474977134		1.200864		1.626919933		}	,
27	{	1.484498561		1.188445		1.528285182		}	,
28	{	1.484581599		1.185772		1.626652843		}	,
29	{	1.484498561		1.188445		1.528285182		}	,
30	{	1.474977134		1.200864		1.626919933		}	,
31	{	1.474894096		1.203538		1.528552273		}	,
32	{	1.475834726		1.272968		1.628161597		}	,
33	{	1.469395582		1.258352		1.529500715		}	,
34	{	1.46947862		1.255679		1.627868376		}	,

รูปที่ 3.10 โค้ดที่ผ่านการจัดเรียงเพื่อนำไปใช้งานต่อไป

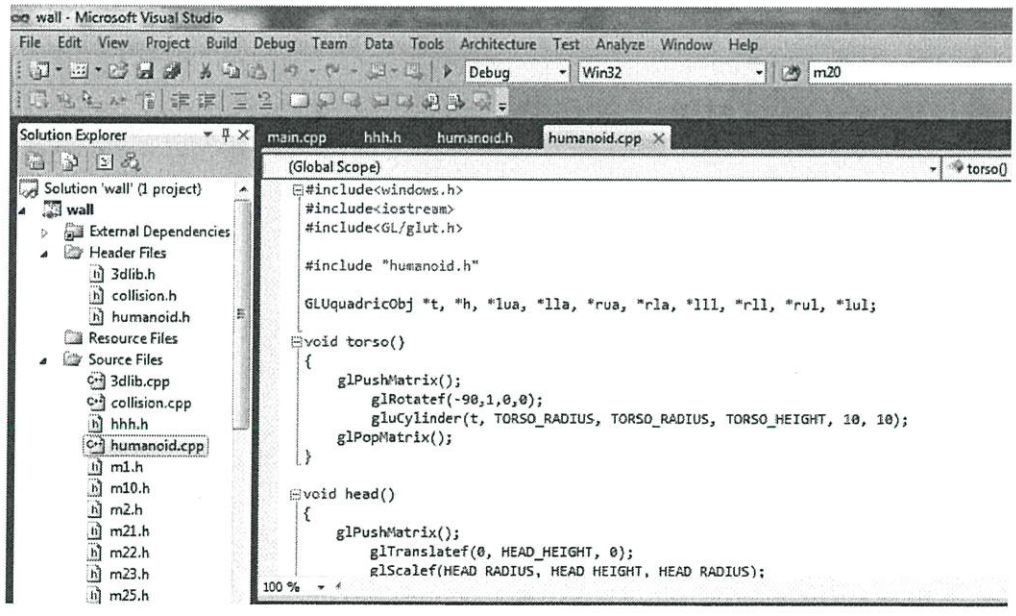
นำข้อมูลที่ได้มาเขียนบนโปรแกรม Microsoft Visual Studio C++ ในโปรแกรมจำลองสภาพแวดล้อมเสมือนจริงจากผู้วิจัยก่อนหน้า โดยใช้ฟังก์ชันสร้างรูปภาพทางเรขาคณิตพื้นฐาน GL_TRIANGLES เพราะเกิดจากจุด 3 จุดบนแกนทั้งสาม สร้างเป็นไฟล์ Header และชุดข้อมูลของคนขึ้นมา เพื่อทำให้ปรากฏภาพคนสามมิติและสภาพแวดล้อมเสมือนจริงบนหน้าจอโปรแกรมที่ถูกสร้างขึ้น พร้อมกำหนดตำแหน่งและเส้นทางเดินของคน ซึ่งเป็นผลลัพธ์จากการประมวลผลผ่านโปรแกรม MATLAB ดังแสดงในรูปที่ 3.11 รูปที่ 3.12 (ก) และ รูปที่ 3.12 (ข)



รูปที่ 3.11 ไฟล์ Header หลังจากถูกสร้างด้วยชุดข้อมูลในรูปที่ 3.10



(ก.)



(ข.)

รูปที่ 3.12 ภาพ ก. และภาพ ข. ชุดข้อมูลของคนที่ใช้เพื่อให้เกิดการติดต่อกันระหว่างโปรแกรมจำลองสภาพแวดล้อมเสมือนจริงกับชุดข้อมูลของวัตถุสามมิติ

บทที่ 4
ผลการศึกษาและทดลอง

4.1 ศึกษาทฤษฎีที่เกี่ยวข้องกับปริญญานิพนธ์

ผู้วิจัยได้ทำการศึกษาข้อมูลและทฤษฎี เพื่อใช้ในการเขียนโปรแกรมจำลองสถานการณ์เสมือนจริงผ่านแว่นตาสามมิติเพื่อการจัดการความปลอดภัย โดยศึกษาจากผู้เชี่ยวชาญ หนังสือ และอินเทอร์เน็ต เพื่อนำข้อมูลต่างๆ เหล่านั้นมาเขียนโปรแกรมให้ถูกต้อง สมบูรณ์ โดยได้รวบรวมไว้ในบทที่ 2

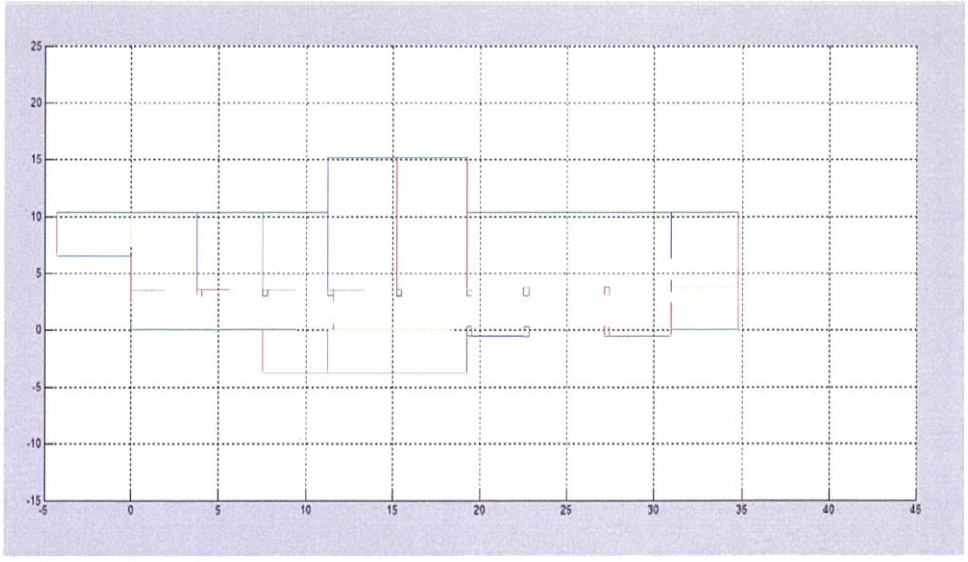
4.2 เขียนโปรแกรมจำลองการเคลื่อนที่ของคนภายในสภาพแวดล้อมที่กำหนด

ผลการศึกษาและทดลองการเขียนโปรแกรมจำลองการเคลื่อนที่ของคนภายในสภาพแวดล้อมที่กำหนดนั้น มีด้วยกัน 3 หัวข้อใหญ่ๆ คือ

1. การสร้างของเขตของสภาพแวดล้อมเสมือนจริง
2. การเขียนโปรแกรมเพื่อจำลองการเคลื่อนที่ของคน 50 คน
3. การแสดงภาพกราฟิก ซึ่งได้ผลการทดลองดังต่อไปนี้

4.2.1 การสร้างขอบเขตของสภาพแวดล้อมเสมือนจริง

จากการวัดขนาดจากโรงปฏิบัติการทางวิศวกรรม คณะวิศวกรรมศาสตร์ สาขาวิศวกรรมอุตสาหการ และสร้างเป็นภาพจำลองสองมิติขึ้นในโปรแกรม MATLAB ได้ภาพออกมาเป็นไป ดังแสดงในรูปที่ 4.1



