ระบบการขี่รถจักรยานในสภาพแวดล้อมเสมือนจริง

VIRTUAL BICYCLE II

นายสุวิชา แก้วละเอียด
นายสุวิทย์ ไชยอำนาจ
นายองศกร ตรีสินชุ่ไชย
นายอชิระ ตระการสุดเลิศ

ปริญญานิพนธ์เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต
ภาควิชาวิฒวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2547

ระบบการขี่รถจักรยานในสภาพแวดล้อมเสมือนจริง

VIRTUAL BICYCLE II

โดย

นายสุวิชา แก้วละเอียด

นายสุวิทย์ ไชยอำนาจ

นายองศกร ตรีสินธุ์ไชย

นายอชิระ ตระการสุดเลิศ

อาจารย์ที่ปรึกษา

ดร.สมศักดิ์ วลัยรัชต์

ดร.อรัญญา วลัยรัชต์

อ.สมเกียรติ วังศิริพิทักษ์

ปริญญานิพนธ์เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2547

ปริญญานิพนธ์ปีการศึกษา 2547

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สภาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบการขี่รถจักรยานในสภาพแวดล้อมเสมือนจริง

VIRTUAL BICYCLE II

ผู้จัดทำ

1.นาย สุวิชา แก้วละเอียด รหัสประจำตัว 44010564

2.นาย สุวิทย์ ไชยอำนาจ รหัสประจำตัว 44010565

3.นาย องศกร ตรีสินธุ์ไชย รหัสประจำตัว 44010573

4.นาย อชิระ ตระการสุคเลิศ รหัสประจำตัว 44010574

_____ อาจารย์ที่ปรึกษา

(ดร.สมศักดิ์ วลัยรัชต์)

_____ อาจารย์ที่ปรึกษา

(ดร.อรัญญา วลัยรัชต์)

_____ อาจารย์ที่ปรึกษา

(อ.สมเกียรติ วังศิริพิทักษ์)

ระบบการขี่รถจักรยานในสภาพแวดล้อมเสมือนจริง

นาย สุวิชา แก้วละเอียด 44010564
นาย สุวิทย์ ไชยอำนาจ 44010565
นาย องศกร ตรีสินธุ์ไชย 44010573
นาย อชิระ ตระการสุดเลิศ 44010574
ดร.สมศักดิ์ วลัยรัชต์ อาจารย์ที่ปรึกษา
ดร.อรัญญา วลัยรัชต์ อาจารย์ที่ปรึกษาร่วม
อ.สมเกียรติ วังศิริพิทักษ์ อาจารย์ที่ปรึกษาร่วม
ปีการศึกษา 2547

บทคัดย่อ

โครงงานการขี่รถจักรยานในสภาพแวดล้อมเสมือนจริง นั้นมีจุดประสงค์เพื่อให้ความเพลิดเพลินและเป็นเสมือนเครื่องออกกำลังกายชิ้นหนึ่งซึ่งเป็นอุปกร์อกกำลังกายที่น่าใช้กว่าการขี่จักรยานในสถานที่ออกกำลังกายเพราะว่า โครงงานขี่รถจักรยานในสภาพแวดล้อมเสมือนจริง นั้นจะไม่ใช่แค่การขี่จักรยานธรรมดาได้มีการออกแบบอุปกรณ์ให้คล้ายกับสภาวะแวดล้อมจริง มีหน้าจอเสมือนผู้ขี่กำลังมองไปในสภาวะแวดล้อมจริง โดยถ้ามีการผ่านพื้นผิวขรุขระโปรแกรมก็จะมีการสั่งงานให้จักรยานสั่น หรือถ้ามีการขี่ขึ้นเนินโปรแกรมก็จะสั่งให้ยกด้านหน้าให้สูงกว่าด้านหลังและมีการหน่วงที่เพื่อผู้ขี่ให้รู้สึกเหมือนขี่ขึ้นเนินจริงๆ โดยตัวโปรแกรมนั้นพัฒนาด้วยภาษา C++, MFC, WIN32 API ทำงานกับไลบรารี่ที่ช่วยจัดการภาพ 3 มิติอย่าง DirectX และอุปกรณ์ที่ใช้ในการสร้างสภาวะจำลองในการขับขี่นั้นถูกออกแบบและควบคุมด้วย ไมโครคอนโทรเลอร์ MCS-51 ติดต่อกับโปรแกรมบน คอมพิวเตอร์ ผ่านพอร์ตอนุกรม โดยจักรยานมีการส่งอินพุต เกี่ยวกับองศาการเลี้ยวและความเร็ว เข้ามาประมวลผล และที่ คอมพิวเตอร์ จะมีโปรแกรมจำลองสภาพแวดล้อมอยู่ ก็จะนำมาประมวลผลร่วมกับอินพุตเพื่อประมวลผลเอาพุตส่งไปที่จักรยาน แต่ว่าโครงงานนี้ก็ยังมีข้อเสียอยู่อย่างเช่น ยิ่งต้องการความเหมือนจริงมาก อุปกรณ์ก็ยิ่งมีราคาแพง การสร้างก็จะยากขึ้น ซึ่งถ้ามีงบประมาณเพียงพอโครงงานนี้น่าจะเป็นตัวเลือกของเครื่องออกกำลังกายที่ดีตัวหนึ่งเลยทีเดียว

VIRTUAL BICYCLE II

Suwicha Kaew la – iad
Suwit Chaiamnart
Ongsakorn Treesinchai
Achira Trakansutleart
Dr. Somsak Walairach Advisor
Dr. Aranya Walairach Co-Advisor
Somkiat Waangsiripitak Co-Advisor

ABSTRACT

This Virtual Bicycle II project is built in order to make a virtual bicycle for entertainment and exercise and to replace a traditional exercising bicycle. The difference between Virtual Bicycle and traditional exercising bicycle is its virtual so this makes the Virtual Bicycle more interesting than traditional exercising bicycle or even more interesting than the real bicycle.

The Virtual Bicycle is not only a simple riding machine but it is installed with some devices for more realistic riding. It has a monitor that shows a three-dimensional (3D) scene, it also has three sensors that detect speed, turn angle and heart rate respectively. Additional, it has some feedbacks from the system to user such as bicycle lifting feedback and wheel braking feedback when riding up or down the bridge and also a bicycle vibration feedback when riding through a rough way.

The program is developed by using C++ language with MFC and Win32 API and also using DirectX to manage a three-dimensional (3D) graphic. The devices on the Virtual Bicycle are controlled by MCS-51 microcontroller which is connected to the "Hardware Server" computer by a serial connection. The "Hardware Server" is used to receive speed, turn angle and heart rate from the Virtual Bicycle, then compute and change these data into the format that the "Graphic Server" can use. It is also used to receive feedbacks from the "Graphic Server", and then create signals to control feedback devices on the Virtual Bicycle. The "Graphic Server" is used to create and render a 3D scene. However, this project has still some problems such as the device problem because if we want more realistic, we must use more expensive devices. Therefore if the budget is efficient, this project will be one of the good choices for exercise.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จได้ด้วยความกรุณาของ อาจารย์ ดร.สมศักดิ์ วลัยรัชต์ อาจารย์ที่ปรึกษาโครงงาน ซึ่งได้ให้คำปรึกษา ข้อชี้แนะ และความช่วยเหลือในหลายสิ่งหลายอย่าง จนกระทั่งลุล่วงไปได้ด้วยดี คณะวิจัยขอกราบขอบพระคุณเป็นอย่างสูง ณ ที่นี้

ขอกราบขอบพระคุณ คณาจารย์ภาควิชาคอมพิวเตอร์ทุกท่านที่ได้ให้ความรู้ตลอดการศึกษาที่ผ่านมา ขอขอบคุณพี่ๆ และเจ้าหน้าที่ ที่ให้ความกรุณาให้ยืมอุปกรณ์ต่างๆ ในการทำงานด้านฮาร์ดแวร์

ขอบคุณและขอบใจ พี่ๆเพื่อนๆน้องๆ ที่ช่วยให้ทั้ง แนวคิด คำแนะนำต่างที่ช่วยในการทำงานโครงงานนี้เพื่อให้งานออกตามที่ต้องการ ทั้งยังช่วยให้ กำลังใจในยามที่เหนื่อยล้าและ ท้อแท้รวมถึงผู้มีพระคุณที่มิได้เอ่ยนามไว้ ณ ที่นี้

สุดท้ายนี้ต้องขอขอบพระคุณบุคคลสำคัญที่สุดที่ทำให้ข้าพเจ้ามีวันนี้ ก็คือ บิดา มารดา อันเป็นที่เคารพรักยิ่ง ซึ่งเลี้ยงดูมาเป็นอย่างดี ทั้งยังให้การศึกษาอย่างเต็มที่ และยังให้กำลังใจเอาใจใส่เสมอมา ในทุกๆด้านอันหาที่เปรียบไม่ได้ ข้าพเจ้าขอระลึกพระคุณอันสุดประมาณและขอกราบขอบพระคุณมา ณ ที่นี้

สารบัญ

สารบัญตาราง

สารบัญรูปภาพ

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

ในปัจจุบันนี้ คอมพิวเตอร์ได้เข้ามามีบทบาทกับชีวิตประจำวันของมนุษย์แทบทุกคน ไม่ว่าจะนำมาใช้ทำงาน ติดต่อสื่อสาร หรือเพื่อความบันเทิง และในอนาคตคอมพิวเตอร์ก็จะยิ่งมีบทบาทต่อชีวิตมนุษย์มากขึ้นไปอีก และด้วยเหตุที่คอมพิวเตอร์มีความสำคัญมากขึ้นนี่เอง จึงทำให้เกิดโปรแกรมต่าง ๆ ขึ้นมากมาย และพัฒนาขึ้นเรื่อยมาอย่างรวดเร็ว

การที่มีโปรแกรมมากมายนี่เอง ทำให้โปรแกรมเมอร์จะต้องออกแบบและเขียนโปรแกรมให้มีคุณภาพมากที่สุด และดึงดูดผู้ใช้ให้ได้มากที่สุด ซึ่งสิ่งหนึ่งที่สามารถดึงดูดผู้ใช้ได้มากที่สุด ก็คือ ภาพ ดังนั้น โปรแกรมใดก็ตามถ้ายิ่งมีภาพสวยงามเท่าใด ก็ยิ่งน่าสนใจมากขึ้นเท่านั้น

การแสดงความเป็นจริงเสมือนหรือเวอร์ชวล เรียลลิตี้ (Virtual Reality) ให้กับภาพก็เป็นเทคโนโลยีหนึ่งที่ช่วยให้โปรแกรมที่เขียนขึ้นดึงดูดผู้ใช้ได้ และไม่เพียงแต่ภาพเท่านั้นที่เราสามารถใช้หลักการนี้มาช่วยทำให้เสมือนจริงได้ ประสาทสัมผัสทั้ง 5 ของมนุษย์ทั้งรูป รส กลิ่น เสียง สัมผัสก็สามารถนำหลักการนี้มาใช้ได้เหมือนกัน ซึ่งการทำเช่นนี้ จะช่วยโปรแกรมตอบสนองความต้องการของผู้ใช้ได้ดียิ่งขึ้น

โครงงานนี้ เป็นโครงงานที่ใช้เทคโนโลยีเวอร์ชวล เรียลลิตี้มาช่วยในการสร้างระบบจำลองการขี่รถจักรยานในสภาพแวดล้อมเสมือน โดยจะนำมาใช้กับภาพ เสียง และสัมผัส ซึ่งจะช่วยให้ผู้ใช้สามารถขี่จักรยานออกกำลังกายพร้อมกับได้รับความบันเทิงและความรู้สึกเสมือนออกไปขี่จักรยานในสถานที่จริง โดยที่ไม่ต้องออกไปขี่จักรยานข้างนอกบ้าน

1.2 วัตถุประสงค์ของโครงงาน

1.2.1 เพื่อสร้างโปรแกรมประยุกต์จำลองการขี่จักรยานในสภาพแวดล้อมเสมือน ซึ่งจำลองสถานการณ์การกระทำในขณะขี่จักรยานด้วยภาพและเสียง 3 มิติ

1.2.2 เพื่อเป็นเครื่องขี่จักรยานออกกำลังกายที่มีความสมจริง ทำให้ผู้ที่ไม่มีเวลาว่างและไม่สะดวกที่จะขี่จักรยานจริง ๆ ใช้ออกกำลังกายได้

1.2.3 เพื่อใช้สำหรับการช่วยบำบัดผู้ป่วยพักฟื้น หรือผู้ที่ต้องการขี่จักรยานออกกำลังกายอย่างปลอดภัย

1.2.4 เพื่อศึกษาเทคโนโลยีในการสร้างโปรแกรมประยุกต์จำลองภาพ 3 มิติโดยใช้คอมโพเน็นท์ไดเร็กซ์เอ็กซ์กราฟฟิกของไดเร็กซ์เอ็กซ์ 9.0 (DirectX 9.0)

1.2.5 เพื่อศึกษาการพัฒนาโปรแกรมประยุกต์ด้วยการเขียนโปรแกรมในเชิงวัตถุด้วยภาษา C++ โดยใช้ไมโครซอฟท์ วิชวล สตูดิโอ ทูลคิท (Microsoft Visual C++ toolkit)

1.2.6 เพื่อพัฒนาอุปกรณ์ที่ใช้ติดต่อสื่อสารระหว่างตัวโปรแกรมและอุปกรณ์ที่ทำหน้าที่เป็นเซ็นเซอร์ (sensor) 1.และฟีดแบ็ค (feedback)

1.3 ขอบเขตของโครงงาน

โครงงานนี้เป็นการสร้างโปรแกรมประยุกต์จำลองการขี่จักรยานในสภาพแวดล้อมเสมือน โดยจำลองโลกเสมือนขึ้นมาเป็นแผนที่เล็ก ๆ แผนที่หนึ่ง และนำเสนอออกมาในรูปแบบกราฟฟิก 3 มิติ พร้อมเสียงซึ่งจะช่วยเพิ่มความเสมือนจริงมากยิ่งขึ้น

ผู้ใช้จะสามารถใช้งานโปรแกรมได้โดยการขี่จักรยานซึ่งตัวจักรยานจะเป็นฮาร์ดแวร์ที่ถูกสร้างขึ้นเพื่อให้สามารถทำงานต่าง ๆ ได้ ดังนี้

- สามารถสร้างแรงต้านที่เกิดจากการโน้มเอียงเวลาเลี้ยว
- สามารถยกตัวถังจักรยานและผู้ใช้ให้ยกขึ้นและกดลง เวลาขึ้นเนินและลงเนิน
- สามารถสร้างการสั่นสะเทือนเป็นจังหวะเบา ๆ เวลาผู้ใช้ขี่ผ่านทางขรุขระหรือลูกระนาด
- สามารถหน่วงล้อให้ล้อหมุนช้าลง ซึ่งจะทำให้ผู้ใช้ออกแรงปั่นมากขึ้นเมื่อเวลาขึ้นทางชัน
- สามารถรับรู้ถึงความเร็วของการขี่จักรยานเพื่อนำข้อมูลดังกล่าวไปประมวลผลเป็นการเคลื่อนที่ของจักรยานในโปรแกรมสามมิติ
- สามารถรับรู้ถึงองศาการเลี้ยวของผู้ใช้เพื่อใช้สร้างภาพการเลี้ยวของจักรยานในโปรแกรม 3 มิติ

สัญญาณข้อมูลต่าง ๆ ของฮาร์ดแวร์ที่ตัวจักรยานจะถูกนำมารวมกันและประมวลผลในขั้นต้นโดยใช้ MCS-51 หลังจากนั้น MCS-51 จะทำหน้าที่เป็นตัวส่งข้อมูลที่ผ่านการประมวลผลในขั้นต้นไปให้กับคอมพิวเตอร์เครื่องแรกที่เรียกว่า ฮาร์ดแวร์เซิร์ฟเวอร์ (Hardware Server) ผ่านทางพอร์ตอนุกรม ฮาร์ดแวร์เซิร์ฟเวอร์จะทำหน้าที่เป็นตัวเชื่อมต่อระหว่างตัวจักรยานกับคอมพิวเตอร์เครื่องที่สองที่เรียกว่า กราฟฟิกเซิร์ฟเวอร์ (Graphic Server) รวมทั้งเป็นตัวประมวลผลข้อมูลต่าง ๆ ทั้งที่รับมาจากตัวจักรยานและกราฟฟิกเซิร์ฟเวอร์ให้อยู่ในรูปแบบที่ทั้ง 2 อุปกรณ์ดังกล่าวสามารถนำไปใช้งานได้

ในส่วนของกราฟฟิกเซิร์ฟเวอร์จะทำหน้าที่สร้างภาพและเสียง 3 มิติ รวมทั้งรับข้อมูลต่าง ๆ มาจากฮาร์ดแวร์เซิร์ฟเวอร์ผ่านทางวินซอกส์ (Winsocks) แล้วนำมาแปลงเป็นข้อมูลที่สามารถนำไปใช้ในการเคลื่อนไหวภาพของโปรแกรมสามมิติ

1.4 วิธีการดำเนินงาน

1.4.1 ศึกษาภาพรวมของโครงงานเพื่อนำไปวางขอบเขตของโครงงาน และศึกษาเพื่อเลือกใช้เครื่องมือสำหรับการพัฒนาโครงงาน

1.4.2 วางขอบเขตของโครงงานจำลองการขี่จักรยานในสภาพแวดล้อมเสมือนจริง

1.4.3 วิเคราะห์และออกแบบระบบ และกำหนดและเลือกเครื่องมือที่จะนำไปพัฒนาโครงงาน

1.4.4 ศึกษาการพัฒนาโปรแกรมด้วยโปรแกรมไมโครซอฟท์ วิชวล ซีพลัสพลัส 6.0 ร่วมกับโปรแกรมจำลองกราฟฟิก 3 มิติ ไดเร็กซ์เอ็กซ์ 9.0 เพื่อนำมาใช้ในกราฟฟิกเซิร์ฟเวอร์ รวมทั้งศึกษาโปรแกรมในการสร้างกราฟฟิก 3 มิติต่าง ๆ เช่น ทรีดี สตูดิโอ แม็กซ์ 6.0 (3D Studio Max 6.0)

1.4.5 ศึกษาความเป็นไปได้และความเหมาะสมในการเลือกใช้อุปกรณ์ต่าง ๆ ที่จะนำมาใช้เป็นฮาร์ดแวร์ติดตัวจักรยาน รวมไปถึงศึกษาการเขียนโปรแกรม MCS-51 ด้วยภาษาซี

1.4.6 ศึกษาการพัฒนาโปรแกรมด้วยโปรแกรมไมโครซอฟท์ วิชวล ซีพลัสพลัส 6.0 (Microsoft Visual Studio 6.0) แบบเอ็มเอฟซี (MFC) เพื่อนำมาใช้ในฮาร์ดแวร์เซิร์ฟเวอร์

1.4.7 ศึกษาการติดต่อและรับส่งข้อมูลผ่านทางพอร์ตอนุกรมของโปรแกรมที่ถูกเขียนขึ้นด้วยภาษาซีพลัสพลัสแบบเอ็มเอฟซี และศึกษาการใช้งานวินซอกส์ซึ่งจะถูกนำมาใช้ในส่วนของการรับส่งข้อมูลระหว่างกราฟฟิกเซิร์ฟเวอร์และฮาร์ดแวร์เซิร์ฟเวอร์