รูปที่ 4.1 ขอบเขตจำลองโดยโปรแกรม MATLAB ในหน่วยเมตร

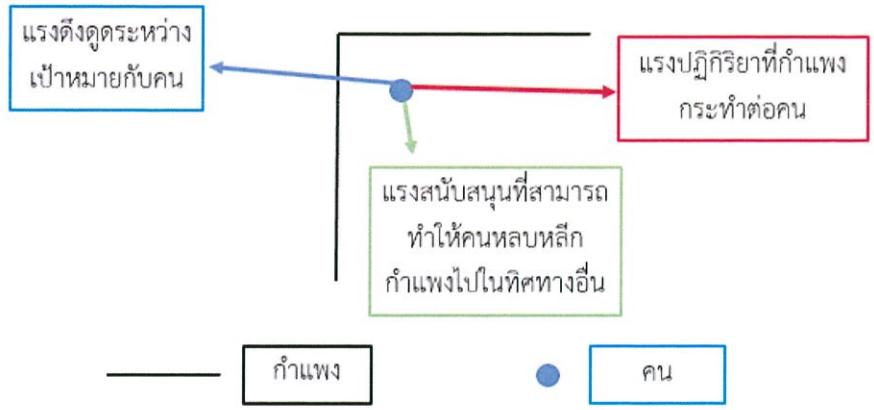
4.2.2 การเขียนโปรแกรมเพื่อจำลองการเคลื่อนที่ของคน 50 คน

ในขั้นตอนนี้ แบ่งขั้นตอนการเขียนโปรแกรมเป็น 3 ขั้นตอนด้วยกัน คือ

1. โปรแกรมสร้างแรงดึงดูดระหว่างเป้าหมายกับคน 1 คน และแรงกระทำระหว่างกำแพงกับคน 1 คน
2. ขยายโปรแกรมเพื่อรองรับคนที่เพิ่มขึ้น จำนวน 50 คน
3. โปรแกรมสร้างแรงกระทำระหว่างคนแต่ละคน ซึ่งผลการทดลองมีดังต่อไปนี้

4.2.2.1 <u>โปรแกรมสร้างแรงดึงดูดระหว่างเป้าหมายกับคน 1 คน และแรงกระทำระหว่างกำแพงกับคน 1 คน</u>

ภายหลังการสร้างแรงดึงดูดระหว่างเป้าหมายกับคน 1 คน และแรงกระทำระหว่างกำแพงกับคน 1 คน จะเกิดแรงกระทำขึ้น 3 แรงด้วยกัน คือ 1. แรงดึงดูดระหว่างเป้าหมายกับคน 2. แรงปฏิกิริยาที่กำแพงกระทำต่อคน 3.แรงสนับสนุนที่สามารถทำให้คนหลบหลีกกำแพงในทิศทางอื่น ดังแสดงในรูปที่ 4.2

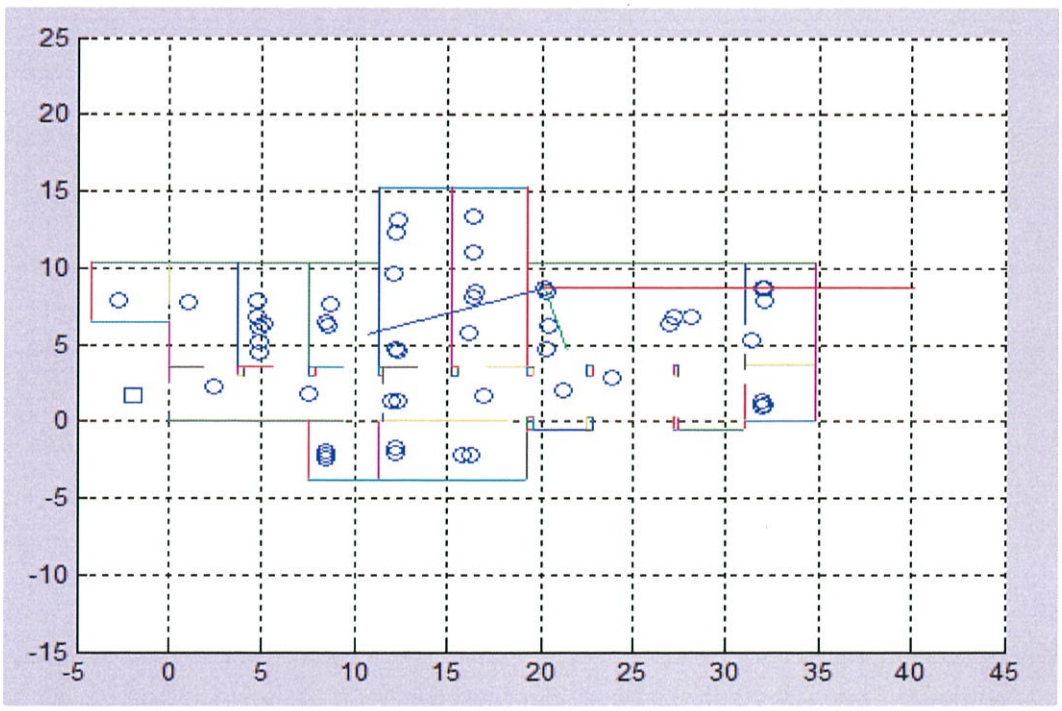


รูปที่ 4.2 แรงดึงดูดและแรงกระทำระหว่างคนกับกำแพง

จากภาพแสดงถึงแรงกระทำ 3 แรงที่เกิดขึ้นต่อคนเมื่อเกิดการเคลื่อนที่ ซึ่งแรงทั้ง 3 แรง จะถูกนำเสนอในรูปแบบของเส้นตรงสีต่างๆดังนี้ เส้นสีน้ำเงิน แสดงถึงแรงดึงดูดระหว่างเป้าหมายกับคน เส้นสีแดง แสดงถึงแรงปฏิกิริยาที่กำแพงกระทำต่อคน เส้นสีเขียว แสดงถึงแรงสนับสนุนที่สามารถทำให้คนหลบหลีกกำแพงไปในทิศทางอื่นได้

4.2.2.2 ขยายโปรแกรมเพื่อรองรับคนที่เพิ่มขึ้น จำนวน 50 คน

ภายหลังการขยายโปรแกรมเพื่อรองรับคนที่เพิ่มขึ้น จำนวน 50 คน เป็นการขยายโปรแกรมจากการสร้างโปรแกรมสร้างแรงดึงดูดระหว่างเป้าหมายกับคน 1 คน และแรงกระทำระหว่างกำแพงกับคน 1 คน เกิดเป็นการเคลื่อนที่ของคนจำนวน 50 คน ดังแสดงในรูปที่ 4.3

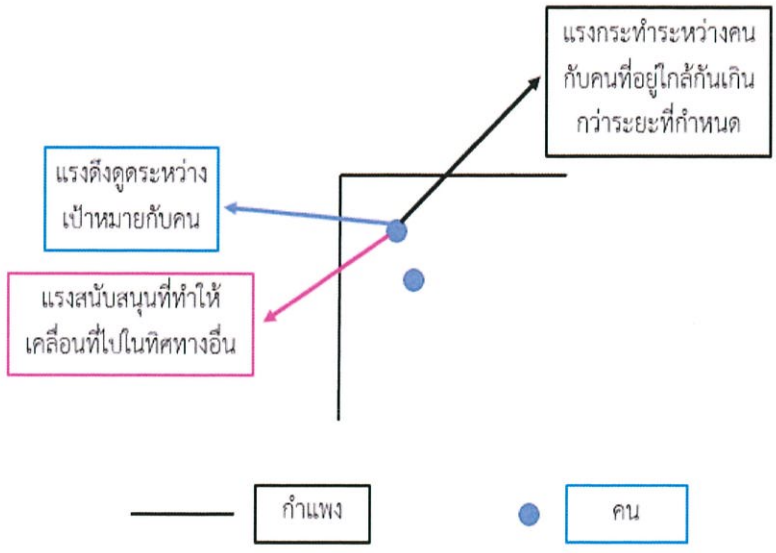


รูปที่ 4.3 คนจำนวน 50 คนในสภาพแวดล้อมที่กำหนด

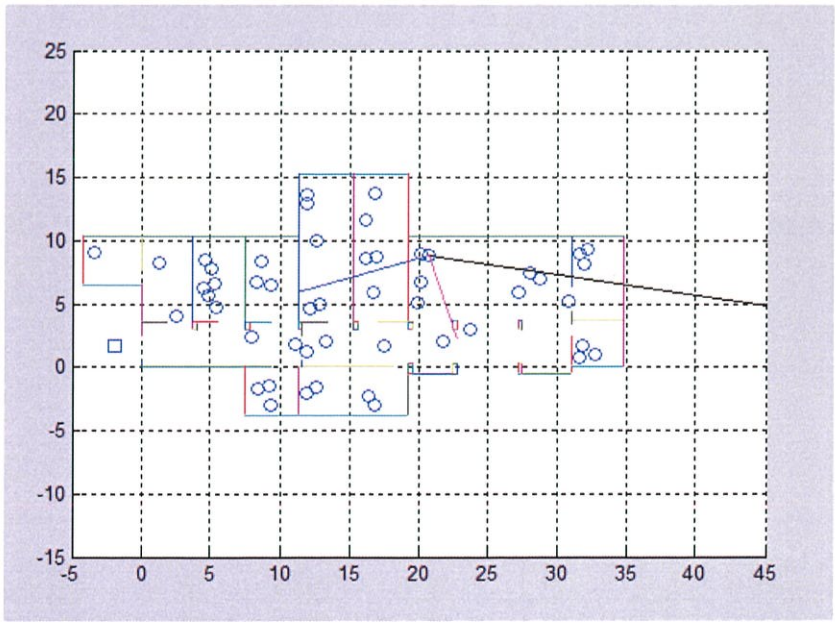
จากภาพแสดงให้เห็นการเคลื่อนที่ของคนจำนวน 50 คน ภายในสภาพแวดล้อมที่กำหนด ที่ได้ทำการสร้างแรงกระทำทั้ง 3 แรง ดังที่กล่าวไปแล้วในข้างต้น แต่จะเห็นว่าในภาพยังคงเกิดปัญหาคนที่ทับซ้อนกัน ในขั้นตอนต่อไปเป็นการแก้ไขปัญหานี้ เพื่อให้เกิดความเสมือนจริงมากยิ่งขึ้น

4.2.2.3 <u>โปรแกรมสร้างแรงกระทำระหว่างคนแต่ละคน</u>

ภายหลังการเขียนโปรแกรมสร้างแรงกระทำระหว่างคนแต่ละคน เป็นการแก้ไขปัญหาการซ้อนทับของคนแต่ละคน ในขั้นตอนนี้จะเกิดแรงกระทำที่เพิ่มขึ้น คือ 1. แรงกระทำระหว่างคนกับคนที่อยู่ใกล้กันเกินกว่าระยะที่กำหนด 2. แรงผลักที่ผลักคนที่อยู่ใกล้กันให้แยกออกจากกัน แสดงดังรูปที่ 4.4 และรูปที่ 4.5



รูปที่ 4.4 แรงกิริยาและแรงปฏิกิริยาที่เกิดขึ้นทั้งหมด (ก)



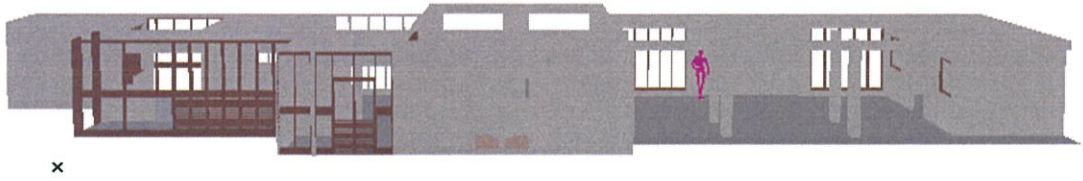
รูปที่ 4.5 แรงกิริยาและแรงปฏิกิริยาที่เกิดขึ้นทั้งหมด (ข)

จากภาพจะเห็นว่าไม่เกิดการทับซ้อนกันของคนดังแสดงในรูปที่ 4.2 จากภาพจะปรากฏเส้นสีดำและสีชมพูเพิ่มเติมมา ซึ่งเส้นสีดำ แสดงถึงแรงกระทำระหว่างคนกับคนที่อยู่ใกล้กันเกินกว่าระยะที่กำหนด เส้นสีชมพู แสดงถึงแรงผลักที่ผลักคนที่อยู่ใกล้กันให้แยกออกจากกัน โดยจะนำเสนอให้เห็นเพียงคนเดียว เนื่องจากหากใส่เส้นสีต่างๆกับคนทุกคนแล้วจะทำให้ภาพลายและไม่สามารถตีความหมายของแรงที่เกิดขึ้นได้ และจากการเขียนโปรแกรมในขั้นตอนดังกล่าวยังมีข้อผิดพลาดอยู่บางประการ คือ ยังคงมีคนบางส่วนติดอยู่ในห้องและไม่สามารถออกมาได้ เนื่องด้วยการเขียนโปรแกรมในทิศทางเดียว แต่ทางคณะผู้จัดทำได้ทำการคิดหาแนวทางในการแก้ไขปัญหาดังกล่าวไว้แล้ว

4.2.3 การแสดงภาพกราฟิก

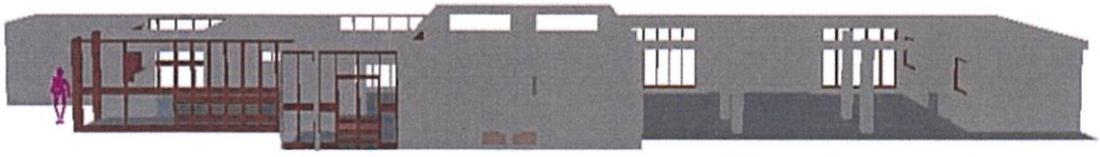
4.2.3.1 <u>การแสดงภาพกราฟิกของคนจำนวนหนึ่งคน</u>

แสดงภาพกราฟิกของคนจำนวนหนึ่งคนผ่านมุมมองต่างๆดังแสดงในรูปที่ 4.6 รูปที่ 4.7 และรูปที่ 4.8



รูปที่ 4.6 มุมมองด้านหน้าของผลลัพธ์

รูปที่ 4.7 มุมมองด้านอื่นๆ ของผลลัพธ์

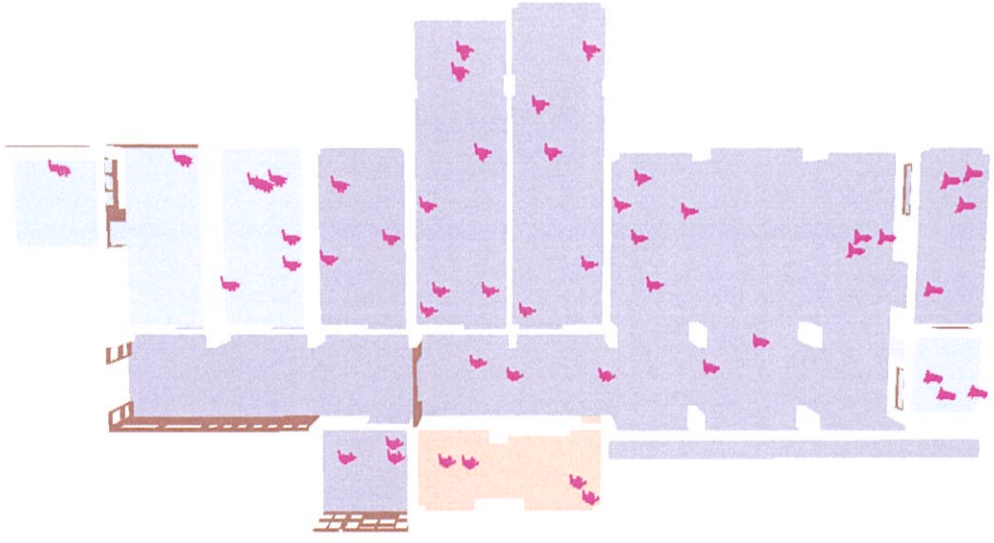


รูปที่ 4.8 มุมมองด้านหน้าของผลลัพธ์หลังสิ้นสุดการเคลื่อนที่

ภาพแสดงผลลัพธ์จากการเขียนโปรแกรมสร้างคนสามมิติในสภาพแวดล้อมที่กำหนด เป็นการแสดงผลลัพธ์รวมกับโปรแกรมของผู้วิจัยก่อนหน้า และทำให้คนเคลื่อนที่จากตำแหน่งปัจจุบันไปยังตำแหน่งกากบาทข้างซ้ายมือของโรงปฏิบัติการทางวิศวกรรม ในรูปที่ 4.5 โดยใช้ผลลัพธ์จากการประมวลผลทางโปรแกรม MATLAB เป็นตัวกำหนดเส้นทางการเคลื่อนที่ของคน