1.4.8 เขียนโปรแกรมจำลองการขี่รถจักรยานในสภาพแวดล้อมเสมือน โดยจะแบ่งการเขียนโปรแกรมออกเป็น 3 ส่วนหลัก ๆ ดังนี้

1) กราฟฟิกเซิร์ฟเวอร์
2) ฮาร์ดแวร์เซิร์ฟเวอร์
3) MCS-51 ที่ฮาร์ดแวร์ซึ่งติดกับจักรยาน

รวมทั้งทดสอบโปรแกรมในแต่ละส่วน

1.4.9 สร้างวงจรพร้อมทั้งทดสอบอุปกรณ์ที่จะนำมาทำเป็นฮาร์ดแวร์ของจักรยานทีละส่วน และประกอบอุปกรณ์เหล่านั้นเข้าด้วยกัน

1.4.10 ทำการรวมโครงงานแต่ละส่วนเข้าด้วยกัน เพื่อให้สามารถทำงานร่วมกันได้

1.4.11 ทำการทดสอบโครงงานรวมทั้งหมด

1.4.12 แก้ไขส่วนที่ผิดพลาดให้มีการใช้งานที่ถูกต้อง

บทที่ 2

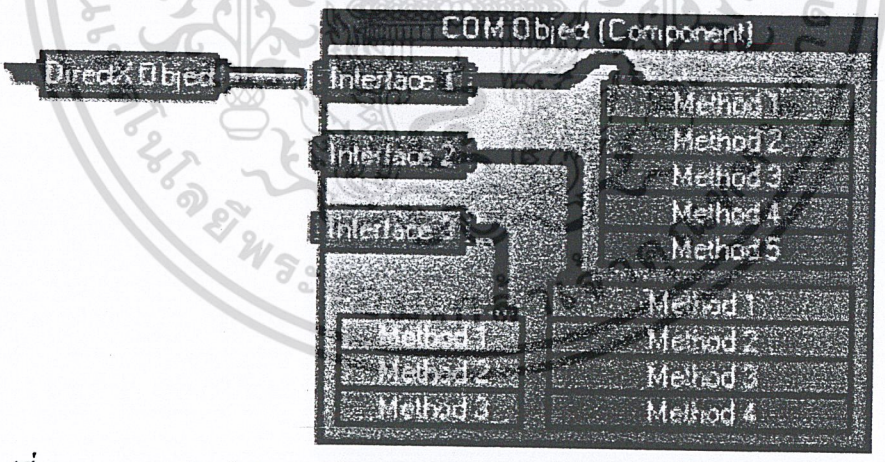
ทฤษฎีคอมโพเน็นท์ DirectX

2.1 คอมโพเน็นท์ออบเจ็กท์โมเดล (Component Object Model – COM)

คอมโพเน็นท์ออบเจ็กท์โมเดล (Component Object Model - COM) เป็นเทคโนโลยีที่ทำให้ไดเร็กซ์เอ็กซ์ (DirectX) เป็นโปรแกรมที่ไม่ขึ้นกับภาษาที่เขียนและมีความสามารถในการนำมาใช้ใหม่ ซึ่งจะช่วยให้การพัฒนาแอพพลิเคชันมีความง่ายขึ้น แนวคิดวิธีที่ทำให้ได้ผลดังกล่าว คือ ใช้หลักการออกแบบวัตถุ (object) แล้วรวมกันเป็นคอมโพเน็นท์ (component) โดยจะเรียกวัตถุย่อย ๆ เหล่านี้ใหม่ว่า อินเตอร์เฟส (interface)

2.2 ไดเร็กซ์เอ็กซ์ออบเจ็กท์ กับ คอมโพเน็นท์ออบเจ็กท์

คอมโพเน็นท์เปรียบเสมือนกล่องที่มีรูหรือช่องไว้ให้เสียบ ซึ่งอาจจะมีหลาย ๆ รู แต่ละรูเปรียบเสมือนอินเตอร์เฟส โดยแต่ละอินเตอร์เฟสก็จะมีฟังก์ชันการทำงานที่เรียกว่า เมธอด (Method) สำหรับไดเร็กซ์เอ็กซ์ออบเจ็กท์นั้นก็เปรียบเสมือนแจ็คที่สามารถเสียบที่รูหรืออินเตอร์เฟสได้ หากวัตถุของไดเร็กเอ็กซ์เสียบเข้าที่อินเตอร์เฟสใด ก็จะสามารถเรียกใช้งานฟังก์ชันการทำงานในอินเตอร์เฟสนั้นๆ ได้ หากต้องการใช้งานฟังก์ชันการทำงานอื่น ที่อยู่ในอินเตอร์เฟสอื่นๆ ก็ต้องสร้างไดเร็กซ์เอ็กซ์ออบเจ็กท์ขึ้นมาใหม่ เพื่อนำไปเสียบกับอินเตอร์เฟสที่จะใช้งานฟังก์ชันการทำงานที่ต้องการ และถ้าเลิกใช้อินเตอร์เฟสใด ก็ต้องถอดแจ็คหรือวัตถุนั้นออก ดังรูปที่ 2-1



รูปที่ 2-1 แสดงการติดต่อกันระหว่างคอมโพเน็นท์ออบเจ็กท์กับไดเร็กซ์เอ็กซ์ออบเจ็กท์

อินเตอร์เฟสแต่ละอันจะมีฟังก์ชันการทำงานต่าง ๆ ของตัวเอง หากต้องการจะใช้งานฟังก์ชันการทำงานใด ก็จะต้องสร้างไดเร็กซ์เอ็กซ์ออบเจ็กท์ที่ชี้ไปยังอินเตอร์เฟสที่มีฟังก์ชันการทำงานที่ต้องการ โดยการสร้างไดเร็กซ์เอ็กซ์ออบเจ็กท์นี้สามารถทำได้โดยการเรียกใช้ฟังก์ชันที่ไดเร็กซ์เอ็กซ์เตรียมไว้ให้ สำหรับการปรับปรุงความสามารถของวัตถุประเภท COM จะใช้การสร้างอินเตอร์เฟสตัวใหม่ขึ้นมา เพื่อใช้รองรับความสามารถใหม่มากกว่าที่จะเปลี่ยนแปลงฟังก์ชันที่มีอยู่ในอินเตอร์เฟสปัจจุบัน

2.3 อินเตอร์เฟสของ COM ออบเจ็กท์

ทุก COM ออบเจ็กท์ จะสนับสนุนอินเตอร์เฟสที่ชื่อ IUnknown อินเตอร์เฟสนี้เป็นอินเตอร์เฟสมาตรฐาน ซึ่งคอยควบคุมความสามารถในการเข้าใช้งานของไดเร็กซ์เอ็กซ์ออบเจ็กท์ที่เข้ามาใช้งานอินเตอร์เฟสต่าง ๆ โดยทุกอินเตอร์เฟสจะสืบทอดมาจากคลาส IUnknown ซึ่งเป็นคลาสที่ใช้สร้าง IUnknown อินเตอร์เฟส

สำหรับ IUnknown อินเตอร์เฟสนี้จะมีเมธอดอยู่ 3 ตัวด้วยกันคือ AddRef(), QueryInterface() และ Release() ดังนั้นอินเตอร์เฟสตัวอื่น ๆ ที่สืบทอดไปจึงมีเมธอดทั้ง 3 นี้ด้วยเสมอ ซึ่ง COM ออบเจ็กท์จะใช้ IUnknown เป็นตัวจัดการไดเร็กซ์เอ็กซ์ออบเจ็กท์ที่เข้ามาใช้อินเตอร์เฟส ที่อยู่ใน COM ออบเจ็กท์นั้น ๆ

- AddRef() จะใช้ในการเพิ่มค่าการนับให้กับตัวนับจำนวน (reference count) ของออบเจ็กท์ ขึ้นไป 1 เมื่ออินเตอร์เฟสหรือแอพพลิเคชันอื่น ๆ เข้าใช้งานออบเจ็กท์นั้น ๆ

- QueryInterface() ทำหน้าที่ตรวจดูว่าออบเจ็กท์นั้นมีอินเตอร์เฟสที่ต้องการหรือไม่ และยังสามารถใช้สร้างไดเร็กซ์เอ็กซ์ออบเจ็กท์ตัวใหม่ จากไดเร็กซ์เอ็กซ์ออบเจ็กท์เดิมที่มีอยู่ได้ด้วย โดยจะไปสอบถามCOM ออบเจ็กท์ ซึ่งเป็นเจ้าของอินเตอร์เฟสที่ไดเร็กซ์เอ็กซ์ออบเจ็กท์ตัวเดิมกำลังใช้อยู่ว่า มีอินเตอร์เฟสที่ไดเร็กซ์เอ็กซ์ออบเจ็กท์ตัวใหม่ต้องการหรือไม่

- Release() จะใช้ในการลดค่าการนับแก่ตัวนับจำนวนของออบเจ็กท์ลงไป 1 เมื่อลดลงไปถึง 0 ออบเจ็กท์จะถูกทำลาย

ขั้นตอนการทำงานเมื่อมีการสร้างไดเร็กซ์เอ็กซ์ออบเจ็กท์

ทุกครั้งที่มีการสร้างไดเร็กซ์เอ็กซ์ออบเจ็กท์ ซึ่งก็คือ การต้องการจะติดต่อกับอินเตอร์เฟสตัวใดในคอมโพเน็นท์ของไดเร็กซ์เอ็กซ์ โดย COM ออบเจ็กท์ จะเรียกฟังก์ชัน QueryInterface() เพื่อตรวจดูว่ามีอินเตอร์เฟสที่ต้องการใช้มีอยู่ในออบเจ็กท์หรือไม่ หากมีก็จะเตรียมพื้นที่ในหน่วยความจำแล้วส่งตำแหน่งแอดเดรสกลับมา พร้อมทั้งเรียกใช้ AddRef() เพื่อเพิ่มการนับแก่ตัวนับจำนวนอีก 1 ถ้ามีไดเร็กซ์เอ็กซ์ออบเจ็กท์ตัวอื่น ๆ มาใช้อินเตอร์เฟสอื่น ๆ หรือใช้อินเตอร์เฟสซ้ำกันกับไดเร็กซ์เอ็กซ์ออบเจ็กท์ตัวอื่น ตัวนับจำนวนของออบเจ็กท์ก็จะถูกบวกค่าเพิ่มขึ้นไปเรื่อย ๆ โดยหน่วยความจำที่ไดเร็กซ์เอ็กซ์ออบเจ็กท์ใช้นี้จะยังคงอยู่ตลอด หากไม่ทำการเรียกใช้ Release() แม้จะปิดโปรแกรมไปแล้วก็ตาม ซึ่งทำให้เกิดปัญหาเรื่องทรัพยากรตอบสนองไม่เพียงพอ ดังนั้น จะต้องเรียกใช้ Release() เมื่อไม่ต้องการใช้งานแล้วเสมอ เพื่อเป็นการคืนหน่วยความจำ และลดค่าตัวนับจำนวนของออบเจ็กท์ทีละ 1 เมื่อลดจนเหลือ 0 ก็แสดงว่า ไม่มีไดเร็กซ์เอ็กซ์ออบเจ็กท์ตัวใดใช้ COM ออบเจ็กท์นั้นแล้ว กลไกภายใน COM ก็จะลบ COM ออบเจ็กท์ ตัวนั้นออกไปจากหน่วยความจำโดยอัตโนมัติ

2.4 ไดเร็กซ์เอ็กซ์ 9.0

ไดเร็กซ์เอ็กซ์ เป็นชุดคำสั่ง API (Application programming interfaces) ที่ใช้ในการสร้างงานทางด้านมัลติมีเดียบนระบบปฏิบัติการวินโดวส์ ซึ่งไดเร็กซ์เอ็กซ์จะช่วยให้ผู้พัฒนาซอฟท์แวร์สามารถสร้างแอพพลิเคชันได้โดยไม่ต้องเข้าใจโค้ดระดับล่างที่ใช้ในการติดต่อฮาร์ดแวร์ เพียงแต่เรียกใช้อินเตอร์เฟสของไดเร็กซ์เอ็กซ์เท่านั้น เทคโนโลยีนี้ถูกนำมาใช้เมื่อปี ค.ศ. 1995 และได้กลายเป็นมาตรฐานในการพัฒนาแอพพลิเคชันทางด้านมัลติมีเดียบนระบบปฏิบัติการวินโดว์

นอกจากนี้ API ของไดเร็กซ์เอ็กซ์จะทำหน้าที่เสมือนหนึ่งเป็นสะพานในการเชื่อมการติดต่อระหว่างฮาร์ดแวร์และซอฟท์แวร์ ไดเร็กซ์เอ็กซ์ โดยไดเร็กซ์เอ็กซ์ API จะช่วยให้แอพพลิเคชันทางด้านมัลติมีเดียต่าง ๆ สามารถเรียกใช้การ์ดจอในการสร้างภาพ 3 มิติ สามารถเรียกใช้การ์ดเสียงเพื่อสร้างเสียง สามารถติดต่อกับอุปกรณ์อินพุทมาตรฐานต่าง ๆ ไดเร็กซ์เอ็กซ์ทั้งคีย์บอร์ด เมาส์ และจอยสติ๊ก รวมไปถึงการติดต่อกับกล้องเพื่อรับภาพเข้ามาใช้ในแอพพลิเคชัน

ไดเร็กซ์เอ็กซ์เวอร์ชัน 9.0 มีส่วนประกอบสำคัญ 8 ส่วน ดังนี้

1) ไดเร็กซ์เอ็กซ์กราฟฟิกส์ (DirectX Graphics) เป็นส่วนประกอบที่เกิดจากการรวมกันระหว่าง ไดเร็กซ์ดรอว์ (DirectDraw) และ ไดเร็กซ์สามดี (Direct3D) ซึ่งเป็นส่วนประกอบที่ใช้ในการสร้างกราฟฟิกทั้งหมด โดยจะต้องเรียกใช้ไลบรารี D3DX ก่อนการใช้ด้วย

2) ไดเร็กซ์อินพุท (DirectInput) เป็นส่วนที่ใช้ในการรับค่าจากอุปกรณ์อินพุทหลายชนิด ทั้งคีย์บอร์ด เมาส์ จอยสติ๊ก โดยจะสนับสนุนเทคโนโลยีฟอร์สฟีดแบ็ค (force-feedback) ด้วย

3) ไดเร็กซ์เพลย์ (DirectPlay) เป็นส่วนที่สนับสนุนเกมที่ต้องมีการติดต่อกันผ่านเครือข่าย

4) ไดเร็กซ์ซาวน์ (DirectSound) เป็นส่วนที่ใช้ในการพัฒนาแอพพลิเคชันที่มีการสร้างเสียงคุณภาพสูง

5) ไดเร็กซ์มิวสิก (DirectMusic) เป็นส่วนที่ใช้ในการสร้างเสียงเพลงให้กับแอพพลิเคชัน

6) ไดเร็กซ์โชว์ (DirectShow) เป็นส่วนที่ใช้ในการจับภาพหรือข้อมูลที่ได้จากอุปกรณ์มัลติมีเดียต่าง ๆ แล้วนำมาแสดงในแอพพลิเคชัน

7) ไดเร็กซ์เซ็ตอัพ (DirectSetup) เป็นส่วนที่ใช้สำหรับการจัดการทางด้านการติดตั้งองค์ประกอบของไดเร็กซ์เอ็กซ์ที่จะต้องใช้บนเครื่องคอมพิวเตอร์ของผู้ใช้อย่างอัตโนมัติ หากเครื่องคอมพิวเตอร์นั้นยังไม่ได้ทำการติดตั้ง

8) ไดเร็กซ์เอ็กซ์ มีเดีย ออบเจ็กท์ (DirectX Media Objects) เป็นส่วนที่ใช้ในการเขียนและใช้วัตถุที่มีรูปแบบข้อมูลเป็นสตรีม เช่น ตัวเข้ารหัสภาพและเสียง (Encoders), ตัวถอดรหัส (Decoders) และเอฟเฟกส์ (Effects)

บทที่ 3

Direct3D

3.1 ภาพรวมของ Direct3D

ไดเร็กซ์สามดีประกอบด้วยอินเตอร์เฟสและฟังก์ชันต่าง ๆ ที่ใช้ในการติดต่อกับอุปกรณ์กราฟฟิก โดยจะมีส่วนที่อยู่ระหว่างไดเร็กซ์สามดีและอุปกรณ์กราฟฟิกอีกชั้น ที่เรียกว่า HAL (Hardware Abstraction Layer) ทั้งนี้เพราะไดเร็กซ์สามดีไม่สามารถติดต่อโดยตรงกับอุปกรณ์กราฟฟิกที่มีความสามารถแตกต่างกันและมีอยู่มากมายตามท้องตลาดได้ ดังนั้นจึงต้องให้ผู้ผลิตอุปกรณ์กราฟฟิกแต่ละรุ่นสร้าง HAL ซึ่งเป็นชุดคำสั่งเฉพาะอุปกรณ์ที่สั่งให้อุปกรณ์นั้น ๆ ทำงาน ซึ่งถ้าเรียกใช้ฟังก์ชันของไดเร็กซ์เอ็กซ์ที่ไม่มีใน HAL ของอุปกรณ์ก็จะทำให้การทำงานผิดพลาดได้

3.2 การตั้งค่าเริ่มต้นให้กับไดเร็กซ์สามดี

ในการตั้งค่าเริ่มต้นให้กับไดเร็กซ์สามดีนั้น มีขั้นตอนการทำงานแยกเป็นข้อ ๆ ได้ดังนี้

1. สร้างพอยน์เตอร์ชี้ไปยัง IDirect3D9 ซึ่งเป็นอินเตอร์เฟสที่ใช้ในการหาข้อมูลของฮาร์ดแวก่อนที่จะสร้าง IDirect3DDevice9
2. ตรวจสอบความสามารถของฮาร์ดแวร์หรือการ์ดจอจาก D3DCAPS9 โดยจะต้องตรวจสอบความสามารถในการทำ Hardware vertex processing ก่อนที่จะสร้าง IDirect3DDevice9
3. ตั้งค่าเริ่มต้นให้กับ D3DPRESENT_PARAMETERS ซึ่งเป็นสตรัคท์ที่มีตัวแปรที่ใช้กำหนดลักษณะของ IDirect3DDevice9
4. สร้าง IDirect3DDevice9 ตามโครงสร้างของ D3DPRESENT_PARAMETERS โดย IDirect3DDevice นี้จะเป็นวัตถุที่เป็นตัวแทนของอุปกรณ์ฮาร์ดแวร์ที่ถูกนำมาใช้สำหรับแสดงภาพ 3 มิติ

3.2.1 การได้มาซึ่งอินเตอร์เฟส IDirect3D9

การตั้งค่าเริ่มต้นให้กับไดเร็กซ์สามดีจะเริ่มต้นโดยการได้มาซึ่งพอยน์เตอร์ที่ชี้ไปยังอินเตอร์เฟส IDirect3D9 โดยใช้ฟังก์ชันตามตารางที่ 3-1

```
IDirect3D9* d3d9;
_d3d9 = Direct3DCreate9(D3D_SDK_VERSION);
```