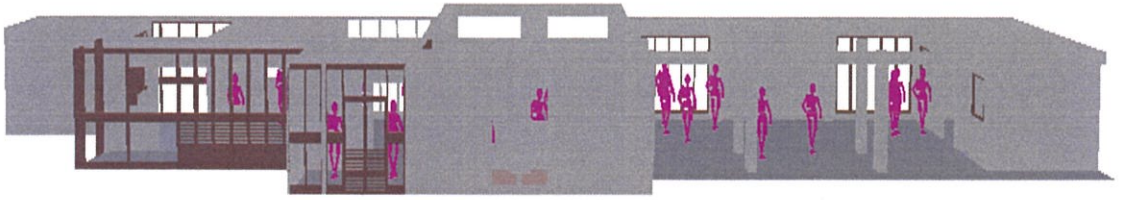
4.2.3.2 <u>การแสดงภาพกราฟิกของคนจำนวน 50 คน</u>

รูปที่ 4.9 ภาพแสดงคนจำนวน 50 ภายใน โรงปฏิบัติการทางวิศวกรรม โดยมีการอ้างอิงตำแหน่งตามการเขียนโปรแกรมบนโปรแกรม MATLAB แสดงออกมาในรูปแบบของภาพ สามมิติบนโปรแกรม Visual Studio C++ ในมุมมองด้านบน



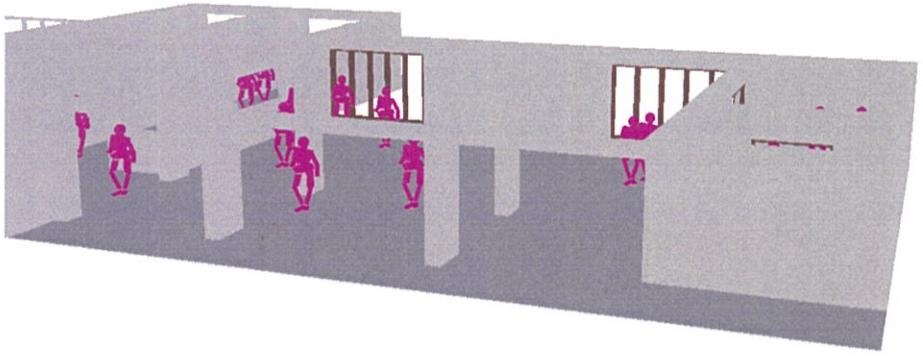
รูปที่ 4.9 มุมมองด้านบนของผลลัพธ์

จากรูปที่ 4.10 ภาพแสดงตำแหน่งของคน จำนวน 50 ในมุมมองด้านหน้า ในตำแหน่งเดียวกับบนโปรแกรม MATLAB ซึ่งแสดงถึงบริเวณทั้งหมดของโรงปฏิบัติการทางวิศวกรรม



รูปที่ 4.10 มุมมองด้านหน้าของผลลัพธ์

จากรูปที่ 4.11 ภาพแสดงตำแหน่งของคน จำนวน 50 ในมุมมอง Isolated แสดงภาพบริเวณพื้นที่ของ Machining ซึ่งจะมีขนาดใหญ่และมีผู้ทำงานในบริเวณนี้มากที่สุด ซึ่งมีโอกาสของการเกิดอุบัติเหตุได้มากที่สุดในโรงปฏิบัติการทางวิศวกรรม



รูปที่ 4.11 มุมมองแบบ Isolated ของผลลัพธ์

บทที่ 5
สรุปผลการดำเนินงาน

5.1 สรุปผลการดำเนินงาน

ปริญญานิพนธ์นี้จัดทำขึ้นเพื่อสร้างต้นแบบโปรแกรมจำลองสถานการณ์เสมือนจริงผ่านแว่นตาสามมิติ เพื่อการจัดการ ความปลอดภัย มีจุดมุ่งหมายที่สำคัญ คือ การออกแบบการจัดการความปลอดภัย โดยโปรแกรมถูกสร้างขึ้นผ่านโปรแกรม MATLAB ซึ่งจะทำการจำลองสถานการณ์การเคลื่อนที่ออกจากโรงปฏิบัติการภายใต้สถานการณ์สมมุติในรูปแบบของภาพ 2 มิติ ของคนจำนวน 50 คน ซึ่งมีการเคลื่อนที่แบบสุ่มเพื่อหาทางออกที่ใกล้ที่สุด จากนั้นได้ทำการนำเสนอในรูปแบบของภาพจำลอง ผ่าน Visual C++ บนโปรแกรม Visual Studio 2010 เพื่อแสดงเป็น ภาพสามมิติ โดยใช้ Open Graphics Library เป็นตัวช่วยประมวลผลภาพให้แสดงผลบนจอของคอมพิวเตอร์และแว่นตาสามมิติได้

ผลลัพธ์ที่ได้จากโครงการ คือ สามารถแสดงภาพการเคลื่อนที่ของคนภายในโรงงานหรืออาคารต่างๆเพื่อออกไปยังเป้าหมายในกรณีที่เกิดอุบัติเหตุหรืออุบัติภัย โดยมีการเคลื่อนที่ไปยังเป้าหมายอย่างอิสระภายใต้การเขียนโปรแกรมให้สามารถหลบหลีกสิ่งขีดขวางหรือสิ่งขวางกั้นต่างๆ ซึ่งโปรแกรมที่เขียนขึ้นนั้นยังสามารถนำไปวิเคราะห์ข้อมูลทางสถิติถึงเวลาที่ใช้ในการเคลื่อนที่ออกไปยังเป้าหมาย เพื่อนำไปวิเคราะห์หาแนวทางในการแก้ไขหรือปรับเปลี่ยนโครงสร้างโรงงานหรืออาคารต่างๆเหล่านั้น รวมถึงในกรณีการเกิดอัคคีภัย ที่มีระยะเวลาในการเคลื่อนย้ายออกจากตัวอาคารอย่างจำกัด โดยโปรแกรมที่สร้างขึ้นจะสามารถบอกถึงโอกาสการหนีรอดของคนภายในสภาพแวดล้อมดังกล่าว เพื่อเป็นแนวทางในการป้องกันความสูญเสียต่างๆที่อาจเกิดได้เป็นอย่างดี แต่ด้วยข้อจำกัดทางเวลาของการจัดทำโครงการ ทำให้ยังไม่สามารถแปลงข้อมูลบน MATLAB ไปเป็น Visual C++ ได้ทั้งหมด แต่สามารถนำไปทำการปรับปรุงพัฒนาโปรแกรมต่อไปได้ในอนาคต

5.2 ข้อเสนอแนะ

ทางคณะผู้จัดทำมีแนวทางที่จะดำเนินงานเพื่อพัฒนาและปรับปรุงโปรแกรมการจำลองการเคลื่อนที่ของคนจำนวน 50 คน ดังนี้

1. ควรทำการเพิ่มเติมส่วนที่เป็นการสุ่มการเคลื่อนที่ของคนให้เป็นไปตามธรรมชาติมากกว่านี้
2. วิธีแก้ปัญหาการติดของคนที่มุมห้องต่างๆ สามารถทำได้โดยการเพิ่มจุดเป้าหมายตามประตูของห้องต่างๆ เพื่อให้คนสามารถเดินออกตามเส้นทางที่กำหนดไว้ได้
3. ควรเพิ่มเติมในส่วนของเครื่องจักรหรืออุปกรณ์ต่างๆภายในโรงปฏิบัติการให้มากกว่านี้ เพื่อให้เกิดความสมจริงมากยิ่งขึ้น และมีประโยชน์ต่อการคาดการณ์ของสถานการณ์จำลองได้ดียิ่งขึ้น

เอกสารอ้างอิง

[1] มนัส สังวรศิลป์ และวรรัตน์ ภัทรอมรกุล. 2543. คู่มือการใช้งาน MATLAB ฉบับสมบูรณ์. นนทบุรี: อินโฟเพรส

[2] Programming Systems Laboratory. Screenshots. 2011. Available: http://psl.cs.columbia.edu/chime. November 19, 2013.

[3] เมทริกซ์ (คณิตศาสตร์). 2556. เข้าถึงได้จาก: http://th.wikipedia.org. สืบค้นวันที่ 19 พฤศจิกายน 2556

[4] Sutee Kawsongsee. 2556. การคำนวณเมตริกซ์ด้วย MATLAB. เข้าถึงได้จาก: http://matabthai.blogspot.com/2013/05/5-matlab_25.html. สืบค้นวันที่: 19 พฤศจิกายน 2556

[5] คณรักษ์ วงศ์ชูศิริ, 2556. การเขียนเวกเตอร์. เข้าถึงได้จาก: http://www.tewlek.com. สืบค้นวันที่: 19 พฤศจิกายน 2556

[6] ดร.อรรถกฤต ฉัตรภูติ. 2556. ผลคูณของเวกเตอร์ 1 หน่วย. เข้าถึงได้จาก: http://www.rmutphysics.com. สืบค้นวันที่: 19 พฤศจิกายน 2556

[7] นางลักษมน เทียมลม. 2553. การเขียนโปรแกรม. เข้าถึงได้จาก: http://neung.kaengkhoi.ac.th. สืบค้นวันที่: 19 พฤศจิกายน 2556

[8] มนตรี พิรุณเกษตร, 2550. กลศาสตร์วิศวกรรม ภาคพลศาสตร์. กรุงเทพฯ: วิทยพัฒน์

[9] กฎการเคลื่อนที่ของนิวตัน. 2555. เข้าถึงได้จาก: http://www.kmitl.ac.th. สืบค้นวันที่: 19 พฤศจิกายน 2556

[10] action-reaction. 2556. เข้าถึงได้จาก: http://www.dichotomistic.com. สืบค้นวันที่: 19 พฤศจิกายน 2556

[11] Weisstein, Eric W. 1999-2014. Distance between point to line formular. Available: http://www.mathworld.wolfram.com. November 19, 2013.

[12] ไพศาล โมลิสกุลมงคล. 2550. คอมพิวเตอร์กราฟิกส์ใช้ OpenGL (Computer Graphics using OpenGL). กรุงเทพฯ: ไทยเจริญการพิมพ์

ภาคผนวก ก

Source Code

MATLAB CODE

```
clear all;clc;
tic
p1 = [0 0;0 9.45;7.5 7.5;7.5 19.25;11.25 11.25;11.25 18.35;19.25 19.25;19.25 22.8;27.15 31;31 31;31
34.75;34.75 34.75;31 34.75;31 31;31 31;34.75 19.25;19.25 19.25;11.25 19.25;15.25 15.25;17.05 19.25;11.25
13.45;11.25 11.25;4.3 11.25;-4.3 -4.3;-4.3 0;0 0;0 0;0 1.95;3.75 3.75;7.5 7.5;3.75 5.7;7.5 9.45;3.75 3.75;3.75
4.05;4.05 4.05;7.5 7.5;7.5 7.8;7.8 7.8;11.25 11.25;11.25 11.55;11.55 11.55;15.25 15.25;15.25 15.55;15.55
15.55;15.55 15.25;19.25 19.25;19.25 19.55;19.55 19.55;19.55 19.25;22.5 22.5;22.5 22.8;22.8 22.8;22.8
22.5;27.15 27.15;27.15 27.45;27.45 27.45;27.45 27.15;19.25 19.25;19.25 19.55;19.55 19.55;19.55 19.25;22.5
22.5;22.5 22.8;22.8 22.8;22.8 22.5;27.15 27.15;27.15 27.45;27.45 27.45;27.45 27.1511.55 11.55;11.55 11.55]';

p2 = [0 .6;0 0;0 -3.8;-3.8 -3.8;0 -3.8;0 0;0 -3.8;-0.6 -0.6;-0.6 -0.6;2.37 -0.6;0 0;0 10.32;3.61 3.61;4.37 3.27;10.32
6.17;10.32 10.32;15.2 3.5;15.2 15.2;15.2 3.5;3.5 3.5;3.5 3.5;15.2 3.5;10.32 10.32;10.32 6.52;6.52 6.52;6.52
2.4;10.32 7.52;3.5 3.5;10.32 3.5;10.32 3.5;3.5 3.5;3.5 3.5;3.5 3;3 3;3 3.5;3.5 3;3 3;3 3.5;3.5 3;3 3;3 3.5;3.5 3;3
3;3 3.5;3.5 3.5;3.5 3;3 3;3 3.5;3.5 3.5;3.7 3;3 3;3 3.7;3.7 3.7; 3.7 3;3 3;3 3.7;3.7 3.7;0.25 -0.6;-0.6 -0.6;-0.6
0.25;0.25 0.25;0.25 -0.6;-0.6 -0.6;-0.6 0.25;0.25 0.25;0.25 -0.6;-0.6 -0.6;-0.6 0.25;0.25 0.25;3.5 2.4;0 0.6]';

p3 = [0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0 0;0
0;0 0;0 0;0 0;0 0; 0 0; 0 0; 0 0; 0 0; 0 0; 0 0; 0 0; 0 0;0 0; 0 0; 0 0; 0 0; 0 0; 0 0; 0 0; 0 0; 0 0; 0 0; 0 0; 0
0; 0 0; 0 0; 0 0; 0 0; 0 0; 0 0; 0 0; 0 0,0 0; 0 0; 0 0; 0 0; 0 0; 0 0; 0 0; 0 0; 0 0; 0 0;0 0]';

ppt1 = [p1(1,:) ; p2(1,:); p3(1,:)];
ppt2 = [p1(2,:); p2(2,:); p3(2,:)];
pa = [22 9 0;15 3 0;11 3 0;19 9 0;7 5 0;6 6 0;20 7 0;5 9 0;12 5 0;3 4 0;13 1 0;6 7 0;14 5 0;10 7 0;15 3 0;5 8
0;11 7 0;16 12 0;16 9 0;18 6 0; 25 3 0;32 5 0;34 1 0;29 7 0;13 14 0;23 2 0;30 7 0;18 14 0;20.1 9.25 0;28.75
6.5 0;    32.5 9.13 0;33.15 8.2 0;32.13 1.6 0;18.75 1.7 0;32.7 0.92 0;20.65 5.21 0;8 9 0;10.61 -2.71 0;10.56 -
2.17 0;8.11 -1.5 0;12.21 -1.65 0;13.17 -1.7 0;18.12 -3.1 0;17.6 -2.48 0;33.4 9.4 0;12.81 13.21 0;13.72 10.22 0;-
3.23 9.61 0;2.22 8.76 0;4.17 7.89 0]';



pb = [-2 1.725 0]';
ma = 10;
na=50;
dt = 0.01;
tmax=100;
nt=tmax/dt;
```

```
mt = 20;
mb = 20;
va = zeros(3,na);
va(1,:) = -1;
fatt = zeros(3,na);
Fx = zeros(3,na);
Fx1 = zeros(3,na);
Fa = zeros(3,na);
rep = zeros(3,na);
[nx,nz]=size(ppt1);
pb2(1,na)= pb(1);pb2(2,na)= pb(2);pb2(3,na)= pb(3);
Fpa = zeros(3,na);
Fpa1 =zeros(3,na);
ds1 = zeros(3,na);

while max(abs(sum(pb2-pa)))>0.3

   for i=1:na
      a=[ppt1(1,:)-pa((i-1)*3+1);ppt1(2,:)-pa((i-1)*3+2);ppt1(3,:)-pa(i*3)];
      b = ppt2-ppt1;
      ss=sum(b.^2);
      t= -((dot(a,b))./ss);
      t=(t>1)+ (not(t>1).*t);
      t=not(t<0).*t;
      d = sqrt(sum(cross(b,a)).^2)./sqrt(ss);
      pt = ppt1+(b).*[t;t;t];
      ds = sqrt(sum([pt(1,:)-pa((i-1)*3+1);pt(2,:)-pa((i-1)*3+2);pt(3,:)-pa(i*3)].^2));
      ds = (ds<0.65).*ds;
      rep1 = (50*(-[pt(1,:)-pa((i-1)*3+1);pt(2,:)-pa((i-1)*3+2);pt(3,:)-pa(i*3)])./[ds;ds;ds]);
      rep1 = rep1(:,~isnan(rep1(1,:).*double(abs(rep1(1,:))<1000))~=0);
         if  sum(size(sum(rep1,2))>2);
            rep(:,i) = sum(rep1,2);
         else rep(:,i) = 0;
         end
      ds1 = sqrt(sum([pa(1,:)-pa((i-1)*3+1);pa(2,:)-pa((i-1)*3+2);pa(3,:)-pa(i*3)].^2));
      ds1 = (ds1<.7).*ds1;
      Fpa = (50*(-[pa(1,:)-pa((i-1)*3+1);pa(2,:)-pa((i-1)*3+2);pa(3,:)-pa(i*3)])./[ds1;ds1;ds1]);
      Fpa = Fpa(:,~isnan(Fpa(1,:).*double(abs(Fpa(1,:))<1000))~=0);
```

```
        if  sum(size(sum(Fpa,2))>2);
            Fpa1(:,i) = sum(Fpa,2);
        else Fpa1(:,i) = 0;
        end
    fatt(:,i) = 15*(pb-pa(:,i));
    nfatt=norm(fatt(:,i));
    if nfatt>10
        fatt(:,i)=10*fatt(:,i)/nfatt;
    end
    if norm(cross(fatt(:,i)/nfatt,rep(:,i)/norm(rep(:,i))))<.3 && norm(rep(:,i))~=0
        Fx(:,i) = cross([0;0;200],fatt(:,i)/nfatt);
    else Fx(:,i) = 0;
    end
    if norm(cross(fatt(:,i)/nfatt,Fpa1(:,i)/norm(Fpa1(:,i))))<.3 && norm(Fpa1(:,i))~=0
        Fx1(:,i) = cross([0;0;250],fatt(:,i)/nfatt);
    else Fx1(:,i) = 0;
    end
    Fa(:,i) = fatt(:,i) + rep(:,i) + Fx(:,i) + Fpa1(:,i) + Fx1(:,i)- 10*va(:,i);
    va(:,i) = va(:,i)+((Fa(:,i)./ma).*dt);
    if norm(va(:,i))>50;
        va(:,i) = 50*(va(:,i)./norm(va(:,i)));
    end
    pa(:,i) = pa(:,i) + (va(:,i).*dt);

end
clf


axis([-5 45 -15 25])
hold on
grid on
plot3(p1,p2,p3);
plot3(pa(1,:),pa(2,:),pa(3,:),'o');
plot3(pb(1,:),pb(2,:),pb(3,:),'s');
plot3([pa(1) pa(1)+fatt(1)],[pa(2) pa(2)+fatt(2)],[pa(3) pa(3)+fatt(3)],'b');
plot3([pa(1) pa(1)+Fx(1)],[pa(2) pa(2)+Fx(2)],[pa(3) pa(3)+Fx(3)],'g');
plot3([pa(1) pa(1)+Fpa1(1)],[pa(2) pa(2)+Fpa1(2)],[pa(3) pa(3)+Fpa1(3)],'k');
plot3([pa(1) pa(1)+Fx1(1)],[pa(2) pa(2)+Fx1(2)],[pa(3) pa(3)+Fx1(3)],'m');
```

```
    tom = sum(rep,2);
    if sum(size(tom))>2;
        plot3([pa(1) pa(1)+tom(1)],[pa(2) pa(2)+tom(2)],[pa(3) pa(3)+tom(3)],'o');
    else
        plot3([pa(1) pa(1)+rep(1)],[pa(2) pa(2)+rep(2)],[pa(3) pa(3)+rep(3)],'r');
    end
    drawnow
end
toc
```