ตารางที่ 3-1 แสดงการได้มาซึ่งอินเตอร์เฟส IDirect3D9

วัตถุ IDirect3D9 จะถูกใช้งาน 2 อย่าง คือ เป็นตัวระบุอุปกรณ์ และ ใช้ในการสร้างวัตถุ IDirect3DDevice9 โดยจะสามารถระบุได้ทั้งความสามารถ โหมดในการแสดงภาพ รูปแบบที่จะใช้ และข้อมูลอื่น ๆ เกี่ยวกับการ์ดจอที่ใช้อยู่

3.2.2 ตรวจสอบความสามารถในการทำฮาร์ดแวร์เวอร์เท็กซ์โพรเซสซิง (Hardware Vertex Processing)

ในการสร้างวัตถุ IDirect3DDevice9 นั้น จะต้องมีการกำหนดชนิดของการทำเวอร์เท็กซ์โพรเซสซิง (vertex processing) ที่จะใช้เสียก่อน โดยมี 2 ชนิด คือ ฮาร์ดแวร์เวอร์เท็กซ์โพรเซสซิง (Hardware vertex processing) และ ซอฟท์แวร์เวอร์เท็กซ์โพรเซสซิง (Software vertex processing) ซึ่งมีความแตกต่างกัน ดังนี้

ฮาร์ดแวร์เวอร์เท็กซ์โพรเซสซิง – จะมีการทำงานที่เร็วกว่า แต่ต้องใช้ความสามารถของการ์ดจอซึ่งจะมีความสามารถนี้การ์ดจอบางรุ่นเท่านั้น

ซอฟท์แวร์เวอร์เท็กซ์โพรเซสซิง – จะใช้การสร้างเวอร์เท็กซ์โพรเซสซิงจำลองขึ้นมาทำงานในหน่วยความจำ ซึ่งจะทำงานได้ช้ากว่าฮาร์ดแวร์เวอร์เท็กซ์โพรเซสซิงค่อนข้างมาก ในการตรวจสอบนั้นจะต้องสร้างวัตถุ D3DCAPS9 ขึ้นมาก่อนโดยใช้ฟังก์ชันตามตารางที่ 3-2

```
HRESULT IDirect3D9::GetDeviceCaps (
        UINT Adapter,
        D3DDEVTYPE DeviceType,
        D3DCAPS9 *pCaps)
```

ตารางที่ 3-2 แสดงฟังก์ชัน GetDeviceCaps

ซึ่งมีตัวอย่างการใช้งาน ดังตารางที่ 3-3

```
D3DCAPS caps;
_d3d9 ->GetDeviceCaps (
        D3DADAPTER_DEFAULT,
        deviceType,
        &caps);
int vp = 0;
if (caps.DevCaps & D3DDEVCAPS_HWTRANSFORMANDLIGHT)
{
        vp = D3DCREATE_HARDWARE_VERTEXPROCESSING;
}
else
{
        vp = D3DCREATE_SOFTWARE_VERTEXPROCESSING;
}
```

ตารางที่ 3-3 แสดงตัวอย่างการใช้งานฟังก์ชัน GetDeviceCaps

จากตารางที่ 3-3 จะเห็นได้ว่าตัวแปร vp จะถูกใช้ในการเก็บชนิดของเวอร์เท็กซ์โพรเซสซิงที่ใช้

3.2.3 กำหนดค่าให้ตัวแปรในสตรัคท์ D3DPRESENT_PARAMETERS

สตรัคท์ D3DPRESENT_PARAMETERS มีตัวแปรดังตารางที่ 3-4

```
typedef struct _D3DPRESENT_PARAMETERS_ {
UINT BackBufferWidth;
UINT BackBufferHeight;
D3DFORMAT BackBufferFormat;
UINT BackBufferCount;
D3DMULTISAMPLE_TYPE MultiSampleType;
DWORD MultiSampleQuality;
D3DSWAPEFFECT SwapEffect;
HWND hDeviceWindow;
BOOL Windowed;
BOOL EnableAutoDepthStencil;
D3DFORMAT AutoDepthStencilFormat;
DWORD Flags;
UINT FullScreen_RefreshRateInHz;
UINT PresentationInterval;
} D3DPRESENT_PARAMETERS;
```

ตารางที่ 3-4 แสดงโครงสร้างของสตรัคท์ D3DPRESENT_PARAMETERS

ตัวอย่างของการกำหนดค่าตัวแปร D3DPRESENT_PARAMETERS แสดงไว้ตามตารางที่ 3-5

```
D3DPRESENT_PARAMETERS d3dpp;
d3dpp.BackBufferWidth = 800;
d3dpp.BackBufferHeight = 600;
d3dpp.BackBufferFormat = D3DFMT_A8R8G8B8; d3dpp.BackBufferCount = 1;
d3dpp.MultiSampleType = D3DMULTISAMPLE_NONE;
d3dpp.MultiSampleQuality = 0;
d3dpp.SwapEffect = D3DSWAPEFFECT_DISCARD;
d3dpp.hDeviceWindow = hwnd;
d3dpp.Windowed = false;
d3dpp.EnableAutoDepthStencil = true;
d3dpp.AutoDepthStencilFormat = D3DFMT_D24S8;
d3dpp.Flags = 0;
d3dpp.FullScreen_RefreshRateInHz = D3DPRESENT_RATE_DEFAULT;
d3dpp.PresentationInterval = D3DPRESENT_INTERVAL_IMMEDIATE;
```

ตารางที่ 3-5 แสดงตัวอย่างของการกำหนดค่าให้ตัวแปรใน D3DPRESENT_PARAMETERS

3.2.4 สร้างอินเตอร์เฟส IDirect3DDevice9

หลังจากกำหนดค่าตัวแปรต่าง ๆ ของ D3DPRESENT_PARAMETERS แล้ว ก็จะสามารถสร้างวัตถุ IDirect3DDevice9 ด้วยคำสั่งในตารางที่ 3-6

```
HRESULT IDirect3D9::CreateDevice(
UINT Adapter,
D3DDEVTYPE DeviceType,
HWND hFocusWindow,
DWORD BehaviorFlags,
D3DPRESENT_PARAMETERS *pPresentationParameters,
IDirect3DDevice9** ppReturnedDeviceInterface
);
```

ตารางที่ 3-6 แสดงฟังก์ชัน CreateDevice ที่ใช้ในการสร้างอินเตอร์เฟส IDirect3DDevice9

ตัวอย่างของการสร้าง IDirect3DDevice9 แสดงได้ตามตารางที่ 3-7

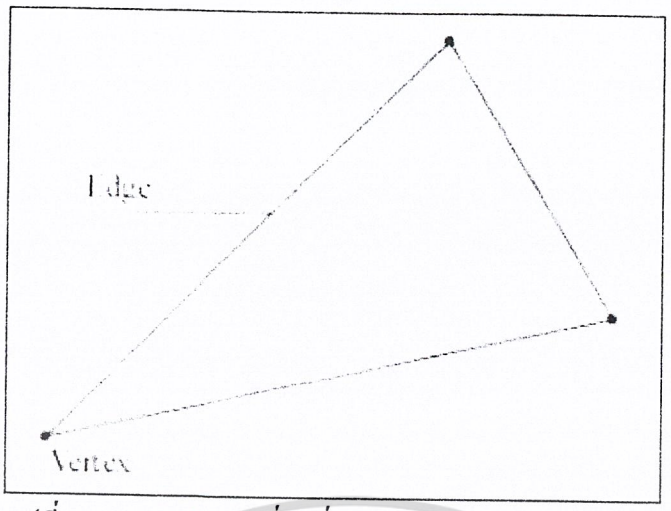
```
IDirect3DDevice9* device = 0;
hr = d3d9->CreateDevice(
D3DADAPTER_DEFAULT,
D3DDEVTYPE_HAL,
hwnd,
D3DCREATE_HARDWARE_VERTEXPROCESSING,
&d3dpp,
&device);
if( FAILED(hr) )
{
::MessageBox(0, "CreateDevice() - FAILED", 0, 0);
return 0;
}
```

ตารางที่ 3-7 แสดงตัวอย่างการสร้างอินเตอร์เฟสของ IDirect3DDevice9

3.3 การแสดงรูปทรง 3 มิติ

ฉาก (scene) ในคอมพิวเตอร์กราฟฟิก เกิดจากการรวมกันของวัตถุ และรูปทรง 3 มิติ โดยวัตถุจะถูกแสดงด้วยเครือข่ายสามเหลี่ยม (triangle mesh) เราจะเรียกรูปสามเหลี่ยมแต่ละรูปว่า รูปพื้นฐาน (primitive) นอกจากนี้ยังมีรูปพื้นฐานอื่น ๆ อีก เช่น เส้นตรง จุด แต่รูปพื้นฐานเหล่านี้ไม่สามารถนำมาสร้างวัตถุ 3 มิติได้

จุดที่เกิดจากการตัดกันของเอจ (edge) 2 เส้น จะเรียกว่า เวอร์เท็กซ์ (vertex) โดยที่รูปสามเหลี่ยมหนึ่ง ๆ จะมีเวอร์เท็กซ์อยู่ด้วยกัน 3 จุด ซึ่งแต่ละจุดจะเชื่อมต่อถึงกันด้วยเอจ ดังรูปที่ 3-1



รูปที่ 3-1 แสดงรูปสามเหลี่ยมที่ประกอบด้วยเวอร์เท็กซ์และเอจ

ในการสร้างรูปแบบเวอร์เท็กซ์ของตัวเอง จะต้องสร้างสตรัคท์ที่ใช้สำหรับเก็บข้อมูลเวอร์เท็กซ์ที่เราใช้ โดยตารางที่ 3-8 จะแสดงโค้ดในการกำหนดสตรัคท์ที่แตกต่างกัน 2 แบบ

```
struct ColorVertex
{
float _x, _y, _z;
DWORD _color;
};
struct NormalTexVertex
{
float _x, _y, _z;
float _nx, _ny, _nz;
float _u, _v;
};
```

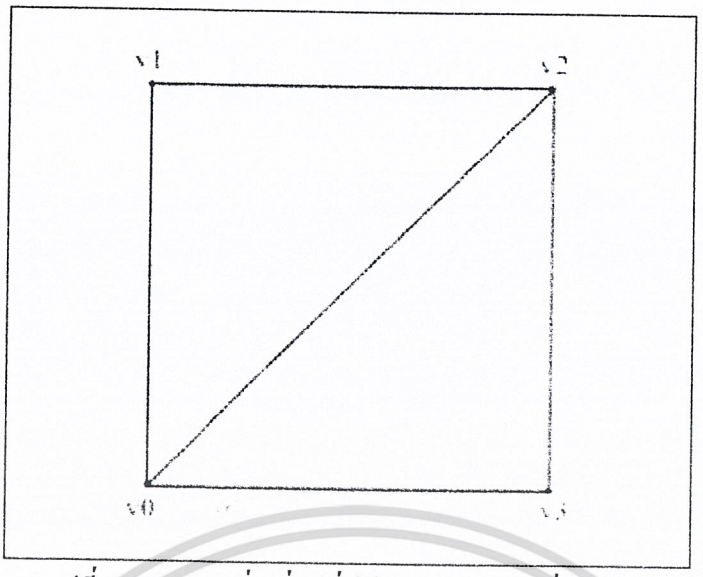
ตารางที่ 3-8 แสดงตัวอย่างการกำหนดสตรัคท์ให้กับเวอร์เท็กซ์

หลังจากสร้างสตรัคท์เสร็จแล้ว จะต้องมีการกำหนดรูปแบบที่เราจะเลือกใช้ผ่านทางแฟลก FVF (Flexible Vertex Format) โดยจะต้องกำหนดให้ตรงกันกับสตรัคท์ที่ได้สร้างขึ้น ดังตารางที่ 3-9

```
#define FVF_COLOR (D3DFVF_XYZ | D3DFVF_DIFFUSE)
#define FVF_NORMAL_TEX (D3DFVF_XYZ | D3DFVF_NORMAL |
D3DFVF_TEX1)
```

ตารางที่ 3-9 แสดงตัวอย่างการกำหนด FVF

การสร้างรูปทรง 3 มิติที่เกิดจากรูปสามเหลี่ยมหลาย ๆ รูปรวมกันนั้น จะต้องสร้างอาร์เรย์ของเวอร์เท็กซ์ไว้เก็บเวอร์เทกซ์ต่าง ๆ เช่นถ้าเราต้องการสร้างรูปสี่เหลี่ยม ดังรูปที่ 3-2



รูปที่ 3-2 แสดงรูปสี่เหลี่ยมที่สร้างจากรูปสามเหลี่ยม 2 รูป

เราจะต้องกำหนดอาร์เรย์ของเวอร์เท็กซ์ ดังตารางที่ 3-10

```
Vertex vertexList[4] = {v0, v1, v2, v3};
```

ตารางที่ 3-10 แสดงการกำหนดอาร์เรย์ของเวอร์เท็กซ์

แต่การสร้างเพียงแค่อาร์เรย์ของเวอร์เท็กซ์ไม่เพียงพอต่อการสร้างรูปสี่เหลี่ยม ดังนั้นจะต้องมีการกำหนดอาร์เรย์ของอินเด็กซ์ขึ้นมาเพื่อใช้ในการสร้างรูปสามเหลี่ยมจากเวอร์เท็กซ์ที่มีอยู่ ดังตารางที่ 3-11

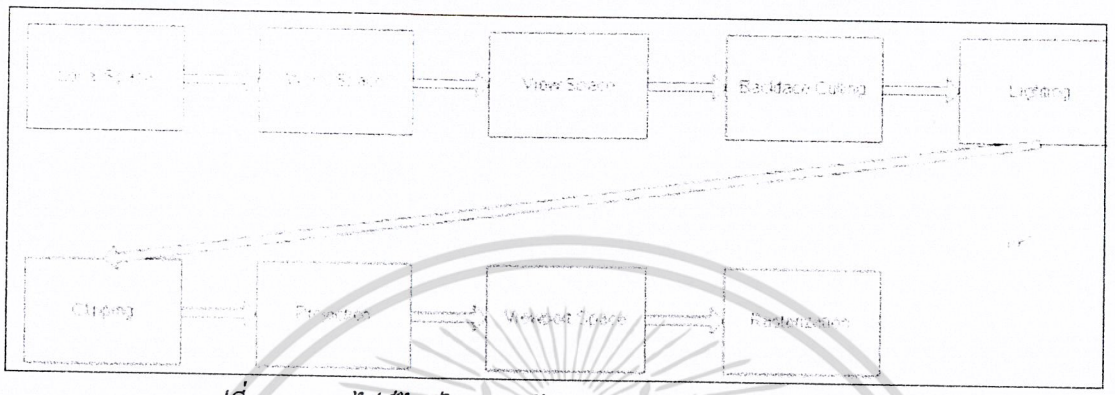
```
WORD indexList[6] = {0, 1, 2, // รูปสามเหลี่ยมรูปที่ 1
                     0, 2, 3}; // รูปสามเหลี่ยมรูปที่ 2
```

ตารางที่ 3-11 แสดงการกำหนดอาร์เรย์ของอินเด็กซ์

โดยข้อมูลที่อยู่ในอาร์เรย์ของอินเด็กซ์จะเป็นตัวชี้ไปยังข้อมูลที่อยู่ในอาร์เรย์เวอร์เท็กซ์อีกทีหนึ่ง การสร้างกล้องเสมือน (Virtual Camera) กล้องถือได้ว่าเป็นตัวกำหนดมุมมองในเวิร์ล (world) ให้ผู้ใช้แอพพลิเคชันเห็น โดยที่กล้องจะมีคุณสมบัติต่าง ๆ เช่น ตำแหน่ง (position), ทิศทางที่กล้องหันไป (orientation), ขอบเขตการมองเห็น (volume of space) เป็นต้น ซึ่งคุณสมบัติต่าง ๆ เหล่านี้จะเป็นตัวกำหนดภาพที่ผู้ใช้แอพพลิเคชันมองเห็น

3.4 ไปป์ไลน์ในการสร้างภาพ 3 มิติ (Rendering Pipeline)

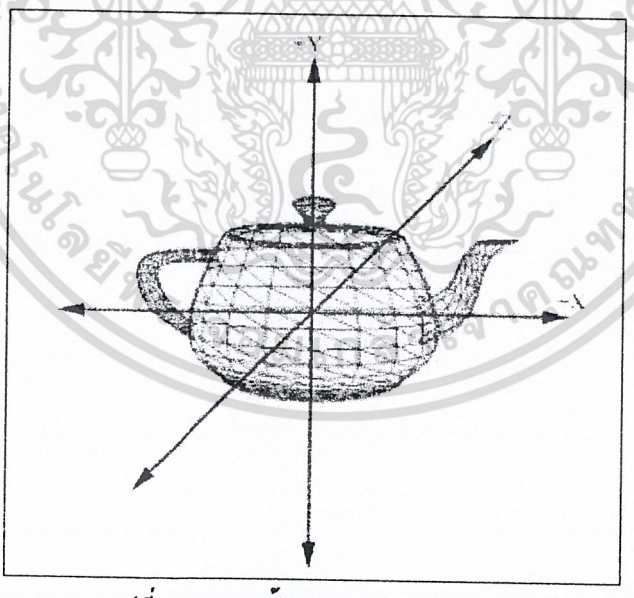
ไปป์ไลน์ในการสร้างภาพ 3 มิติ (Rendering Pipeline) คือ อนุกรมของการทำงานตั้งแต่การสร้างวัตถุใน 3 มิติ จนถึงการนำภาพออกไปแสดงบนหน้าจอในลักษณะของภาพ 2 มิติ ซึ่งไดเร็กซ์เอ็กซ์มีไปป์ไลน์ในการสร้างภาพ 3 มิติ ดังรูปที่ 3-3



รูปที่ 3-3 แสดงไปป์ไลน์ในการสร้างภาพ 3 มิติ (Rendering Pipeline)

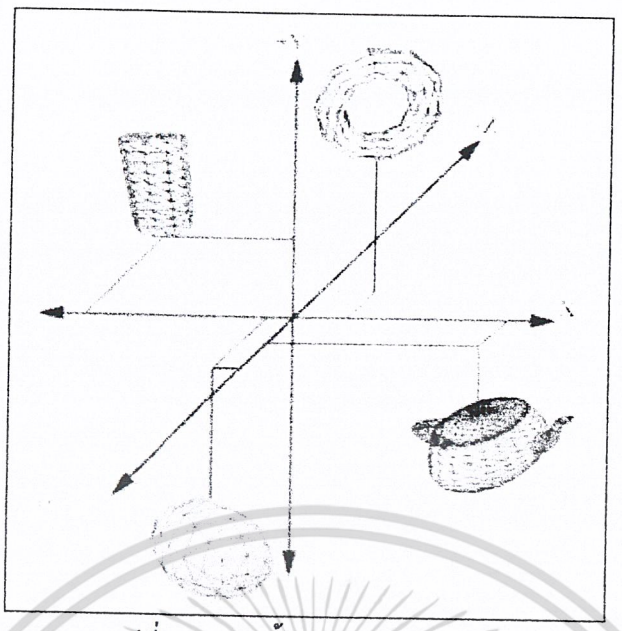
คำอธิบายแต่ละขั้นตอนของไปป์ไลน์

- โลคอลสเปส (Local space) เป็นขั้นตอนในการกำหนดตำแหน่งของสามเหลี่ยมที่ประกอบกันขึ้นเป็นวัตถุในระบบแกนโคออร์ดิเนตของวัตถุนั้น ๆ ดังรูปที่ 3-4



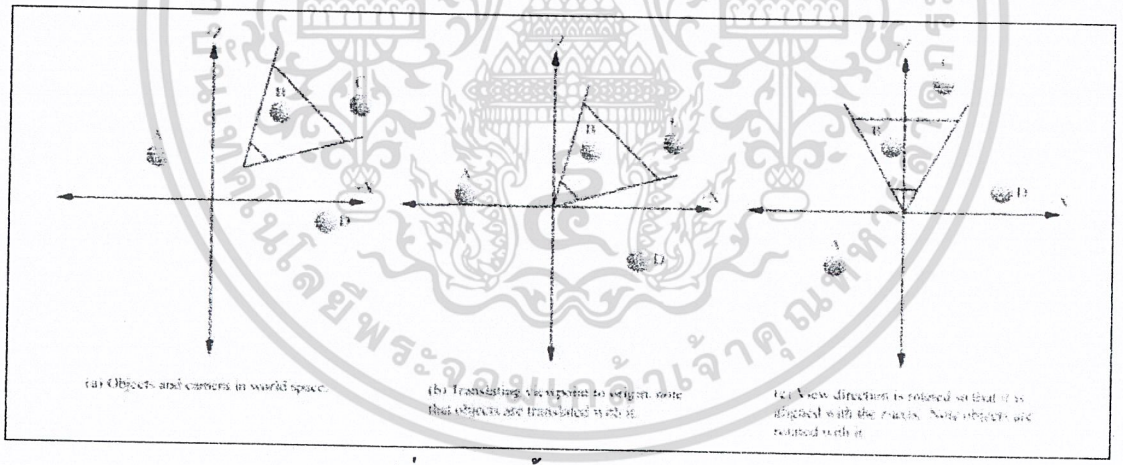
รูปที่ 3-4 แสดงขั้นตอนของโลคอลสเปส

- เวิร์ลสเปส (World space) เป็นขั้นตอนในการกำหนดตำแหน่งของวัตถุซึ่งถูกสร้างในขั้นตอนของโลคอลสเปสให้อยู่ในตำแหน่งที่ต้องการภายในระบบแกนโคออร์ดิเนตของเวิร์ล ดังรูปที่ 3-5



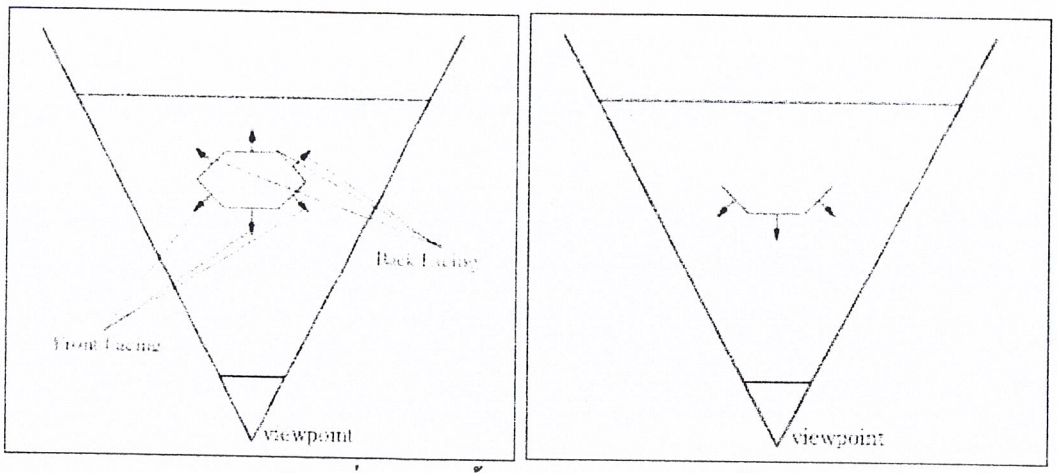
รูปที่ 3-5 แสดงขั้นตอนของเวิร์ลสเปส

- วิวสเปส (View space) เป็นขั้นตอนในการเปลี่ยนตำแหน่งกล้องให้มาอยู่ที่จุดออริจิน (0,0,0) และให้หน้ากล้องหันไปทางทิศ +z โดยการเปลี่ยนตำแหน่งกล้องนี้จะทำให้ตำแหน่งของวัตถุเปลี่ยนแปลงตามไปด้วย ดังรูปที่ 3-6



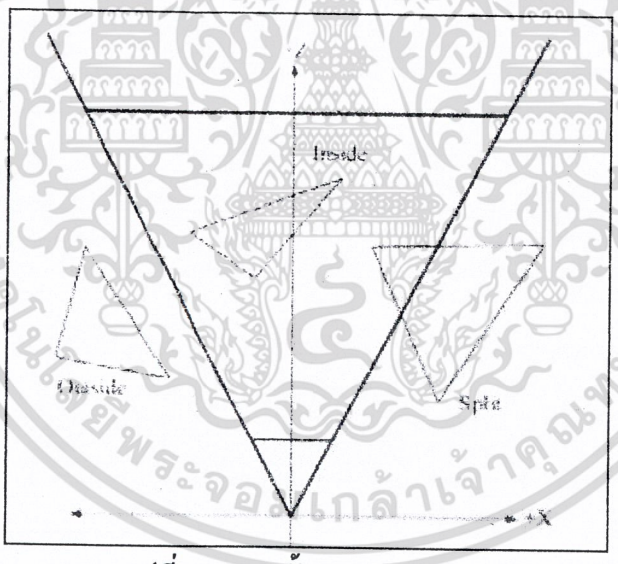
รูปที่ 3-6 แสดงขั้นตอนของวิวสเปส

- แบ็คเฟสคัลลิง (Backface Culling) เป็นขั้นตอนการตัดส่วนของวัตถุ (สามเหลี่ยม) ที่หันหลังให้กับกล้องซึ่งเป็นส่วนที่ไม่สามารถมองเห็นได้ออกไป ดังรูปที่ 3-7



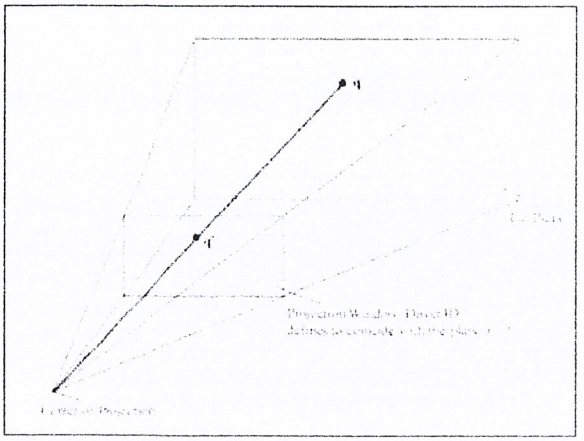
รูปที่ 3-7 แสดงขั้นตอนของแบ็คเฟสคัลลิง

- ไลท์ทิง (Lighting) เป็นขั้นตอนการคำนวณเรื่องแสงเพื่อให้ภาพที่ได้ออกมามีความสมจริงมากขึ้น
- คลิปปิง (Clipping) เป็นขั้นตอนการตัดวัตถุหรือส่วนของวัตถุที่อยู่ภายนอกขอบเขตการมองเห็น (Viewing Volume) ออกไป ดังรูปที่ 3-8



รูปที่ 3-8 แสดงขั้นตอนของคลิปปิง

- โปรเจคชัน (Projection) เป็นขั้นตอนในการทำให้วัตถุที่อยู่ใน 3 มิติ เกิดเป็นภาพ 2 มิติบนฉาก ซึ่งการทำโปรเจคชันนั้น มีอยู่หลายแบบ เช่น แอ็กโซโนเมทริก (Axonometric), ออบลิค (Oblique), เพอร์สเปคทีพ (Perspective) เป็นต้น โดยรูปที่ 3-9 เป็นการทำโปรเจคชันแบบเพอร์สเปคทีพ

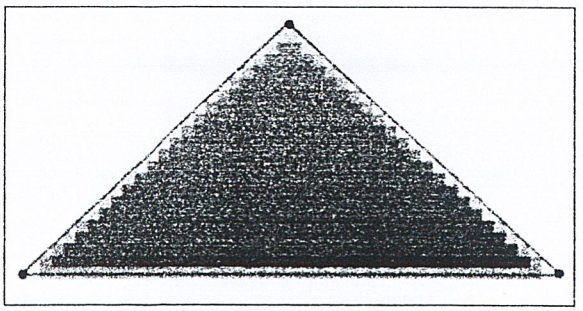


รูปที่ 3-9 แสดงการทำโปรเจคชันแบบ เพอร์สเปคทีฟ (Perspective)

- วิวพอร์ตทรานสฟอร์ม (Viewport Transform) เป็นขั้นตอนการแปลงจากระบบแกนโคออร์ดิเนตของโปรเจคชันวินโดว์ (Projection window) ที่เกิดจากการทำโปรเจคชัน ให้กลายเป็นระบบแกนโคออร์ดิเนตของฉากสี่เหลี่ยมบนจอ ที่เรียกว่า วิวพอร์ต (Viewport) ดังรูปที่ 3-10

รูปที่ 3-10 แสดงขั้นตอนการทำวิวพอร์ตทรานสฟอร์ม

- การปรับคุณภาพรูป (Rasterization) เป็นขั้นตอนการใช้เทคนิคต่าง ๆ มาปรับคุณภาพของรูปให้ดูเรียบและสวยงามขึ้น ดังรูปที่ 3-11



รูปที่ 3-11 แสดงขั้นตอนการปรับคุณภาพรูป (Rasterization)

บทที่ 4
การสร้างรูป 3 มิติด้วย Direct3D

4.1 เวอร์เท็กซ์และอินเด็กซ์บัฟเฟอร์ (Vertex Buffer & Index Buffer)

ในการสร้างภาพ 3 มิตินั้น จะต้องสร้างโดยนำรูปสามเหลี่ยมซึ่งเป็นรูปพื้นฐานมาประกอบกันจนเกิดเป็นรูปที่ต้องการ และในการสร้างรูปสามเหลี่ยมแต่ละรูปนั้น จะต้องมีการกำหนดเวอร์เท็กซ์ให้กับรูปสามเหลี่ยมนั้น โดยเวอร์เท็กซ์ที่กำหนดจะเก็บอยู่ในหน่วยความจำที่เรียกว่า เวอร์เท็กซ์บัฟเฟอร์ (Vertex Buffer)

นอกจากนี้ในการสร้างรูป 3 มิติที่ต้องมีการใช้เวอร์เท็กซ์ร่วมกัน จะต้องมีการใช้อินเด็กซ์บัฟเฟอร์ (Index Buffer) สำหรับรูปสามเหลี่ยมแต่ละรูปเพื่อใช้เป็นที่เก็บค่าพอยน์เตอร์ที่ชี้ไปยังเวอร์เท็กซ์ต่าง ๆ ที่ถูกเก็บไว้ในเวอร์เท็กซ์บัฟเฟอร์

4.1.1 ความแตกต่างระหว่างเวอร์เท็กซ์บัฟเฟอร์และอินเด็กซ์บัฟเฟอร์

เวอร์เท็กซ์บัฟเฟอร์จะใช้เก็บจุดต่าง ๆ ที่ประกอบกันขึ้นเป็นวัตถุ โดยจุดที่เก็บนั้นจะเป็นจุดที่ไม่ซ้ำกัน เรียกว่า เวอร์เท็กซ์

ส่วนอินเด็กซ์บัฟเฟอร์นั้น จะใช้เก็บจุดยอดของสามเหลี่ยมทุกรูปที่ประกอบกันขึ้นเป็นวัตถุ ซึ่งสามเหลี่ยม 1 รูป ก็จะใช้อินเด็กซ์บัฟเฟอร์ 3 ช่อง ซึ่งแต่ละช่องจะเก็บพอยน์เตอร์ที่ชี้ไปยังเวอร์เท็กซ์ที่ใช้ซึ่งเก็บอยู่ในเวอร์เท็กซ์บัฟเฟอร์อีกทีหนึ่ง

4.1.2 การสร้างเวอร์เท็กซ์และอินเด็กซ์บัฟเฟอร์

ในการสร้างเวอร์เท็กซ์และอินเด็กซ์บัฟเฟอร์นั้น จะใช้ฟังก์ชันของไดเร็กซ์สามดี ดังตารางที่ 4-1

```
HRESULT IDirect3DDevice9::CreateVertexBuffer(
UINT Length,
DWORD Usage,
DWORD FVF,
D3DPOOL Pool
IDirect3DVertexBuffer9** ppVertexBuffer,
HANDLE* pSharedHandle
);
HRESULT IDirect3DDevice9::CreateIndexBuffer(
UINT Length,
DWORD Usage,
D3DFORMAT Format,
D3DPOOL Pool,
IDirect3DIndexBuffer9** ppIndexBuffer,
HANDLE* pSharedHandle
);
```

ตารางที่ 4-1 แสดงฟังก์ชัน CreateVertexBuffer และฟังก์ชัน CreateIndexBuffer

สำหรับตัวอย่างในการสร้างเวอร์เท็กซ์บัฟเฟอร์ที่ใช้เก็บเวอร์เท็กซ์จำนวน 8 เวอร์เท็กซ์ จะแสดงได้ดังโค้ดในตารางที่ 4-2

```
IDirect3DVertexBuffer9* vb;
_device->CreateVertexBuffer(
8 * sizeof( Vertex ),
0,
D3DFVF_XYZ,
D3DPOOL_MANAGED,
&vb,
0);
```

ตารางที่ 4-2 แสดงตัวอย่างการสร้างเวอร์เท็กซ์บัฟเฟอร์ด้วยฟังก์ชัน CreateVertexBuffer

ตัวอย่างในตารางที่ 4-3 เป็นการสร้างอินเด็กซ์บัฟเฟอร์ที่ใช้เก็บอินเด็กซ์จำนวน 36 อินเด็กซ์ ซึ่งแต่ละอินเด็กซ์จะมีขนาด 16 บิต

```
IDirect3DIndexBuffer9* ib;
_device->CreateIndexBuffer(
36 * sizeof( WORD ),
D3DUSAGE_DYNAMIC |
D3DUSAGE_WRITEONLY,
D3DFMT_INDEX16,
D3DPOOL_MANAGED,
&ib,
0);
```

ตารางที่ 4-3 แสดงตัวอย่างการสร้างอินเด็กซ์บัฟเฟอร์ด้วยฟังก์ชัน CreateIndexBuffer

หลังจากได้สร้างเวอร์เท็กซ์บัฟเฟอร์และอินเด็กซ์บัฟเฟอร์ตามขนาดที่ต้องการใช้แล้ว ก็จะเข้าสู่ขั้นตอนของการกำหนดค่าเวอร์เท็กซ์และอินเด็กซ์ที่จะเก็บลงไปในเวอร์เท็กซ์บัฟเฟอร์และอินเด็กซ์บัฟเฟอร์ โดยการจะเข้าไปอ่านหรือเขียนข้อมูลในบัฟเฟอร์ดังกล่าว จะต้องใช้ฟังก์ชัน Lock เพื่อที่จะเอาค่าพอยน์เตอร์ที่ชี้ไปยังบัฟเฟอร์ส่วนที่ต้องการอ่านหรือเขียนมาเสียก่อน โดยฟังก์ชัน Lock มีโครงสร้างดังตารางที่ 4-4

```
HRESULT
IDirect3DVertexBuffer9::Lock(
UINT OffsetToLock,
UINT SizeToLock,
BYTE** ppbData,
DWORD Flags
);
HRESULT IDirect3DIndexBuffer9::Lock(
UINT OffsetToLock,
UINT SizeToLock,
BYTE** ppbData,
DWORD Flags
);
```

ตารางที่ 4-4 แสดงฟังก์ชัน Lock

และหลังจากที่ได้เขียนหรืออ่านค่าจากบัฟเฟอร์ที่ทำการ Lock เสร็จเรียบร้อยแล้ว จะต้องเรียกใช้ฟังก์ชัน Unlock ด้วย ดังแสดงในตัวอย่างโค้ดตามตารางที่ 4-5

```
Vertex* vertices;
_vb->Lock(0, 0, (void**)&vertices, 0);
vertices[0] = Vertex(-1.0f, 0.0f, 2.0f);
vertices[1] = Vertex( 0.0f, 1.0f, 2.0f);
vertices[2] = Vertex( 1.0f, 0.0f, 2.0f);
_vb->Unlock();
```

ตารางที่ 4-5 แสดงตัวอย่างการกำหนดค่าในเวอร์เท็กซ์บัฟเฟอร์ร่วมกับฟังก์ชัน Lock และ Unlock

4.2 การวาดภาพ (Render)

ก่อนที่จะวาดภาพ (Render) ในไดเร็กซ์สามดีนั้น จะต้องมีการกำหนดค่าให้กับสถานะของการวาดภาพ (Render State) ก่อน โดยไดเร็กซ์สามดีจะมีสถานะของการวาดภาพให้ปรับได้หลากหลายขึ้นอยู่กับความต้องการของโปรแกรมเมอร์ โดยสถานะของการวาดภาพดังกล่าวจะมีค่าดีฟอลต์อยู่ ซึ่งถ้าไม่มีการปรับค่าใด ๆ ก็จะเรียกใช้ค่าดีฟอลต์นั้น สำหรับวิธีการปรับค่าสถานะของการวาดภาพนั้นจะใช้ฟังก์ชัน SetRenderState ดังแสดงตามตารางที่ 4-6

```
HRESULT IDirect3DDevice9::SetRenderState(
D3DRENDERSTATETYPE State,
DWORD Value
);
```

ตารางที่ 4-6 แสดงฟังก์ชัน SetRenderState

ตัวอย่างการปรับค่าสถานะของการวาดภาพแสดงไว้ในตารางที่ 4-7

```
m_device->SetRenderState(D3DRS_FILLMODE, D3DFILL_WIREFRAME);
```

ตารางที่ 4-7 แสดงตัวอย่างการปรับค่าสถานะของการวาดภาพ

4.2.1 ขั้นตอนการเตรียมข้อมูลก่อนการวาดภาพ

หลังจากได้สร้างและกำหนดค่าให้กับเวอร์เท็กซ์บัฟเฟอร์และอินเด็กซ์บัฟเฟอร์แล้ว ก็ถึงขั้นตอนการเตรียมข้อมูลก่อนที่จะวาดภาพ ซึ่งมี 3 ขั้นตอน ดังนี้