ภาคผนวก ข

Source Code

Visual C++

```
void human1 (void)
{
        int i ;
        //man//
                        glPushMatrix();
                        glTranslatef(22,8,0);
                        glBegin(GL_TRIANGLES);
                        setColor(1.000f,0.000f,1.000f);
                        for (i=0;i<4068;i++){
                                glVertex3f(hhhp1[i][0]-0.5,hhhp1[i][1]-1.225,hhhp1[i][2]);
                        }
                        glEnd();
                        glPopMatrix();
}
void human2 (void)
{
        int i;
                        glPushMatrix();
                        glTranslatef(15,3,0);
                        glBegin(GL_TRIANGLES);
                        setColor(1.000f,0.000f,1.000f);
                        for (i=0;i<4068;i++){
                                glVertex3f(hhhp1[i][0]-0.5,hhhp1[i][1]-2.5,hhhp1[i][2]);
                        }
                        glEnd();
                        glPopMatrix();
}

void human3 (void)
{
        int i;
                        glPushMatrix();
                        glTranslatef(11,3,0);
                        glBegin(GL_TRIANGLES);
                        setColor(1.000f,0.000f,1.000f);
```

```
                for (i=0;i<4068;i++){
                        glVertex3f(hhhp1[i][0],hhhp1[i][1],hhhp1[i][2]);
                }
                glEnd();
                glPopMatrix();
}

void human4 (void)
{
        int i;
                glPushMatrix();
                glTranslatef(19,8,0);
                glBegin(GL_TRIANGLES);
                setColor(1.000f,0.000f,1.000f);
                for (i=0;i<4068;i++){
                        glVertex3f(hhhp1[i][0]-0.2,hhhp1[i][1]-1,hhhp1[i][2]);
                }
                glEnd();
                glPopMatrix();
}

void human5 (void)
{
        int i;
                glPushMatrix();
                glTranslatef(7,5,0);
                glBegin(GL_TRIANGLES);
                setColor(1.000f,0.000f,1.000f);
                for (i=0;i<4068;i++){
                        glVertex3f(hhhp1[i][0],hhhp1[i][1],hhhp1[i][2]);
                }
                glEnd();
                glPopMatrix();
}

void human6 (void)
{
        int i;
```

```
                glPushMatrix();
                glTranslatef(6,6,0);
                glBegin(GL_TRIANGLES);
                setColor(1.000f,0.000f,1.000f);
                for (i=0;i<4068;i++){
                        glVertex3f(hhhp1[i][0]-0.5,hhhp1[i][1]-1.225,hhhp1[i][2]);
                }
                glEnd();
                glPopMatrix();
}

void human7 (void)
{
        int i;
                glPushMatrix();
                glTranslatef(20,7,0);
                glBegin(GL_TRIANGLES);
                setColor(1.000f,0.000f,1.000f);
                for (i=0;i<4068;i++){
                        glVertex3f(hhhp1[i][0]-0.5,hhhp1[i][1]-1.225,hhhp1[i][2]);
                }
                glEnd();
                glPopMatrix();
}

void human8 (void)
{
        int i;
                glPushMatrix();
                glTranslatef(5,9,0);
                glBegin(GL_TRIANGLES);
                setColor(1.000f,0.000f,1.000f);
                for (i=0;i<4068;i++){
                        glVertex3f(hhhp1[i][0]-0.5,hhhp1[i][1]-1.225,hhhp1[i][2]);
                }
                glEnd();
                glPopMatrix();
}
```

```
void human9 (void)
{
        int i;
                        glPushMatrix();
                        glTranslatef(12,5,0);
                        glBegin(GL_TRIANGLES);
                        setColor(1.000f,0.000f,1.000f);
                        for (i=0;i<4068;i++){
                                glVertex3f(hhhp1[i][0]-0.5,hhhp1[i][1]-1.225,hhhp1[i][2]);
                        }
                        glEnd();
                        glPopMatrix();
}

void human10 (void)
{
        int i;
                        glPushMatrix();
                        glTranslatef(3,4,0);
                        glBegin(GL_TRIANGLES);
                        setColor(1.000f,0.000f,1.000f);
                        for (i=0;i<4068;i++){
                                glVertex3f(hhhp1[i][0],hhhp1[i][1],hhhp1[i][2]);
                        }
                        glEnd();
                        glPopMatrix();
}

void human11 (void)
{
        int i;
                        glPushMatrix();
                        glTranslatef(13,1,0);
                        glBegin(GL_TRIANGLES);
                        setColor(1.000f,0.000f,1.000f);
                        for (i=0;i<4068;i++){
                                glVertex3f(hhhp1[i][0],hhhp1[i][1],hhhp1[i][2]);
```

```
                        }
                        glEnd();
                        glPopMatrix();
}

void human12 (void)
{
        int i;
                        glPushMatrix();
                        glTranslatef(6,7,0);
                        glBegin(GL_TRIANGLES);
                        setColor(1.000f,0.000f,1.000f);
                        for (i=0;i<4068;i++){
                                glVertex3f(hhhp1[i][0]-0.5,hhhp1[i][1]-1.225,hhhp1[i][2]);
                        }
                        glEnd();
                        glPopMatrix();
}

void human13 (void)
{
        int i;
                        glPushMatrix();
                        glTranslatef(14,5,0);
                        glBegin(GL_TRIANGLES);
                        setColor(1.000f,0.000f,1.000f);
                        for (i=0;i<4068;i++){
                                glVertex3f(hhhp1[i][0]-0.5,hhhp1[i][1]-1.225,hhhp1[i][2]);
                        }
                        glEnd();
                        glPopMatrix();
}

void human14 (void)
{
        int i;
                        glPushMatrix();
                        glTranslatef(10,7,0);
```

```
                glBegin(GL_TRIANGLES);
                setColor(1.000f,0.000f,1.000f);
                for (i=0;i<4068;i++){
                        glVertex3f(hhhp1[i][0]-0.5,hhhp1[i][1]-1.225,hhhp1[i][2]);
                }
                glEnd();
                glPopMatrix();
}

void human15 (void)
{
        int i;
                glPushMatrix();
                glTranslatef(15,3,0);
                glBegin(GL_TRIANGLES);
                setColor(1.000f,0.000f,1.000f);
                for (i=0;i<4068;i++){
                        glVertex3f(hhhp1[i][0],hhhp1[i][1],hhhp1[i][2]);
                }
                glEnd();
                glPopMatrix();
}

void human16 (void)
{
        int i;
                glPushMatrix();
                glTranslatef(5,8,0);
                glBegin(GL_TRIANGLES);
                setColor(1.000f,0.000f,1.000f);
                for (i=0;i<4068;i++){
                        glVertex3f(hhhp1[i][0],hhhp1[i][1],hhhp1[i][2]);
                }
                glEnd();
                glPopMatrix();
}

void human17 (void)
```

```
{
        int i;
                        glPushMatrix();
                        glTranslatef(11,7,0);
                        glBegin(GL_TRIANGLES);
                        setColor(1.000f,0.000f,1.000f);
                        for (i=0;i<4068;i++){
                                glVertex3f(hhhp1[i][0],hhhp1[i][1],hhhp1[i][2]);
                        }
                        glEnd();
                        glPopMatrix();
}

void human18 (void)
{
        int i;
                        glPushMatrix();
                        glTranslatef(16,12,0);
                        glBegin(GL_TRIANGLES);
                        setColor(1.000f,0.000f,1.000f);
                        for (i=0;i<4068;i++){