1) ปรับรูปแบบข้อมูลที่ต้องการวาด (Set stream source) ซึ่งขั้นตอนนี้จะแปลงข้อมูลในเวอร์เท็กซ์บัฟเฟอร์ให้กลายเป็นสตรีมของข้อมูลเพื่อให้ง่ายต่อการนำรูป 3 มิติต่าง ๆ เข้าไปจัดการในไปป์ไลน์ของการวาดภาพ

โดยฟังก์ชันที่ใช้ คือ SetStreamSource ซึ่งมีรูปแบบดังตารางที่ 4-8

```
HRESULT IDirect3DDevice9::SetStreamSource(
UINT StreamNumber,
IDirect3DVertexBuffer9* pStreamData,
UINT OffsetInBytes,
UINT Stride
);
```

ตารางที่ 4-8 แสดงฟังก์ชัน SetStreamSource

ตัวอย่างการเรียกใช้ฟังก์ชัน แสดงตามโค้ดในตารางที่ 4-9

```
m_device->SetStreamSource( 0, vb, 0, sizeof( Vertex ) );
```

ตารางที่ 4-9 แสดงตัวอย่างการใช้ฟังก์ชัน SetStreamSource

2) กำหนดรูปแบบของเวอร์เท็กซ์ (Set Vertex Format) โดยหลังจากการกำหนดรูปแบบเวอร์เท็กซ์แล้ว การวาดภาพครั้งต่อ ๆ มา จะใช้รูปแบบของเวอร์เท็กซ์ตามที่ได้กำหนดไว้ล่าสุด ซึ่งตารางที่ 4-10 เป็นตัวอย่างการกำหนดรูปแบบของเวอร์เท็กซ์

```
m_device->SetFVF( D3DFVF_XYZ | D3DFVF_DIFFUSE | D3DFVF_TEX1 );
```

ตารางที่ 4-10 แสดงการกำหนดรูปแบบเวอร์เท็กซ์ (FVF)

3) กำหนดอินเด็กซ์บัฟเฟอร์ (Set the index buffer) ในกรณีที่มีการใช้อินเด็กซ์บัฟเฟอร์จะต้องมีการกำหนดอินเด็กซ์บัฟเฟอร์ที่จะใช้ในการวาดด้วย โดยที่จะกำหนดอินเด็กซ์บัฟเฟอร์ในการวาดได้ครั้งละ 1 อินเด็กซ์บัฟเฟอร์เท่านั้น ซึ่งตารางที่ 4-11 เป็นตัวอย่างการกำหนดอินเด็กซ์บัฟเฟอร์

```
m_device->SetIndices( m_ib );
```

ตารางที่ 4-11 แสดงการกำหนดอินเด็กซ์บัฟเฟอร์

4.2.2 ขั้นตอนในการวาดภาพด้วยเวอร์เท็กซ์บัฟเฟอร์และอินเด็กซ์บัฟเฟอร์

หลังจากได้เตรียมข้อมูลก่อนการวาดภาพแล้ว ก็จะมาถึงขั้นตอนของการวาดภาพ 3 มิติ โดยการใช้ฟังก์ชัน DrawPrimitive หรือ DrawIndexedPrimitive โดย 2 ฟังก์ชันนี้จะนำค่าเวอร์เท็กซ์ที่อยู่ในเวอร์เท็กซ์สตรีมและค่าอินเด็กซ์ที่อยู่ในอินเด็กซ์สตรีมออกมาใช้วาดตามลำดับ

ตัวอย่างของการเรียกใช้ฟังก์ชัน DrawPrimitive แสดงดังโค้ดในตารางที่ 4-12 และตารางที่ 4-13

```
HRESULT IDirect3DDevice9::DrawPrimitive(
D3DPRIMITIVETYPE PrimitiveType,
UINT StartVertex,
UINT PrimitiveCount
);
```

ตารางที่ 4-12 แสดงฟังก์ชัน DrawPrimitive

```
m_device->DrawPrimitive( D3DPT_TRIANGLELIST, 0, 4);
```

ตารางที่ 4-13 แสดงตัวอย่างการใช้ฟังก์ชัน DrawPrimitive

ตัวอย่างของการใช้งานฟังก์ชัน DrawIndexedPrimitive แสดงได้ดังโค้ดในตารางที่ 4-14 และตารางที่ 4-15

```
HRESULT
IDirect3DDevice9::DrawIndexedPrimitive(
D3DPRIMITIVETYPE Type,
INT BaseVertexIndex,
UINT MinIndex,
UINT NumVertices,
UINT StartIndex,
UINT PrimitiveCount
);
```

ตารางที่ 4-14 แสดงฟังก์ชัน DrawIndexedPrimitive

```
m_device->DrawIndexedPrimitive(D3DPT_TRIANGLELIST, 0, 0, 8, 0, 12);
```

ตารางที่ 4-15 แสดงตัวอย่างการใช้ฟังก์ชัน DrawIndexedPrimitive

ซึ่งโปรแกรมเมอร์จะเลือกใช้ฟังก์ชันใดก็ได้ ขึ้นอยู่กับความเหมาะสม

นอกจากนี้ การเรียกใช้ฟังก์ชันในการวาดภาพทั้ง DrawPrimitive หรือ DrawIndexedPrimitive จะต้องเรียกใช้ระหว่างฟังก์ชัน BeginScene และ EndScene ดังแสดงในตารางที่ 4-16

```
m_device->BeginScene();
m_device->DrawPrimitive(...);
m_device->EndScene();
```

ตารางที่ 4-16 แสดงการใช้ฟังก์ชัน DrawPrimitive ระหว่างฟังก์ชัน BeginScene และ EndScene

4.2.3 รูปทรงมาตรฐานของ D3DX

ในไลบรารีของ D3DX จะมีรูปทรงมาตรฐานที่สามารถเรียกใช้ได้เลยอยู่ด้วยกัน 6 ฟังก์ชัน ดังนี้

- D3DXCreateBox
- D3DXCreateSphere
- D3DXCreateCylinder
- D3DXCreateTeapot
- D3DXCreatePolygon
- D3DXCreateTorus

โดยรูปทรงมาตรฐานทั้งหมดนี้จะมีโครงสร้างเป็นแบบ ID3DXMesh โดยวิธีการเรียกใช้ฟังก์ชันเหล่านี้ แสดงได้ดังตัวอย่างในตารางที่ 4-17

```
HRESULT D3DXCreateTeapot(
LPDIRECT3DDEVICE9 pDevice,
LPD3DXMESH* ppMesh,
LPD3DXBUFFER* ppAdjacency
);

ID3DXMesh* mesh = 0;
D3DXCreateTeapot(_device, &mesh, 0);

_device->BeginScene();
mesh->DrawSubset(0);
_device->EndScene();

_mesh->Release();
_mesh = 0;
```

ตารางที่ 4-17 แสดงตัวอย่างการสร้างรูปมาตรฐานกาน้ำ (Teapot)

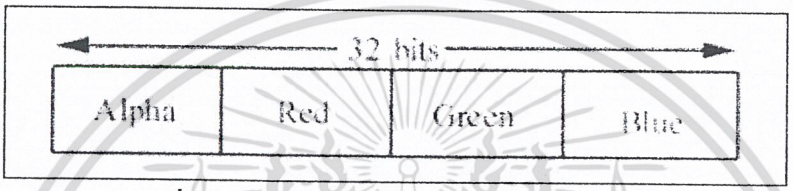
โดยจากตารางที่ 4-17 จะต้องสร้างตัวแปรที่เป็นพอยน์เตอร์ไปยัง ID3DXMesh มารองรับค่าที่ได้จากฟังก์ชันที่ใช้สร้างรูปทรงมาตรฐาน หลังจากนั้นเมื่อต้องการวาดรูปทรงที่สร้างไว้ ก็เรียกใช้ฟังก์ชัน DrawSubset ซึ่งฟังก์ชันนี้จะต้องถูกเรียกใช้ระหว่าง BeginScene และ EndScene เช่นเดียวกับฟังก์ชัน DrawPrimitive และ DrawIndexedPrimitive และเมื่อเลิกใช้งานรูปทรงดังกล่าวแล้ว ก็จะต้องเรียกใช้ฟังก์ชัน Release เพื่อทำการยกเลิกการจองหน่วยความจำ

บทที่ 5
เทคนิคของไดเร็กซ์สามดีที่นำมาใช้

5.1 การกำหนดสี (Color)

ในไดเร็กซ์สามดี จะใช้รูปแบบสีแบบ RGB โดยใช้แม่สี 3 สี คือ แดง เขียว และน้ำเงิน มารวมกันด้วยความเข้มของแม่สีที่แตกต่างกันจนได้มาซึ่งสีที่ต้องการ สำหรับโครงสร้างของสี RGB ในไดเร็กซ์สามดีจะมีอยู่ด้วยกัน 2 แบบ

แบบแรก คือ D3DCOLOR ซึ่งจะมีขนาด 32 บิต โดยแม่สีแต่ละสีจะแทนด้วยข้อมูลขนาด 8 บิต ดังรูปที่ 5-1



รูปที่ 5-1 แสดงรูปแบบสีแบบ ARGB ขนาด 32 บิต

โดยที่บางรูปแบบจะมีค่าความทึบแสงและโปร่งแสงด้วย เรียกค่านี้ว่า อัลฟา (Alpha) ในการกำหนดสีนั้น จะสามารถกำหนดค่าของแม่สีแต่ละสีได้ตั้งแต่ 0 ถึง 255 ดังตัวอย่างในตารางที่ 5-1

```
D3DCOLOR brightRed = D3DCOLOR_ARGB(255, 255, 0, 0);
D3DCOLOR someColor = D3DCOLOR_ARGB(255, 144, 87, 201);
```

ตารางที่ 5-1 แสดงตัวอย่างการกำหนดสีแบบ ARGB

นอกจากนี้ ถ้าไม่ต้องการใช้ค่าอัลฟา สามารถเลือกใช้มาโคร (macro) D3DCOLOR_XRGB แทน D3DCOLOR_ARGB ได้ ดังตารางที่ 5-2

```
#define D3DCOLOR_XRGB(r,g,b) D3DCOLOR_ARGB(0xff,r,g,b)
```

ตารางที่ 5-2 แสดงการกำหนดมาโคร D3DCOLOR_XRGB

แบบที่ 2 ของการเก็บสี คือ ใช้โครงสร้างของ D3DCOLORVALUE โดยโครงสร้างนี้จะใช้การกำหนดค่าให้กับแม่สีโดยใช้ชนิดข้อมูล float ซึ่งค่าที่กำหนดนี้จะอยู่ในช่วงระหว่าง 0.0 ถึง 1.0 โดยรูปแบบของโครงสร้างแบบนี้ แสดงได้โค้ดในตารางที่ 5-3

```
typedef struct _D3DCOLORVALUE {
float r;
float g;
float b;
float a;
} D3DCOLORVALUE;
```

ตารางที่ 5-3 แสดงโครงสร้างสีแบบ D3DCOLORVALUE

นอกจากนี้ อาจใช้โครงสร้างของ D3DXCOLOR แทน D3DCOLORVALUE ได้ โดยที่ D3DXCOLOR จะมีตัวแปรเหมือนกับ D3DCOLORVALUE แต่มีฟังก์ชันต่าง ๆ เพิ่มขึ้นมาเพื่อช่วยให้การใช้สีง่ายขึ้น

สีของเวอร์เท็กซ์

สีของรูปพื้นฐาน (Primitive) ต่าง ๆ เช่น รูปสามเหลี่ยมจะถูกกำหนดโดยสีของเวอร์เท็กซ์ ดังนั้นในการสร้างสีให้กับวัตถุสามมิติใด ๆ จะต้องกำหนดสีให้กับเวอร์เท็กซ์ของวัตถุนั้น ๆ เสียก่อน ดังตารางที่ 5-4

```
struct ColorVertex
{
float _x, _y, _z;
D3DCOLOR _color;
static const DWORD FVF;
}
const DWORD ColorVertex::FVF = D3DFVF_XYZ | D3DFVF_DIFFUSE;
```

ตารางที่ 5-4 แสดงการกำหนดสตรัคท์ของเวอร์เท็กซ์ที่มีการใช้สีด้วย

5.2 การทำเชดดิ้ง (Shading)

เชดดิ้งจะอยู่ในขั้นตอนของการทำ Rasterization ในไปป์ไลน์ โดยจะเป็นการนำสีของเวอร์เท็กซ์มาคำนวณหาสีของรูปพื้นฐานเพื่อนำไปหาสีของพิกเซลที่จะแสดงบนหน้าจอไดเร็กซ์สามดีจะมีการทำ Shading อยู่ 2 แบบ คือ Flat Shading และ Gouraud Shading

Flat Shading จะใช้สีของเวอร์เท็กซ์แรกของรูปสามเหลี่ยมในการคำนวณหาสีของสามเหลี่ยมนั้นโดยไม่สนใจสีของเวอร์เท็กซ์อีก 2 เวอร์เท็กซ์ที่เหลือ ดังตารางที่ 5-5

```
ColorVertex t[3];
t[0]._color = D3DCOLOR_XRGB(255, 0, 0);
t[1]._color = D3DCOLOR_XRGB(0, 255, 0);
t[2]._color = D3DCOLOR_XRGB(0, 0, 255);
```

ตารางที่ 5-5 แสดงรูปสามเหลี่ยมที่มีการกำหนดสีของเวอร์เท็กซ์แตกต่างกัน

จากตัวอย่างในตารางที่ 5-5 จะได้สีของสามเหลี่ยมเป็นสีของเวอร์เท็กซ์ t[0] ซึ่งก็คือ สีแดง จากคุณสมบัติดังกล่าว ทำให้Flat Shading มีข้อเสีย คือ สีของวัตถุจะออกมาไม่เรียบ ซึ่งจะทำให้ภาพที่ได้ออกมาไม่สวยงาม สำหรับ Gouraud Shading นั้นจะใช้การนำค่าสีของแต่ละเวอร์เท็กซ์มาทำการเฉลี่ยกันด้วยวิธีการ Linear Interpolation ซึ่งจะทำให้สีที่ได้ออกมาจากการทำ Gouraud Shading เรียบกว่าสีที่ได้จากการทำ Flat Shading สำหรับการเลือกว่าจะใช้ Shading แบบใดนั้น จะต้องกำหนดค่าโดยใช้ฟังก์ชัน SetRenderState ดังนี้

```
Device->SetRenderState(D3DRS_SHADEMODE, D3DSHADE_FLAT);
Device->SetRenderState(D3DRS_SHADEMODE, D3DSHADE_GOURAUD);
```

ตารางที่ 5-6 แสดงการเลือกใช้ Shading โดยใช้ฟังก์ชัน SetRenderState

โดยโค้ดบรรทัดบนในตารางที่ 5-6 จะเป็นการกำหนดให้ใช้ Flat Shading และโค้ดบรรทัดล่างจะเป็นการกำหนดให้ใช้ Gouraud Shading

5.3 การให้แสง (Lighting)

ในการทำให้ภาพที่ได้ออกมามีความสมจริงเพิ่มมากขึ้นนั้น จะต้องมีการกำหนดแหล่งกำเนิดแสงเพื่อให้แสงกับวัตถุ โดยการคำนวณสีของวัตถุในกรณีที่มีแสงมาเกี่ยวข้องนั้น จะต้องคำนวณจาก 3 ปัจจัยหลัก ได้แก่

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

ส่วนประกอบของแสง ในไดเร็กซ์สามดีนั้น จะกำหนดแสงที่ออกมาจากแหล่งกำเนิดแสงด้วยส่วนประกอบ 3 อย่าง คือ

- Ambient Light เป็นส่วนประกอบของแสงที่ให้ความสว่างกับฉากทั้งหมดอย่างเท่าเทียมกัน โดยไม่ขึ้นกับระยะห่างจากแหล่งกำเนิดแสงและไม่ขึ้นกับทิศทางการหันหน้าของวัตถุที่ทำกับแหล่งกำเนิดแสง

- Diffuse Light เป็นส่วนประกอบของแสงที่เมื่อตกกระทบวัตถุแล้วจะสะท้อนออกไปทุกทิศทุกทาง ดังนั้นแสงแบบนี้จะขึ้นกับทิศทางของแสงและลักษณะของวัตถุเท่านั้น ไม่ขึ้นกับตำแหน่งของกล้อง

- Specular Light เป็นส่วนประกอบของแสงที่เมื่อตกกระทบวัตถุแล้วจะสะท้อนออกไปในทิศทางใดทิศทางหนึ่งเท่านั้น โดยแสงแบบนี้จะทำให้เห็นแสงสว่างได้ในบางมุม ดังนั้นจะต้องนำทั้งตำแหน่งของกล้อง ทิศทางของแสง และลักษณะของวัตถุมาคำนวณ

เนื่องจากแสง Specular เป็นส่วนประกอบของแสงที่ต้องใช้การคำนวณมาก ดังนั้น สามารถกำหนดให้ใช้หรือไม่ใช้ก็ได้ โดยใช้ฟังก์ชัน SetRenderState ดังตารางที่ 5-7

```
Device->SetRenderState(D3DRS_SPECULARENABLE, true);
```

ตารางที่ 5-7 แสดงการกำหนดว่าจะคำนวณ Specular Light หรือไม่โดยใช้ฟังก์ชัน SetRenderState

ในการกำหนดค่าสีให้กับแสงแต่ละแบบ จะใช้โครงสร้างของ D3DCOLORVALUE หรือ D3DXCOLOR ก็ได้ ดังตารางที่ 5-8

```
D3DXCOLOR redAmbient(1.0f, 0.0f, 0.0f, 1.0f);
D3DXCOLOR blueDiffuse(0.0f, 0.0f, 1.0f, 1.0f);
D3DXCOLOR whiteSpecular(1.0f, 1.0f, 1.0f, 1.0f);
```

ตารางที่ 5-8 แสดงการกำหนดสีของแสงโดยใช้โครงสร้างแบบ D3DXCOLOR

5.4 ลักษณะของวัตถุ (Material)

สีของวัตถุที่เห็นในโลกของความเป็นจริงจะถูกกำหนดด้วยสีของแสงที่สะท้อนออกมาจากวัตถุนั้น สำหรับในไดเร็กซ์สามดีนั้น จะคล้าย ๆ กัน แต่จะกำหนดลักษณะของวัตถุ (material) ขึ้นมา ซึ่งลักษณะของวัตถุจะเป็นตัวกำหนดว่าแสงชนิดใดสีใดจะถูกสะท้อนออกมาจากพื้นผิววัตถุได้มากน้อยเพียงใด โดยจะกำหนดลักษณะของวัตถุด้วยโครงสร้างของ D3DMATERIAL9 ดังตารางที่ 5-9

```
typedef struct _D3DMATERIAL9 {
D3DCOLORVALUE Diffuse, Ambient, Specular, Emissive;
float Power;
} D3DMATERIAL9;
```

ตารางที่ 5-9 แสดงโครงสร้างของ D3DMATERIAL9

โดยตัวอย่างของการกำหนดลักษณะของวัตถุให้สะท้อนแต่แสงสีแดงทุกชนิด และดูดซับแสงสีอื่นทั้งหมด แสดงได้ดังโค้ดในตารางที่ 5-10