                                glVertex3f(hhhp1[i][0]-0.5,hhhp1[i][1]-1.225,hhhp1[i][2]);
                        }
                        glEnd();
                        glPopMatrix();
}

void human19 (void)
{
        int i;
                        glPushMatrix();
                        glTranslatef(16,9,0);
                        glBegin(GL_TRIANGLES);
                        setColor(1.000f,0.000f,1.000f);
                        for (i=0;i<4068;i++){
                                glVertex3f(hhhp1[i][0],hhhp1[i][1],hhhp1[i][2]);
                        }
                        glEnd();
```

```
                glPopMatrix();
}

void human20 (void)
{
        int i;
                glPushMatrix();
                glTranslatef(18,6,0);
                glBegin(GL_TRIANGLES);
                setColor(1.000f,0.000f,1.000f);
                for (i=0;i<4068;i++){
                        glVertex3f(hhhp1[i][0]-0.5,hhhp1[i][1]-1.225,hhhp1[i][2]);
                }
                glEnd();
                glPopMatrix();
}

void human21 (void)
{
        int i;
                glPushMatrix();
                glTranslatef(25,3,0);
                glBegin(GL_TRIANGLES);
                setColor(1.000f,0.000f,1.000f);
                for (i=0;i<4068;i++){
                        glVertex3f(hhhp1[i][0]-0.5,hhhp1[i][1]-1.225,hhhp1[i][2]);
                }
                glEnd();
                glPopMatrix();
}

void human22 (void)
{
        int i;
                glPushMatrix();
                glTranslatef(32,5,0);
                glBegin(GL_TRIANGLES);
                setColor(1.000f,0.000f,1.000f);
```

```
                for (i=0;i<4068;i++){
                        glVertex3f(hhhp1[i][0]-0.5,hhhp1[i][1]-1.225,hhhp1[i][2]);
                }
                glEnd();
                glPopMatrix();
}

void human23 (void)
{
        int i;
                glPushMatrix();
                glTranslatef(34,1,0);
                glBegin(GL_TRIANGLES);
                setColor(1.000f,0.000f,1.000f);
                for (i=0;i<4068;i++){
                        glVertex3f(hhhp1[i][0]-0.5,hhhp1[i][1]-1.225,hhhp1[i][2]);
                }
                glEnd();
                glPopMatrix();
}

void human24 (void)
{
        int i;
                glPushMatrix();
                glTranslatef(29,7,0);
                glBegin(GL_TRIANGLES);
                setColor(1.000f,0.000f,1.000f);
                for (i=0;i<4068;i++){
                        glVertex3f(hhhp1[i][0]-0.5,hhhp1[i][1]-1.225,hhhp1[i][2]);
                }
                glEnd();
                glPopMatrix();
}

void human25 (void)
{
        int i;
```

```
                glPushMatrix();
                glTranslatef(13,14,0);
                glBegin(GL_TRIANGLES);
                setColor(1.000f,0.000f,1.000f);
                for (i=0;i<4068;i++){
                        glVertex3f(hhhp1[i][0]-0.5,hhhp1[i][1]-1.225,hhhp1[i][2]);
                }
                glEnd();
                glPopMatrix();
}

void human26 (void)
{
        int i;
                glPushMatrix();
                glTranslatef(23,2,0);
                glBegin(GL_TRIANGLES);
                setColor(1.000f,0.000f,1.000f);
                for (i=0;i<4068;i++){
                        glVertex3f(hhhp1[i][0]-0.5,hhhp1[i][1]-1.225,hhhp1[i][2]);
                }
                glEnd();
                glPopMatrix();
}

void human27 (void)
{
        int i;
                glPushMatrix();
                glTranslatef(30,7,0);
                glBegin(GL_TRIANGLES);
                setColor(1.000f,0.000f,1.000f);
                for (i=0;i<4068;i++){
                        glVertex3f(hhhp1[i][0]-0.5,hhhp1[i][1]-1.225,hhhp1[i][2]);
                }
                glEnd();
                glPopMatrix();
}
```

```
void human28 (void)
{
        int i;
                        glPushMatrix();
                        glTranslatef(18,14,0);
                        glBegin(GL_TRIANGLES);
                        setColor(1.000f,0.000f,1.000f);
                        for (i=0;i<4068;i++){
                                glVertex3f(hhhp1[i][0]-0.5,hhhp1[i][1]-1.225,hhhp1[i][2]);
                        }
                        glEnd();
                        glPopMatrix();
}

void human29 (void)
{
        int i;
                        glPushMatrix();
                        glTranslatef(20.1,9.25,0);
                        glBegin(GL_TRIANGLES);
                        setColor(1.000f,0.000f,1.000f);
                        for (i=0;i<4068;i++){
                                glVertex3f(hhhp1[i][0]-0.5,hhhp1[i][1]-1.225,hhhp1[i][2]);
                        }
                        glEnd();
                        glPopMatrix();
}

void human30 (void)
{
        int i;
                        glPushMatrix();
                        glTranslatef(28.75,6.5,0);
                        glBegin(GL_TRIANGLES);
                        setColor(1.000f,0.000f,1.000f);
                        for (i=0;i<4068;i++){
                                glVertex3f(hhhp1[i][0]-0.5,hhhp1[i][1]-1.225,hhhp1[i][2]);
```

```
                }
                glEnd();
                glPopMatrix();
}

void human31 (void)
{
        int i;
                glPushMatrix();
                glTranslatef(32.5,9.13,0);
                glBegin(GL_TRIANGLES);
                setColor(1.000f,0.000f,1.000f);
                for (i=0;i<4068;i++){
                        glVertex3f(hhhp1[i][0]-0.5,hhhp1[i][1]-1.225,hhhp1[i][2]);
                }
                glEnd();
                glPopMatrix();
}

void human32 (void)
{
        int i;
                glPushMatrix();
                glTranslatef(33.15,8.2,0);
                glBegin(GL_TRIANGLES);
                setColor(1.000f,0.000f,1.000f);
                for (i=0;i<4068;i++){
                        glVertex3f(hhhp1[i][0]-0.5,hhhp1[i][1]-1.225,hhhp1[i][2]);
                }
                glEnd();
                glPopMatrix();
}

void human33 (void)
{
        int i;
                glPushMatrix();
                glTranslatef(32.13,1.6,0);
```

```
                glBegin(GL_TRIANGLES);
                setColor(1.000f,0.000f,1.000f);
                for (i=0;i<4068;i++){
                        glVertex3f(hhhp1[i][0]-0.5,hhhp1[i][1]-1.225,hhhp1[i][2]);
                }
                glEnd();
                glPopMatrix();
}

void human34 (void)
{
        int i;
                glPushMatrix();
                glTranslatef(18.75,1.7,0);
                glBegin(GL_TRIANGLES);
                setColor(1.000f,0.000f,1.000f);
                for (i=0;i<4068;i++){
                        glVertex3f(hhhp1[i][0]-0.5,hhhp1[i][1]-1.225,hhhp1[i][2]);
                }
                glEnd();
                glPopMatrix();
}

void human35 (void)
{
        int i;
                glPushMatrix();
                glTranslatef(32.7,0.92,0);
                glBegin(GL_TRIANGLES);
                setColor(1.000f,0.000f,1.000f);
                for (i=0;i<4068;i++){
                        glVertex3f(hhhp1[i][0]-0.5,hhhp1[i][1]-1.225,hhhp1[i][2]);
                }
                glEnd();
                glPopMatrix();
}

void human36 (void)
```

```
{
        int i;
                        glPushMatrix();
                        glTranslatef(20.65,5.21,0);
                        glBegin(GL_TRIANGLES);
                        setColor(1.000f,0.000f,1.000f);
                        for (i=0;i<4068;i++){
                                glVertex3f(hhhp1[i][0]-0.5,hhhp1[i][1]-1.225,hhhp1[i][2]);
                        }
                        glEnd();
                        glPopMatrix();