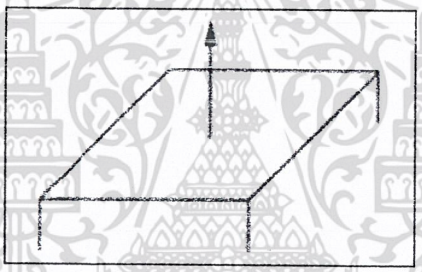
```
D3DMATERIAL9 red;
::ZeroMemory(&red, sizeof(red));
red.Diffuse = D3DXCOLOR(1.0f, 0.0f, 0.0f, 1.0f); // red
red.Ambient = D3DXCOLOR(1.0f, 0.0f, 0.0f, 1.0f); // red
red.Specular = D3DXCOLOR(1.0f, 0.0f, 0.0f, 1.0f); // red
red.Emissive = D3DXCOLOR(0.0f, 0.0f, 0.0f, 1.0f); // no emission
red.Power = 5.0f;
```

ตารางที่ 5-10 แสดงตัวอย่างการกำหนดลักษณะของวัตถุ

ซึ่งถ้ากำหนดลักษณะของวัตถุดังโค้ดด้านบนให้กับวัตถุใด ๆ วัตถุนั้นก็จะกลายเป็นสีแดง โดยจะเป็นสีแดงที่มีทั้งส่วนประกอบของแสงที่เป็น Ambient, Diffuse และ Specular

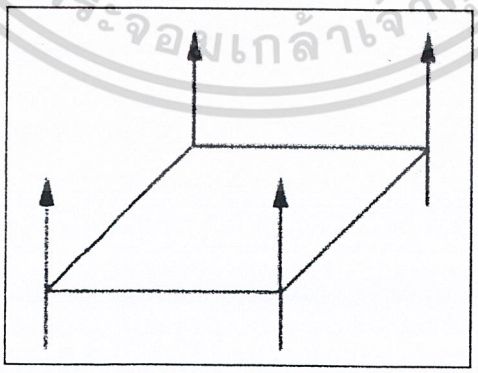
5.5 เส้นปกติของเวอร์เท็กซ์ (Vertex Normal)

เส้นปกติของพื้นผิว (Face Normal) จะเป็นตัวบอกทิศทางที่พื้นผิวหันหน้าอยู่ ดังรูปที่ 5-2



รูปที่ 5-2 แสดงเส้นปกติของพื้นผิว (Face Normal)

สำหรับเส้นปกติของเวอร์เท็กซ์ (vertex normal) นั้นจะกำหนดเส้นปกติที่แต่ละเวอร์เท็กซ์แทนการกำหนดเส้นปกติของพื้นผิว ดังรูปที่ 5-3



รูปที่ 5-3 แสดงเส้นปกติของเวอร์เท็กซ์ (Vertex Normal)

เนื่องจากไดเร็กซ์สามดีจะคำนวณมุมที่แสงตกกระทบพื้นผิวที่ทุก ๆ เวอร์เท็กซ์ของรูป ดังนั้นจึงต้องทราบเส้นปกติของเวอร์เท็กซ์เพื่อนำมาใช้ในการคำนวณ ซึ่งตัวอย่างของโครงสร้างของเวอร์เท็กซ์ที่มีการนำเส้นปกติของเวอร์เท็กซ์มาใช้ จะแสดงได้ดังโค้ดในตารางที่ 5-11

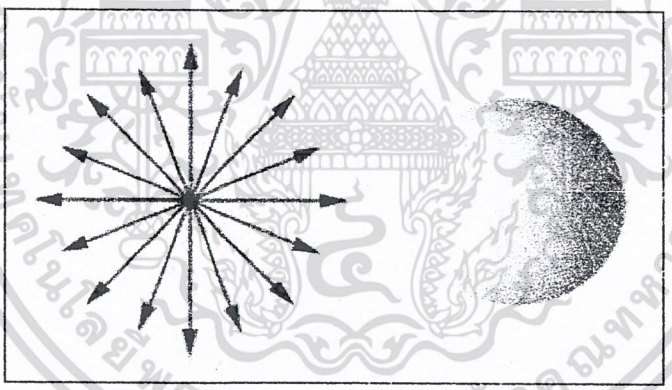
```
struct Vertex
{
float _x, _y, _z;
float _nx, _ny, _nz;
static const DWORD FVF;
}
const DWORD Vertex::FVF = D3DFVF_XYZ | D3DFVF_NORMAL;
```

ตารางที่ 5-11 แสดงโครงสร้างของเวอร์เท็กซ์ที่มีการนำเส้นปกติของเวอร์เท็กซ์มาใช้

5.6 ชนิดของแหล่งกำเนิดแสง

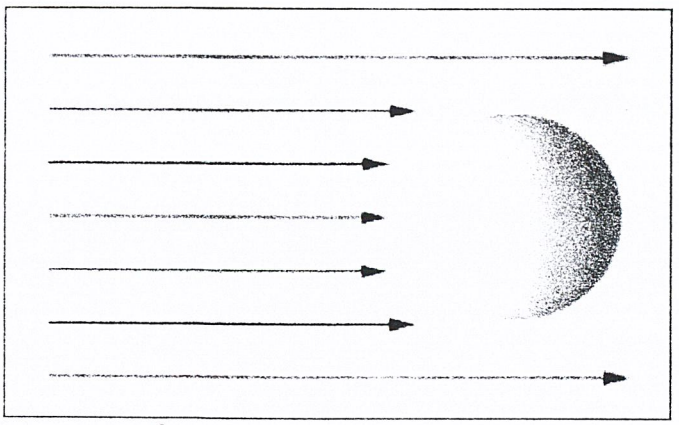
ไดเร็กซ์สามดีมีชนิดของแหล่งกำเนิดแสงอยู่ 3 ชนิด ได้แก่

- แหล่งกำเนิดแสงแบบจุด (Point light) เป็นแหล่งกำเนิดแสงที่ต้องระบุตำแหน่งแหล่งกำเนิดแสงแบบจุดนี้จะส่องแสงออกมาทุกทิศทุกทาง ดังรูปที่ 5-4



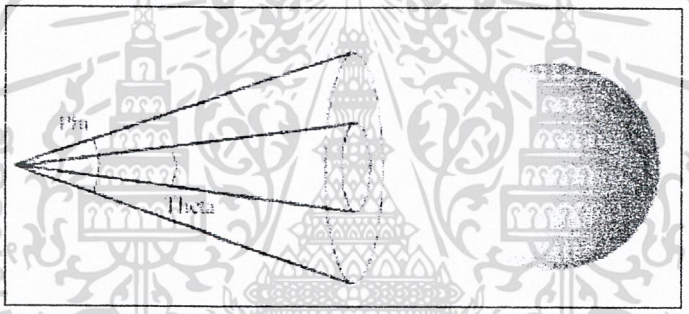
รูปที่ 5-4 แสดงแหล่งกำเนิดแสงแบบจุด

- แหล่งกำเนิดแสงแบบขนาน (Directional light) เป็นแหล่งกำเนิดแสงที่ไม่มีตำแหน่งและส่องแสงเป็นลำแสงขนานในทิศทางใดทิศทางหนึ่งตามที่กำหนด ดังรูปที่ 5-5



รูปที่ 5-5 แสดงแหล่งกำเนิดแสงแบบขนาน

- แหล่งกำเนิดแสงแบบสปอตไลท์ เป็นแหล่งกำเนิดแสงที่มีตำแหน่งและส่องแสงออกมาเป็นรูปกรวยในทิศทางที่กำหนด โดยกรวยแสงจะถูกระบุด้วยค่า 2 ค่า ได้แก่ มุม θ (theta) และมุม φ (phi) โดยมุม θ จะใช้เป็นมุมของกรวยใน และมุม φ จะเป็นมุมของกรวยนอก ดังรูปที่ 5-6



รูปที่ 5-6 แสดงแหล่งกำเนิดแสงแบบสปอตไลท์

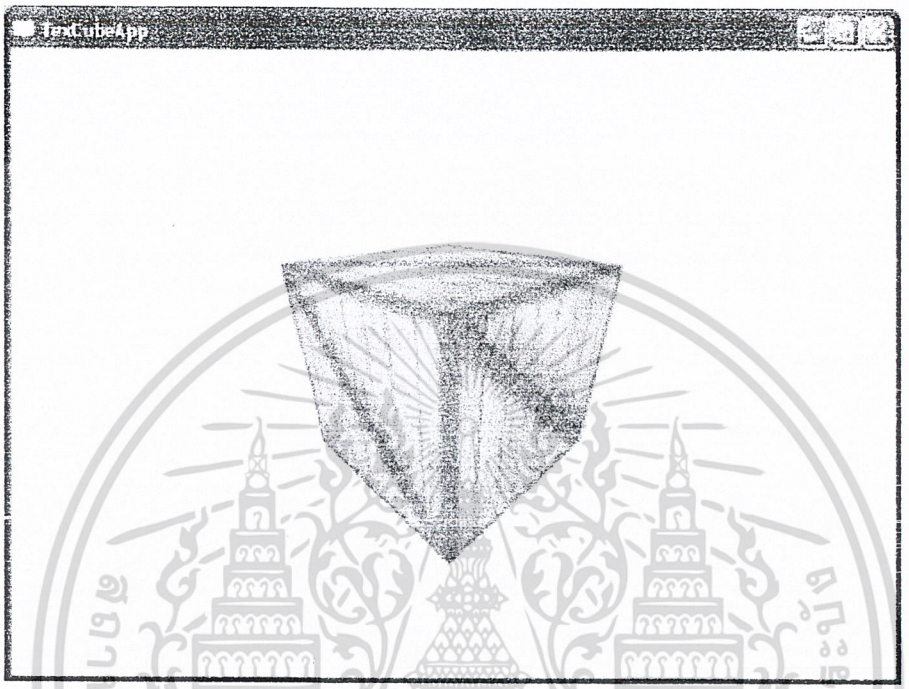
ซึ่งในไดเร็กซ์สามดีแหล่งกำเนิดแสงทั้ง 3 แบบ จะถูกแสดงได้ด้วยโครงสร้าง D3DLIGHT9 ดังตารางที่ 5-12

```
typedef struct _D3DLIGHT9 {
D3DLIGHTTYPE Type;
D3DCOLORVALUE Diffuse;
D3DCOLORVALUE Specular;
D3DCOLORVALUE Ambient;
D3DVECTOR Position;
D3DVECTOR Direction;
float Range;
float Falloff;
float Attenuation0;
float Attenuation1;
float Attenuation2;
float Theta;
float Phi;
} D3DLIGHT9;
```

ตารางที่ 5-12 แสดงโครงสร้างของ D3DLIGHT9

5.7 การแปะภาพเข้ากับวัตถุ (Texturing)

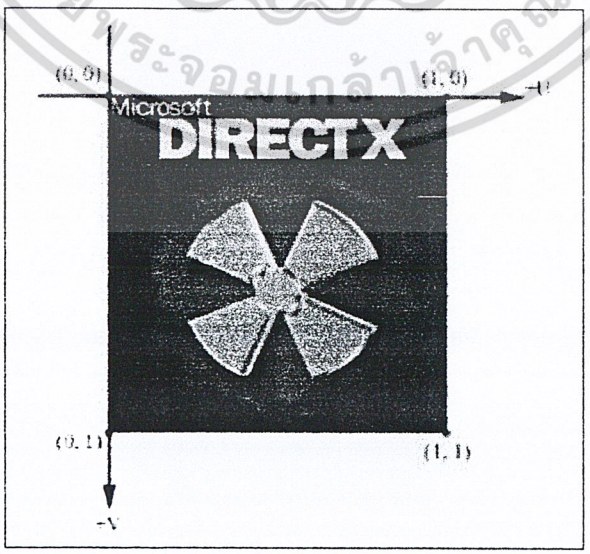
Texturing คือ เทคนิคการนำเอาภาพมาแปะติดเข้ากับวัตถุ 3 มิติ ทำให้วัตถุมีสีสันที่สวยงามและดูสมจริงมากขึ้น รูปที่ 5-7 เป็นการนำเทคนิคนี้มาใช้สร้างลังที่มีความสมจริงมากขึ้น โดยแปะภาพของลังไว้ที่แต่ละด้านของลูกบาศก์ที่สร้างขึ้นมา



รูปที่ 5-7 แสดงตัวอย่างวัตถุหลังการทำ Texturing

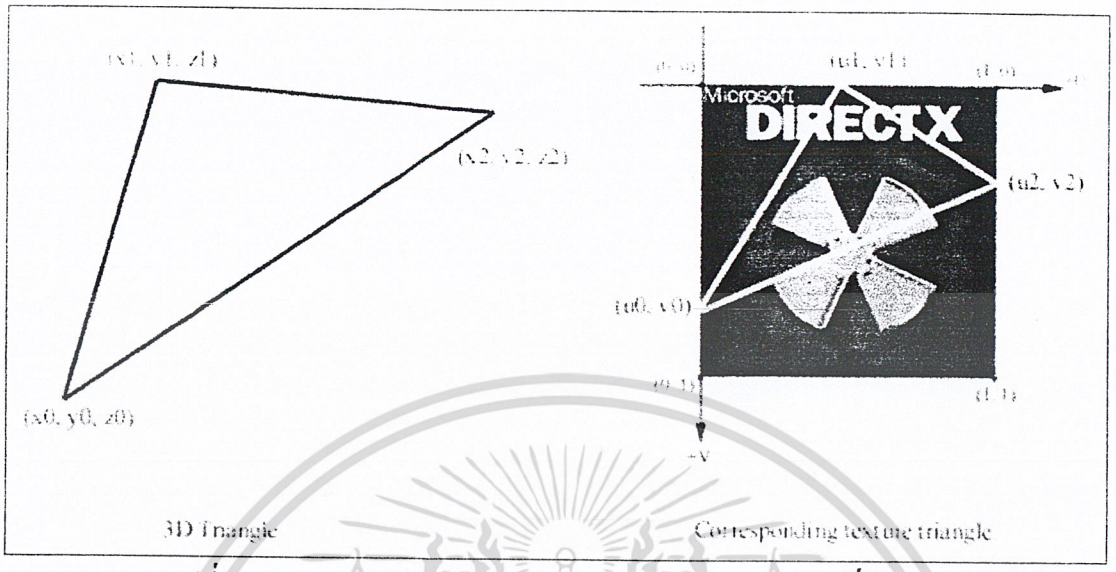
5.7.1 โคออร์ดิเนตของการทำ Texturing

ไดเร็กซ์สามดีจะใช้ระบบแกนโคออร์ดิเนตที่ประกอบด้วยแกน u ในแนวนอน และแกน v ในแนวตั้ง โดยทั้งสองแกนจะมีค่าตั้งแต่ 0 ถึง 1 ดังรูปที่ 5-8



รูปที่ 5-8 แสดงโคออร์ดิเนตของการทำ Texturing

ในการกำหนดรูปที่จะแปะเข้ากับสามเหลี่ยมแต่ละรูปนั้น จะต้องกำหนดตำแหน่งของแกน u และแกน v ของรูปภาพที่ใช้ให้กับเวอร์เท็กซ์แต่ละเวอร์เท็กซ์ของรูปสามเหลี่ยมด้วย ดังรูปที่ 5-9



รูปที่ 5-9 แสดงวิธีการกำหนดค่า u และค่า v ให้กับเวอร์เท็กซ์ตามที่ต้องการ

ดังนั้นในการทำ texturing นั้นจะต้องกำหนดรูปแบบของเวอร์เท็กซ์ใหม่ โดยจะต้องเพิ่มในส่วนของ D3DFVF_TEX1 เพื่อใช้ในการกำหนดค่าของแกน u และแกน v ของรูปภาพให้กับเวอร์เท็กซ์ ตามโค้ดในตารางที่ 5-13

```
struct Vertex
{
float _x, _y, _z;
float _nx, _ny, _nz;
float _u, _v;
static const DWORD FVF;
};
const DWORD Vertex::FVF = D3DFVF_XYZ | D3DFVF_NORMAL |
D3DFVF_TEX1;
```

ตารางที่ 5-13 แสดงโครงสร้างของเวอร์เท็กซ์ที่มีการทำ Texturing ด้วย

5.7.2 การสร้างและการนำรูปภาพ (Texture) มาใช้

ข้อมูลของรูปภาพที่ใช้ในแอพพลิเคชันนั้นปกติจะอ่านมาจากไฟล์รูปภาพที่เก็บอยู่ภายในฮาร์ดดิสก์แล้วนำมาเก็บไว้ในวัตถุที่ชื่อ IDirect3DTexture9 ซึ่งไดเร็กซ์สามดีมีฟังก์ชันที่ใช้ในการทำงานดังกล่าวไว้แล้ว ดังตารางที่ 5-14

```
HRESULT D3DXCreateTextureFromFile(
LPDIRECT3DDEVICE9 pDevice,
LPCSTR pSrcFile,
LPDIRECT3DTEXTURE9* ppTexture
);
```

ตารางที่ 5-14 แสดงฟังก์ชัน D3DXCreateTextureFromFile

ซึ่งสามารถใช้ฟังก์ชันนี้โหลดไฟล์ภาพที่มีนามสกุล BMP, DDS, DIB, JPG, PNG และ TGA ได้ สำหรับตัวอย่างของการสร้างรูปภาพที่จะนำมาแปะจากไฟล์ภาพ wall.bmp มาเก็บไว้ในตัวแปร IDirect3Dtexture9 ที่มีพอยน์เตอร์ชื่อว่า _stonewall ระบุ แสดงได้ตามโค้ดในตารางที่ 5-15

```
IDirect3Dtexture9* _stonewall;
D3DXCreateTextureFromFile(_device, "stonewall.bmp", &_stonewall);
```

ตารางที่ 5-15 แสดงตัวอย่างการสร้างรูปภาพที่จะนำมาแปะ (Texture) จากไฟล์ภาพ

หลังจากนำรูปภาพจากไฟล์ภาพมาเก็บในหน่วยความจำแล้ว ก็มาถึงขั้นตอนการเลือกใช้รูปภาพจากรูปภาพต่าง ๆ ที่ได้นำมาเก็บไว้ ซึ่งการกำหนดว่าจะเรียกใช้รูปภาพใดนั้น จะใช้ฟังก์ชัน SetTexture ที่มีรูปแบบดังตารางที่ 5-16

```
HRESULT IDirect3DDevice9::SetTexture(
DWORD Stage,
IDirect3DBaseTexture9* pTexture);
```

ตารางที่ 5-16 แสดงฟังก์ชัน SetTexture

สำหรับตัวอย่างการเรียกใช้ฟังก์ชัน ถูกแสดงได้ตามโค้ดในตารางที่ 5-17

```
Device->SetTexture(0, _stonewall);
```

ตารางที่ 5-17 แสดงการใช้ฟังก์ชัน SetTexture

ในกรณีที่ไม่ต้องการให้มีการเรียกใช้รูปภาพใด ๆ จะต้องเขียนโค้ดดังตารางที่ 5-18

```
Device->SetTexture(0, 0);
renderObjectWithoutTexture();
```

ตารางที่ 5-18 แสดงการวาดภาพโดยไม่มีการใช้รูปภาพ (Texture) ใด ๆ

และถ้าต้องการสร้างรูปสามเหลี่ยมหลาย ๆ รูป โดยที่แต่ละรูปมีการใช้รูปภาพที่แตกต่างกัน จะต้องเขียนโค้ดดังตารางที่ 5-19

```
Device->SetTexture(0, _tex0);
drawTrisUsingTex0();
Device->SetTexture(0, _tex1);
drawTrisUsingTex1();
```

ตารางที่ 5-19 แสดงการเปลี่ยนรูปภาพ (Texture) ที่ใช้

5.7.3 การปรับภาพ (filtering)

ในการทำ texturing นั้น อาจจะเกิดปัญหาขึ้นได้ ตัวอย่างเช่น ในกรณีที่รูปภาพที่นำมาแปะติดมีขนาดเล็กกว่าวัตถุ ซึ่งจะต้องทำการขยายรูปภาพให้มีขนาดพอดีกับวัตถุก่อนแล้วจึงจะสามารถทำการ texturing ได้ และในกรณีที่รูปภาพที่นำมาแปะติดมีขนาดใหญ่กว่าวัตถุ ก็จะต้องทำการย่อรูปภาพนั้นให้มีขนาดพอดีกับวัตถุก่อนเช่นเดียวกัน ซึ่งการย่อและขยายรูปภาพนั้นจะทำให้ภาพที่ได้ออกมามีความผิดเพี้ยนไปจากเดิม ทำให้เกิดความไม่สวยงามและไม่สมจริงขึ้น ดังนั้น ไดเร็กซ์สามดีจึงมีวิธีการปรับภาพ (filtering) เพื่อให้ภาพที่ได้ออกมาหลังจากการทำ texturing มีความสวยงามขึ้น ซึ่งวิธีการ filtering จะมีอยู่ 3 แบบ คือ

- Nearest point sampling เป็นดีฟอลต์ของการทำ filtering วิธีการนี้เป็นวิธีการทำ filtering ที่แย่ที่สุดใน 3 วิธี แต่ใช้เวลาในการคำนวณน้อยที่สุด โดยตัวอย่างของการกำหนดให้มีการทำ filtering แบบนี้ แสดงได้ตามโค้ดในตารางที่ 5-20

```
Device->SetSamplerState(0, D3DSAMP_MAGFILTER, D3DTEXF_POINT);
Device->SetSamplerState(0, D3DSAMP_MINFILTER, D3DTEXF_POINT);
```

ตารางที่ 5-20 แสดงตัวอย่างการกำหนดให้ทำ filtering แบบ Nearest point sampling

- Linear filtering เป็นวิธีการทำ filtering ที่ได้ผลค่อนข้างดี และสามารถคำนวณได้อย่างรวดเร็วในปัจจุบัน ที่สุด โดยตัวอย่างของการกำหนดให้มีการทำ filtering แบบนี้ แสดงได้ตามโค้ดในตารางที่ 5-21

```
Device->SetSamplerState(0, D3DSAMP_MAGFILTER, D3DTEXF_LINEAR);
Device->SetSamplerState(0, D3DSAMP_MINFILTER, D3DTEXF_LINEAR);
```

ตารางที่ 5-21 แสดงตัวอย่างการกำหนดให้ทำ filtering แบบ Linear filtering

- Anisotropic filtering เป็นวิธีการทำ filtering ที่ดีที่สุด แต่ใช้เวลาในการคำนวณนานที่สุดเช่นเดียวกัน โดยตัวอย่างของการกำหนดให้มีการทำ filtering แบบนี้ แสดงได้ตามโค้ดในตารางที่ 5-22

```
Device->SetSamplerState(0, D3DSAMP_MAGFILTER, D3DTEXF_
ANISOTROPIC);
Device->SetSamplerState(0, D3DSAMP_MINFILTER, D3DTEXF_
ANISOTROPIC);
```

ตารางที่ 5-22 แสดงตัวอย่างการกำหนดให้ทำ filtering แบบ Anisotropic filtering

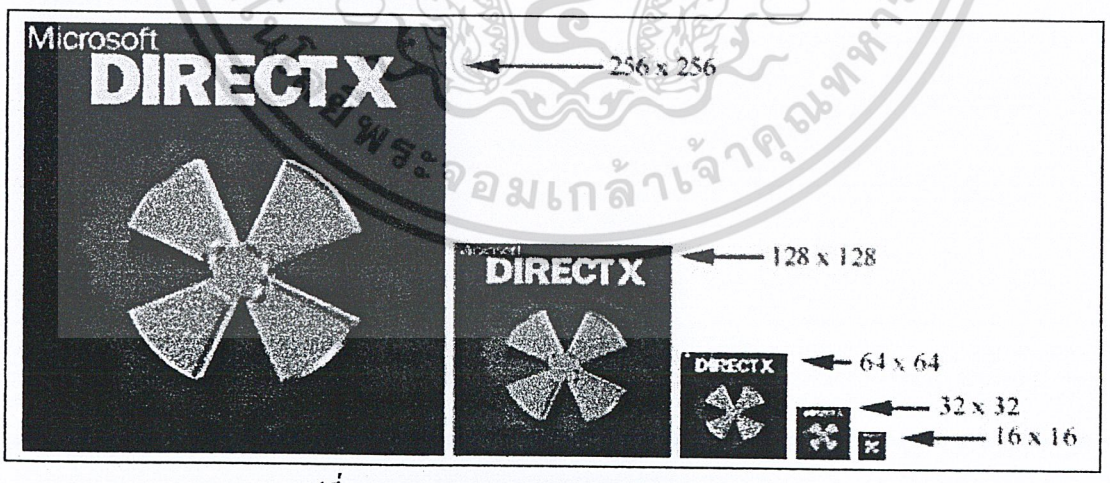
นอกจากนี้ ถ้าใช้การทำ filtering แบบ Anisotropic filtering แล้ว จะต้องมีการกำหนดค่า D3DSAMP_MAXANISOTROPY ด้วย ซึ่งค่าตัวเลขที่กำหนดให้กับค่านี้ยิ่งมากก็จะยิ่งได้ภาพที่ดี โดยตัวอย่างโค้ดด้านล่างจะเป็นการกำหนดค่า D3DSAMP_MAXANISOTROPY เป็น 4

```
Device->SetSamplerState(0, D3DSAMP_MAXANISOTROPY, 4);
```

ตารางที่ 5-23 แสดงตัวอย่างการกำหนดค่า D3DSAMP_MAXANISOTROPY

5.7.4 การทำชุดของรูปภาพ (Mipmaps)

ในการทำ texturing นั้น สามารถสร้างชุดของรูปภาพไว้ก่อนได้ ซึ่งภายในชุดนั้น จะมีรูปภาพที่มีขนาดแตกต่างกัน โดยรูปภาพที่มีขนาดเล็กกว่าจะคำนวณได้มาจากรูปภาพที่มีขนาดใหญ่กว่า และจะเรียกรูปภาพแต่ละขนาดที่อยู่ภายในชุดนั้นว่า mipmap โดยรูปที่ 5-10 จะแสดงการตัวอย่างของการทำชุดของ mipmap



รูปที่ 5-10 แสดงการทำชุดของรูปภาพ (mipmaps)

สำหรับวิธีการใช้ mipmap นั้นจะต้องกำหนดค่า mipmap filter ด้วย ซึ่งสามารถทำได้ดังตารางที่ 5-24

```
Device->SetSamplerState(0, D3DSAMP_MIPFILTER, Filter);
```

ตารางที่ 5-24 แสดงการกำหนดค่า mipmap filter โดยใช้ฟังก์ชัน SetSamplerState

ซึ่งค่า Filter ในโค้ดด้านบนนั้น จะมีค่าได้ 3 ค่า คือ

- D3DTEXF_NONE คือ ไม่มีการใช้ mipmap
- D3DTEXF_POINT คือ ใช้ mipmap ที่มีขนาดใกล้เคียงกับวัตถุมากที่สุด
- D3DTEXF_LINEAR คือ ใช้ mipmap 2 รูปที่มีขนาดใกล้เคียงกับวัตถุมาทำการเฉลี่ยเพื่อหาค่าขนาดของรูปภาพที่จะนำไปใช้จริง

5.7.5 โหมดการทำ texturing (Address Mode)

โหมดการทำ texturing จะถูกใช้ในกรณีที่มีการกำหนดค่าของแกน u และแกน v ให้กับเวอร์เท็กซ์ของวัตถุมากกว่า 1 หรือน้อยกว่า 0 โดยโหมดการทำ texturing จะมีอยู่ด้วยกัน 4 แบบ ได้แก่

1) โหมด Wrap เป็นโหมดที่จะได้ภาพเดิมซ้ำ ๆ กันถ้ามีการกำหนดค่าแกน u และแกน v เกินช่วง [0,1] โดยรูปที่ 5-11 จะแสดงภาพที่ได้จากการทำงานในโหมดนี้ โดยกำหนดค่าแกน u และแกน v ของเวอร์เท็กซ์ของสี่เหลี่ยมเป็น (0,0) (0,3) (3,0) และ (3,3)

รูปที่ 5-11 แสดงการทำ texturing ในโหมด Wrap

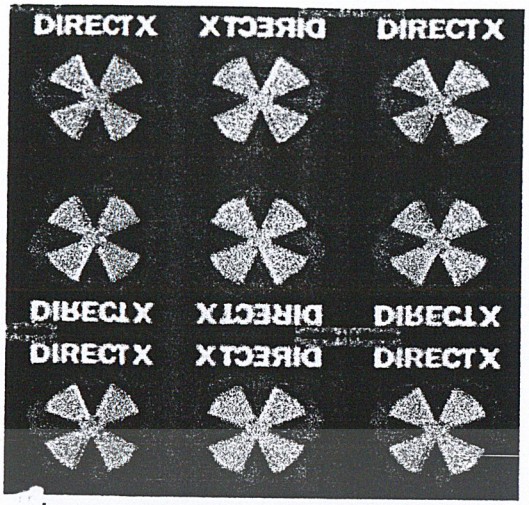
2) โหมด Border Color เป็นโหมดที่จะได้ภาพในส่วนที่เกินจากค่าแกน u และแกน v ในช่วง [0,1] เป็นดังรูปที่ 5-12

รูปที่ 5-12 แสดงการทำ Texturing ในโหมด Border Color

3) โหมด Clamp เป็นโหมดที่จะได้ภาพในส่วนที่เกินจากค่าแกน u และแกน v ในช่วง [0,1] เป็นดังรูปที่ 5-13

รูปที่ 5-13 แสดงการทำ Texturing ในโหมด Clamp

4) โหมด Mirror เป็นโหมดที่จะได้ภาพในส่วนที่เกินจากค่าแกน u และแกน v ในช่วง [0,1] เป็นภาพกลับจากรูปภาพเดิมในลักษณะคล้าย ๆ ภาพที่เกิดจากการมองกระจก ดังรูปที่ 5-14



รูปที่ 5-14 แสดงการทำ Texturing ในโหมด Mirror

5.8 การนำไฟล์นามสกุล .x มาใช้ในแอพพลิเคชัน

ไฟล์ที่มีนามสกุล .x เป็นไฟล์ที่เกิดมาจากการสร้างรูป 3 มิติโดยใช้โปรแกรมสร้างวัตถุต่าง ๆ เช่น ทรีดีเอส แม็กซ์ (3dsMax), มายา (Maya), ไลท์เวฟทรีดี (LightWave3D) ฯลฯ แล้วแปลงรูปแบบการจัดเก็บข้อมูลมาเป็นรูปแบบไฟล์นามสกุล .x

สำหรับการนำไฟล์ .x มาใช้ในแอพพลิเคชันนั้น จะใช้ฟังก์ชัน D3DXLoadMeshFromX โดยฟังก์ชันนี้จะสร้างวัตถุของ ID3DXMesh มาเป็นตัวเก็บข้อมูลของวัตถุที่นำมาจากไฟล์ .x นั้น ๆ โดยโค้ดด้านล่างจะแสดงรูปแบบของฟังก์ชันดังตารางที่ 5-25

```
HRESULT D3DXLoadMeshFromX(
LPCSTR pFilename,
DWORD Options,
LPDIRECT3DDEVICE9 pDevice,
LPD3DXBUFFER *ppAdjacency,
LPD3DXBUFFER *ppMaterials,
LPD3DXBUFFER* ppEffectInstances,
PDWORD pNumMaterials,
LPD3DXMESH *ppMesh
);
```

ตารางที่ 5-25 แสดงฟังก์ชัน D3DXLoadMeshFromX

ข้อมูลที่ 1 ที่รับเข้าไปในฟังก์ชันนี้ จะเป็นชื่อไฟล์ .x ที่นำมาใช้

สำหรับข้อมูลที่ 7 จะเป็นค่าจำนวนลักษณะ (material) ของวัตถุ (mesh) ที่อยู่ในไฟล์ .x นั้น และข้อมูลที่ 5 จะเป็นค่าอาร์เรย์ของ D3DXMATERIAL ซึ่งเป็นที่เก็บข้อมูลของลักษณะของวัตถุแต่ละอันของวัตถุที่ใช้ โดย D3DXMATERIAL จะมีโครงสร้างดังตารางที่ 5-26

```
typedef struct D3DXMATERIAL {
D3DMATERIAL9 MatD3D;
LPSTR pTextureFilename;
} D3DXMATERIAL;
```

ตารางที่ 5-26 แสดงโครงสร้างของสตรัคท์ D3DXMATERIAL

สำหรับการแปลงไฟล์ให้ได้ไฟล์ .x

- ขั้นแรกจากภาพที่เราสร้างขึ้นมาใน 3dsMax นั้น จะได้ไฟล์นามสกุล .max ออกมาซึ่งจะยังไม่สามารถไปแปลงไปเป็นไฟล์ .x ได้
- ขั้นตอนต่อไป จะต้องทำการ export ไฟล์ .max เพื่อจะเปลี่ยนให้เป็น .3DS เพื่อเตรียมที่จะนำไปเปลี่ยนให้เป็น .x

รูปที่ 5-15 แสดงการใช้โปรแกรม conv3ds.exe ในการเปลี่ยนจากไฟล์ .3DS เป็นไฟล์ .x

- สุดท้ายจะใช้โปรแกรม conv3ds.exe มาช่วยในการเปลี่ยน ไฟล์ให้เป็น .x ตามรูปด้านบน จากนั้นจึงนำไฟล์ .x ที่ได้ไปใช้ในโปรแกรมสามมิติตามต้องการ

บทที่ 6
ทฤษฎีทางด้านฮาร์ดแวร์

6.1 ความรู้เบื้องต้นเกี่ยวกับไมโครคอนโทรลเลอร์

6.1.1 ความหมายของไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์ (Microcontroller) เป็นชื่อของอุปกรณ์อิเล็กทรอนิกส์แบบหนึ่งที่รวมเอาหน่วยประมวลผล หน่วยคำนวณทางคณิตศาสตร์และลอจิก วงจรรับสัญญาณอินพุท วงจรขับสัญญาณเอาท์พุท หน่วยความจำ วงจรกำเนิดสัญญาณนาฬิกาไว้ด้วยกัน ทำให้สามารถนำไปใช้งานแทนวงจรอิเล็กทรอนิกส์ที่ซับซ้อนได้เป็นอย่างดี ช่วยลดจำนวนอุปกรณ์และขนาดของระบบ ในขณะที่มีขีดความสามารถสูงขึ้น ภายใต้งบประมาณที่เหมาะสม

ไมโครคอนโทรลเลอร์มาจากคำ 2 คำรวมกันคือ ไมโคร (micro) ซึ่งหมายถึงไมโครโปรเซสเซอร์ (microprocessor) ซึ่งเป็นอุปกรณ์ประมวลผลข้อมูลขนาดเล็ก ภายในประกอบด้วย หน่วยประมวลผลกลางหรือซีพียู (CPU:central processing unit) หน่วยคำนวณทางคณิตศาสตร์และลอจิก (ALU:arithmetic logic unit) วงจรเชื่อมต่อหน่วยความจำ และวงจรสัญญาณนาฬิกา อีกคำหนึ่งคือคำว่า คอนโทรลเลอร์ (controller) หมายถึง อุปกรณ์ควบคุม ดังนั้น ไมโครคอนโทรลเลอร์จึงเป็นอุปกรณ์ที่ใช้ในการควบคุม โดยที่สามารถเขียนโปรแกรมเพื่อกำหนดรูปแบบการควบคุมได้อย่างอิสระ

ความแตกต่างระหว่างไมโครคอนโทรลเลอร์กับไมโครโปรเซสเซอร์ แสดงได้ดังตารางที่ 6-1

คุณสมบัติ	ไมโครโปรเซสเซอร์	ไมโครคอนโทรลเลอร์
ขนาดของหน่วยประมวลกลาง	ไม่น้อยกว่า 8 บิต	ส่วนใหญ่มีขนาดมากกว่า 8 บิต
หน่วยคำนวณทางคณิตศาสตร์และลอจิก	มีอยู่ภายใน	มีอยู่ภายใน
วงจรกำเนิดสัญญาณนาฬิกา	มีอยู่ภายใน	มีอยู่ภายใน
การเชื่อมต่อกับหน่วยความจำโปรแกรม	เชื่อมต่อภายนอกเท่านั้น	ใช้ได้ทั้งภายในและภายนอก
การเชื่อมต่อกับหน่วยความจำข้อมูล	เชื่อมต่อภายนอกเท่านั้น	ใช้ได้ทั้งภายในและภายนอก
การเชื่อมต่อกับพอร์ตอินพุทและเอาท์พุท	เชื่อมต่อภายนอกเท่านั้น	มีอยู่ภายในและสามารถขยายได้
ไทเมอร์/เคาร์เตอร์	ไม่มีในซีพียูขนาดเล็ก	มีอย่างน้อย 1 ตัว ขนาด 8-16บิต
วอตช์ด็อกไทเมอร์	ไม่มีในซีพียูขนาดเล็ก	มีอย่างน้อย 1 ตัว
จำนวนขาต่อใช้งาน	ไม่น้อยกว่า 40 ขา	มีตั้งแต่ 8 ขาขึ้นไป