}

void human37 (void)
{
        int i;
                        glPushMatrix();
                        glTranslatef(8,9,0);
                        glBegin(GL_TRIANGLES);
                        setColor(1.000f,0.000f,1.000f);
                        for (i=0;i<4068;i++){
                                glVertex3f(hhhp1[i][0]-0.5,hhhp1[i][1]-1.225,hhhp1[i][2]);
                        }
                        glEnd();
                        glPopMatrix();
}

void human38 (void)
{
        int i;
                        glPushMatrix();
                        glTranslatef(10.61,-2.71,0);
                        glBegin(GL_TRIANGLES);
                        setColor(1.000f,0.000f,1.000f);
                        for (i=0;i<4068;i++){
                                glVertex3f(hhhp1[i][0]-1,hhhp1[i][1],hhhp1[i][2]);
                        }
                        glEnd();
```

```
                glPopMatrix();
}

void human39 (void)
{
        int i;
                glPushMatrix();
                glTranslatef(10.56,-2.17,0);
                glBegin(GL_TRIANGLES);
                setColor(1.000f,0.000f,1.000f);
                for (i=0;i<4068;i++){
                        glVertex3f(hhhp1[i][0]-1,hhhp1[i][1],hhhp1[i][2]);
                }
                glEnd();
                glPopMatrix();
}

void human40 (void)
{
        int i;
                glPushMatrix();
                glTranslatef(8.11,-1.5,0);
                glBegin(GL_TRIANGLES);
                setColor(1.000f,0.000f,1.000f);
                for (i=0;i<4068;i++){
                        glVertex3f(hhhp1[i][0]-0.5,hhhp1[i][1]-1.225,hhhp1[i][2]);
                }
                glEnd();
                glPopMatrix();
}

void human41 (void)
{
        int i;
                glPushMatrix();
                glTranslatef(12.21,-1.65,0);
                glBegin(GL_TRIANGLES);
                setColor(1.000f,0.000f,1.000f);
```

```
                for (i=0;i<4068;i++){
                        glVertex3f(hhhp1[i][0]-0.5,hhhp1[i][1]-1.225,hhhp1[i][2]);
                }
                glEnd();
                glPopMatrix();
}

void human42 (void)
{
        int i;
                glPushMatrix();
                glTranslatef(13.17,-1.7,0);
                glBegin(GL_TRIANGLES);
                setColor(1.000f,0.000f,1.000f);
                for (i=0;i<4068;i++){
                        glVertex3f(hhhp1[i][0]-0.5,hhhp1[i][1]-1.225,hhhp1[i][2]);
                }
                glEnd();
                glPopMatrix();
}

void human43 (void)
{
        int i;
                glPushMatrix();
                glTranslatef(18.12,-3.1,0);
                glBegin(GL_TRIANGLES);
                setColor(1.000f,0.000f,1.000f);
                for (i=0;i<4068;i++){
                        glVertex3f(hhhp1[i][0]-0.5,hhhp1[i][1]-1.225,hhhp1[i][2]);
                }
                glEnd();
                glPopMatrix();
}

void human44 (void)
{
        int i;
```

```
                glPushMatrix();
                glTranslatef(17.6,-2.48,0);
                glBegin(GL_TRIANGLES);
                setColor(1.000f,0.000f,1.000f);
                for (i=0;i<4068;i++){
                        glVertex3f(hhhp1[i][0]-0.5,hhhp1[i][1]-1.225,hhhp1[i][2]);
                }
                glEnd();
                glPopMatrix();
}

void human45 (void)
{
        int i;
                glPushMatrix();
                glTranslatef(33.4,9.4,0);
                glBegin(GL_TRIANGLES);
                setColor(1.000f,0.000f,1.000f);
                for (i=0;i<4068;i++){
                        glVertex3f(hhhp1[i][0]-0.5,hhhp1[i][1]-1.225,hhhp1[i][2]);
                }
                glEnd();
                glPopMatrix();
}

void human46 (void)
{
        int i;
                glPushMatrix();
                glTranslatef(12.81,13.21,0);
                glBegin(GL_TRIANGLES);
                setColor(1.000f,0.000f,1.000f);
                for (i=0;i<4068;i++){
                        glVertex3f(hhhp1[i][0]-0.5,hhhp1[i][1]-1.225,hhhp1[i][2]);
                }
                glEnd();
                glPopMatrix();
}
```

```
void human47 (void)
{
        int i;
                        glPushMatrix();
                        glTranslatef(13.72,10.22,0);
                        glBegin(GL_TRIANGLES);
                        setColor(1.000f,0.000f,1.000f);
                        for (i=0;i<4068;i++){
                                glVertex3f(hhhp1[i][0]-0.5,hhhp1[i][1]-1.225,hhhp1[i][2]);
                        }
                        glEnd();
                        glPopMatrix();
}

void human48 (void)
{
        int i;
                        glPushMatrix();
                        glTranslatef(-3.23,9.61,0);
                        glBegin(GL_TRIANGLES);
                        setColor(1.000f,0.000f,1.000f);
                        for (i=0;i<4068;i++){
                                glVertex3f(hhhp1[i][0]-0.5,hhhp1[i][1]-1.225,hhhp1[i][2]);
                        }
                        glEnd();
                        glPopMatrix();
}

void human49 (void)
{
        int i;
                        glPushMatrix();
                        glTranslatef(2.22,8.76,0);
                        glBegin(GL_TRIANGLES);
                        setColor(1.000f,0.000f,1.000f);
                        for (i=0;i<4068;i++){
                                glVertex3f(hhhp1[i][0]-1,hhhp1[i][1],hhhp1[i][2]);
```

```
                        }
                        glEnd();
                        glPopMatrix();
}

void human50 (void)
{
        int i;
                        glPushMatrix();
                        glTranslatef(4.17,7.89,0);
                        glBegin(GL_TRIANGLES);
                        setColor(1.000f,0.000f,1.000f);
                        for (i=0;i<4068;i++){
                                glVertex3f(hhhp1[i][0],hhhp1[i][1],hhhp1[i][2]);
                        }
                        glEnd();
                        glPopMatrix();
}

void myDisplay()
{
        int j;
                for (j = 0; j < sizeof(x); j++){
                        glLoadIdentity();
                        initializeGL();
                        glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
                        glPushMatrix();
                        glRotated(-90,1,0,0);
                        Environment();
                        glTranslatef(x[j],y[j],0);
                        glBegin(GL_TRIANGLES);
                        human1();
                /*      human2();
                        human3();
                        human4();
                        human5();
                        human6();
                        human7();
```

human8();

human9();

human10 ();

human11 ();

human12 ();

human13 ();

human14 ();

human15 ();

human16 ();

human17 ();

human18 ();

human19 ();

human20 ();

human21 ();

human22 ();

human23 ();

human24 ();

human25 ();

human26 ();

human27 ();

human28 ();

human29 ();

human30 ();

human31 ();

human32 ();

human33 ();

human34 ();

human35 ();

human36 ();

human37 ();

human38 ();

human39 ();

human40 ();

human41 ();

human42 ();

human43 ();

human44 ();

human45 ();

```
                human46         ();
                human47         ();
                human48         ();
                human49         ();
                human50         ();

                glTranslatef(x[j],y[j],0);
                glBegin(GL_TRIANGLES);
                human1();*/
                glPopMatrix();
                //glFlush();
                glutSwapBuffers();
        }
}
```