ตารางที่ 6-1 แสดงความแตกต่างระหว่างไมโครคอนโทรลเลอร์กับไมโครโปรเซสเซอร์

6.1.2 การเลือกใช้ไมโครโปรเซสเซอร์หรือไมโครคอนโทรลเลอร์

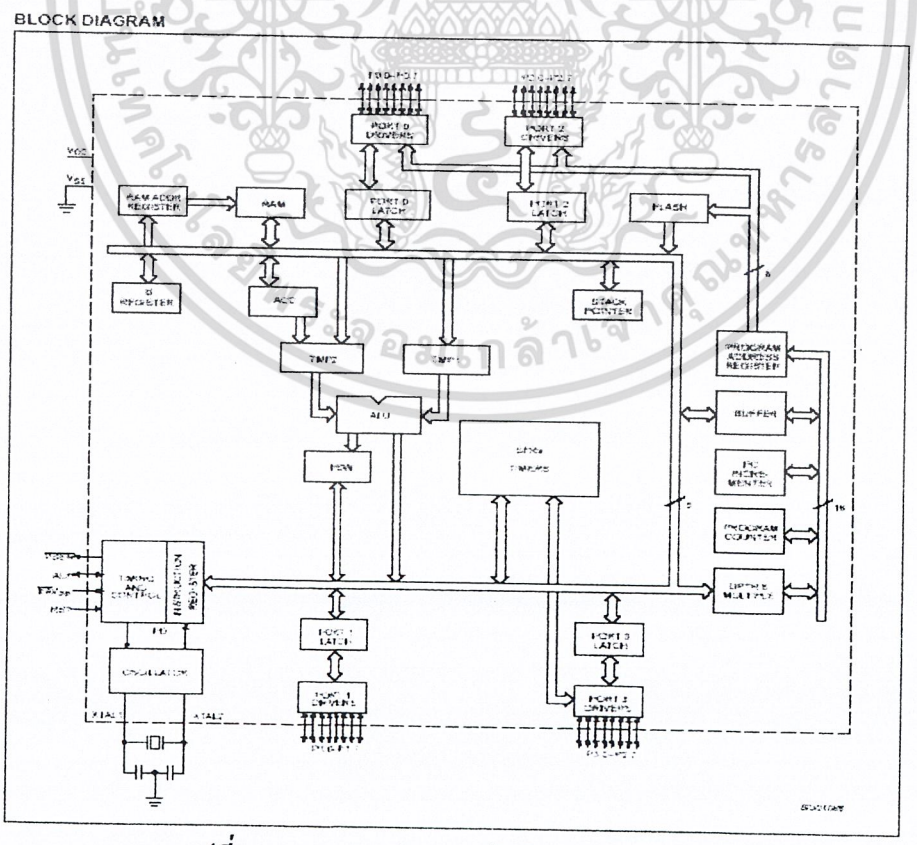
ถ้าเป็นการสร้างระบบควบคุมขนาด 8 บิต มีความต้องการเชื่อมต่อกับอุปกรณ์ภายนอกไม่มากนัก (น้อยกว่า 10 แบบ) ควรเลือกใช้ไมโครคอนโทรลเลอร์ และถ้าหากต้องมีการประมวลผลข้อมูลจำนวนมาก ต้องใช้หน่วยความจำโปรแกรมสูงถึง 8 กิโลไบต์ ทั้งยังมีความต้องการเก็บรักษาข้อมูลลงในหน่วยความจำมากเป็นกิโลไบต์ควรออกแบบให้ไมโครคอนโทรลเลอร์ในระบบควบคุมนี้เชื่อมต่อกับหน่วยความจำภายนอก

ถ้าหากต้องใช้งานกับข้อมูลมากกว่า 8 บิตตลอดเวลา และต้องการความเร็วในการทำงานสูง ๆ สามารถติดต่อหน่วยความจำได้เป็นจำนวนมาก ๆ ติดต่อกันอุปกรณ์อินพุทเอาท์พุทได้จำนวนมากมายในเวลาเดียวกัน ควรเลือกใช้ไมโครโปรเซสเซอร์ ดังจะเห็นได้จากไมโครคอมพิวเตอร์ที่ใช้ไมโครโปรเซสเซอร์เป็นหัวใจหลักในการทำงาน ทำให้สามารถเชื่อมต่อกับหน่วยความจำได้มากเป็นหน่วยกิกะไบต์ ขนาดของข้อมูลสูงสุดถึง 64 บิต ความเร็วสูงเป็นหลายกิกะเฮิรตซ์ เป็นต้น

ดังนั้นจึงพอสรุปได้ว่า ไมโครคอนโทรลเลอร์เหมาะสำหรับการสร้างระบบควบคุม ในขณะที่ไมโครโปรเซสเซอร์เหมาะสำหรับการสร้างระบบประมวลผลข้อมูลความเร็วสูงและระบบควบคุมที่มีขนาดใหญ่มาก ๆ

6.1.3 MCS-51

สำหรับ MCS-51 นั้น จะเป็นไมโครคอนโทรลเลอร์ชนิดหนึ่งที่นิยมใช้กันอย่างแพร่หลาย โดยสถาปัตยกรรมของ MCS-51 แสดงได้ตามรูปที่ 6-1



รูปที่ 6-1 แสดงสถาปัตยกรรมไมโครคอนโทรลเลอร์

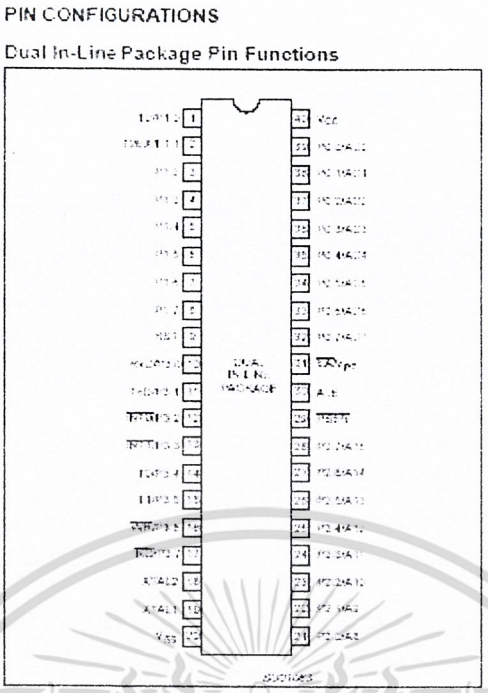
6.1.4 คุณสมบัติของไมโครคอนโทรลเลอร์ MCS-51 อนุกรม AT89Cxx

- เป็นไมโครคอนโทรลเลอร์ที่ใช้ซีพียูขนาด 8 บิต
- ภายในมีหน่วยความจำโปรแกรมเป็นแบบแฟลชสามารถลบและเขียนใหม่ได้มากกว่า 1000 ครั้ง
- หน่วยความจำข้อมูลพื้นฐานเป็นหน่วยความจำแบบแรม
- ขาพอร์ตเป็นแบบ 2 ทิศทาง สามารถใช้งานเป็นได้ทั้งอินพุทและเอาท์พุท
- มีวงจรสื่อสารอนุกรมแบบฟูลดูเพล็กซ์
- ไทเมอร์/เคาน์เตอร์ขนาด 16 บิต อย่างน้อย 2 ตัว
- สามารถรองรับแหล่งกำเนิดอินเตอร์รัปต์ได้ 6 ประเภท
- สามารถขยายหน่วยความจำภายนอกได้สูงสุด 64 กิโลไบต์
- มีวงจรกำเนิดสัญญาณนาฬิกาอยู่ภายในชิป

ไมโครคอนโทรลเลอร์ MCS-51 อนุกรม AT89Cxx แบบต่าง ๆ นั้น จะมีคุณสมบัติส่วนใหญ่ที่คล้ายกัน แต่จะมีบางคุณสมบัติที่แตกต่างกันไปตามรุ่นของ MCS-51 แต่ละรุ่น ดังตารางที่ 6-2

เบอร์ของไมโครคอนทรอเลอร์	หน่วยความจำโปรแกรม	หน่วยความจำข้อมูล	จำนวนไทเมอร์/เคาน์เตอร์ 16 บิต
AT89C1051	แบบแฟลช ขนาด 1 กิโลไบต์	แรม 64 ไบต์	1
AT89C2051	แบบแฟลช ขนาด 2 กิโลไบต์	แรม 128 ไบต์	2
AT89C51	แบบแฟลช ขนาด 4 กิโลไบต์	แรม 128ไบต์	2
AT89C52	แบบแฟลช ขนาด 8 กิโลไบต์	แรม 256 ไบต์	3
AT89C55	แบบแฟลช ขนาด 20 กิโลไบต์	แรม 256 ไบต์	3

ตารางที่ 6-2 แสดงคุณสมบัติของ MCS-51 อนุกรม AT89Cxx แบบต่าง ๆ



รูปที่ 6-2 แสดงรายละเอียดของขา (Pin) ของไมโครคอนโทรลเลอร์ AT89Cxx

6.1.5 การจัดขาของไมโครคอนโทรลเลอร์ AT89Cxx

สำหรับรายละเอียดของขา (Pin) ของไมโครคอนโทรลเลอร์ AT89Cxx นั้นจะแสดงได้ตามรูปที่ 6-2 ซึ่งโดยส่วนใหญ่แล้วไมโครคอนโทรลเลอร์ AT89Cxx จะมีจำนวนขาเท่ากับ 40 ขา โดยมีรายละเอียดของขาต่าง ๆ ดังนี้

ขา Vcc ใช้สำหรับต่อไฟเลี้ยง +5V

ขา GND เป็นขากราวนด์ สำหรับต่อกับกราวนด์ของระบบ

ขาพอร์ต0 (P0.0-P0.7) มี 8 ขา สามารถใช้เป็นอินพุทเอาท์พุททั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ต 0 ขาใดขาหนึ่งเป็นอินพุทสามารถทำได้โดยการเขียนข้อมูล 1 ไปยังแต่ละบิตของพอร์ตที่ต้องการจะติดต่อด้วย นอกจากนั้นขาพอร์ตนี้ยังถูกใช้งานในการติดต่อกับขาแอดเดรสไบต์ต่ำของหน่วยความจำภายนอก (A0-A7) และขาข้อมูล (D0-D7) โดยใช้กระบวนการมัลติเพล็กซ์ (Multiplex) เข้าช่วย เพื่อสลับการทำงานเป็นได้ทั้งขาติดต่อแอดเดรสและขาข้อมูล

ขาพอร์ต1 (P1.0-P1.7) มี 8 ขา สามารถใช้เป็นอินพุทเอาท์พุททั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ตใดเป็นอินพุทสามารถทำได้โดยการเขียนข้อมูล 1 ไปยังแต่ละบิตของพอร์ตที่ต้องการจะติดต่อด้วย

ขาพอร์ต2 (P2.0-P2.7) มี 8 ขา สามารถใช้เป็นอินพุทเอาท์พุททั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ต ใดเป็นอินพุทสามารถทำได้โดยการเขียนข้อมูล 1 ไปยังแต่ละบิตของพอร์ตที่ต้องการจะติดต่อด้วย นอกจากนั้นขาพอร์ตนี้ยังถูกใช้งานในการติดต่อกับขาแอดเดรสไบต์สูงของหน่วยความจำภายนอก (A8-A15)

ขาพอร์ต3 (P3.0-P3.7) มี 8 ขา สามารถใช้เป็นอินพุทเอาท์พุททั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ตใดเป็นอินพุทสามารถทำได้โดยการเขียนข้อมูล 1 ไปยังแต่ละบิตของพอร์ตที่ต้องการจะติดต่อด้วย นอกจากนี้พอร์ต 3 ยังเป็นขาที่มีหน้าที่การใช้งานพิเศษ มีรายละเอียดดังนี้

- P3.0 ใช้เป็นขารับข้อมูลจากการสื่อสารแบบอนุกรม หรือขา RxD
- P3.1 ใช้เป็นขาส่งข้อมูลจากการสื่อสารแบบอนุกรม หรือขา TxD
- P3.2 ใช้เป็นขารับสัญญาณอินเตอร์รัปต์จากภายนอกช่อง 0 หรือขา INT0
- P3.3 ใช้เป็นขารับสัญญาณอินเตอร์รัปต์จากภายนอกช่อง 1 หรือขา INT1
- P3.4 ใช้เป็นขารับสัญญาณไทเมอร์จากภายนอกช่อง 0 หรือขา T0
- P3.5 ใช้เป็นขารับสัญญาณอินเตอร์รัปต์จากภายนอกช่อง 1 หรือขา T1
- P3.6 ใช้เป็นขาสัญญาณ WR ในกรณีที่ใช้เชื่อมต่อกับหน่วยความจำภายนอก
- P3.7 ใช้เป็นขาสัญญาณ RD ในกรณีที่ใช้เชื่อมต่อกับหน่วยความจำภายนอก

ขารีเซต (Reset) ใช้ในการรีเซตการทำงานของไมโครคอนโทรลเลอร์ โดยในการป้อนสัญญาณเพื่อรีเซตสถานะที่ขานี้ต้องอยู่ในระดับรีเซตอย่างน้อย 2 แมชชีนไซเคิล โดยที่วงจรกำเนิดสัญญาณนาฬิกายังคงทำงานต่อเนื่องไปอย่างเป็นปกติ

ขา ALE/PROG (Address Latch Enable/Program pulse input) เป็นขาที่ใช้ในการควบคุมการแลตช์ของขาพอร์ต 0 เมื่อมีการใช้งานหน่วยความจำภายนอก

ขา PSEN (Program Store Enable) ขานี้ใช้ในการส่งสัญญาณเพื่อร้องขอติดต่อกับหน่วยความจำโปรแกรมภายนอก เมื่อไมโครคอนโทรลเลอร์ต้องการอ่านข้อมูลจากหน่วยความจำโปรแกรมภายนอกตัวไมโครคอนโทรลเลอร์ต้องการอ่านข้อมูลจากหน่วยความจำโปรแกรมภายนอก ตัวไมโครคอนโทรลเลอร์จะส่งสัญญาณออกมาที่ขานี้ 2 ครั้งในแต่ละแมชชีนไซเคิล แต่ถ้าหากติดต่อกับหน่วยความจำข้อมูลภายนอก ขานี้จะไม่มีการส่งสัญญาณใด ๆ ออกมา

ขา EA/Vpp (External Access enable/Programming voltage input) ใช้สำหรับเลือกการติดต่อหน่วยความจำโปรแกรมจากภายนอกหรือภายในตัวไมโครคอนโทรลเลอร์ ถ้าหากขานี้เป็น 0 จะเป็นการเลือกให้ไมโครคอนโทรลเลอร์ติดต่อกับหน่วยความจำโปรแกรมภายนอก แต่ถ้าหากขานี้เป็น 1 จะเป็นการเลือกให้ไมโครคอนโทรลเลอร์ติดต่อกับหน่วยความจำภายในตัวไมโครคอนโทรลเลอร์ นอกจากนี้ที่ขานี้ยังใช้เป็นขาอินพุทสำหรับรับแรงดันไฟสูงสำหรับการโปรแกรมหน่ยความจำภายใน
ไมโครคอนโทรลเลอร์ สำหรับไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชต้องการแรงดันสำหรับการโปรแกรม +12V

ขา XTALI /XTAL2 เป็นขาสำหรับต่อคริสตอล (XTAL) เพื่อสร้างสัญญาณนาฬิกาในการกำหนดจังหวะการทำงานของไมโครคอนโทรลเลอร์

6.1.6 โครงสร้างและการทำงานพอร์ต

ไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชมีพอร์ตให้ใช้งานทั้งสิ้น 4 พอร์ตคือ พอร์ต 0 ถึงพอร์ต 3 แต่ละพอร์ตมีขนาด 8 บิต เป็นพอร์ตแบบ 2 ทิศทาง กล่าวคือ สามารถเป็นได้ทั้งอินพุทสำหรับรับสัญญาณข้อมูลเข้าและเอาท์พุทสำหรับส่งสัญญาณข้อมูลออก ทุกพอร์ตของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชมีวงแลตช์และวงจรขับตลอดจนบัฟเฟอร์อินพุท

ที่พอร์ต 0 และพอร์ต 2 จะใช้งานเป็นพอร์ตอินพุทและเอาท์พุทสำหรับงานทั่วไป และใช้ในการติดต่อกับหน่วยความจำภายนอก สำหรับพอร์ต 3 ทั้งพอร์ตและพอร์ต 1 บางขานอกจากจะใช้เป็นขาพอร์ตอินพุทเอาท์พุทตามปกติแล้ว ยังสามารถใช้งานในหน้าที่พิเศษได้อีก

วงจรภายในของแต่ละพอร์ตของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช พอร์ต 0 วงจรแลตช์ของแต่ละบิตในแต่ละพอร์ตก็คือวงจรดีฟลิปฟลอปนั่นเอง การอ่านค่าสถานะของพอร์ตและสถานะของวงจรแลตช์สามารถกระทำได้อย่างอิสระด้วยสัญญาณที่แยกจากกัน นั่นคือ สัญญาณอ่านข้อมูลจากขาพอร์ต และสัญญาณอ่านข้อมูลจากวงจรแลตช์ ส่วนการเขียนข้อมูลมายังพอร์ตต้องส่งสัญญาณมายังขา CLK ของดีฟลิปฟลอปในขณะที่ข้อมูลจะผ่านมาทางขาบัสข้อมูลภายในเข้าสู่ขา D ของดีฟลิปฟลอป

ที่ผ่านนี้มีวงจรมัลติเพล็กซ์สำหรับกำหนดลักษณะการทำงานของพอร์ตว่า ต้องการใช้งานเป็นขาพอร์ตอินพุทเอาท์พุทปกติหรือใช้ในการติดต่อกับหน่วยความจำภายนอกไมโครคอนโทรลเลอร์

เนื่องจากที่ขาพอร์ต 0 ไม่มีวงจรพูลอัพภายใน หากมีการนำพอร์ต 0 ไปใช้งานเป็นพอร์ตอินพุทจะต้องต่อตัวต้านทางพูลอัพภายนอกเข้าที่ขาพอร์ต 0 ทุกขาด้วย

วงจรของพอร์ต 1 ซึ่งมีลักษณะโดยทั่วไปคล้ายกับพอร์ต 0 หากแต่ไม่มีวงจรมัลติเพล็กซ์ เนื่องจากพอร์ตนี้จะไม่ใช้ในการติดต่อกับหน่วยความจำนอก แต่จะมีวงจรพูลอัพภายในที่แต่ละบิตของพอร์ตนี้แทน

วงจรภายในของพอร์ต 2จะคล้ายกับพอร์ต 0 มาก ต่างกันเพียงมีวงจรพูลอัพเพิ่มเติมเข้ามา พอร์ต 3 จะเห็นได้ว่าคล้ายกับพอร์ต 3 จะเห็นได้ว่าคล้ายกับพอร์ต 1 มีการเพิ่มเติมวงจรบัฟเฟอร์และวงจรอินพุทเอาท์พุทเพื่อทำงานในฟังก์ชั่นพิเศษเข้ามา เนื่องจากพอร์ต 3 สามารถนำไปใช้งานในหน้าที่พิเศษได้ทุก

6.1.7 การนำไมโครคอนโทรลเลอร์ MCS-51 ไปใช้งานในโครงงาน

การใช้งานเป็นพอร์ตอินพุท

เนื่องจากพอร์ตทั้งหมดของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชสามารถเป็นได้ทั้งอินพุทและเอาท์พุท ดังนั้นจึงมีความจำเป็นอย่างยิ่งต้องทำความเข้าใจถึงการกำหนดลักษณะการทำงานให้แก่พอร์ตของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

ในการกำหนดให้เป็นพอร์ตอินพุท ต้องเริ่มต้นด้วยการเขียนข้อมูล 1 มาที่แต่ละบิตของพอร์ตที่ต้องการใช้งานเป็นอินพุท เพื่อหยุดการทำงานของแฟตที่ใช้ในการขับสัญญาณเอาท์พุทของบิตนั้น ๆ ทำให้ขาสัญญาณของพอร์ตเชื่อมต่อเข้ากับวงจรพูลอัพภายในโดยตรง ส่งผลให้ขาพอร์ตนั้นมีลอจิกเป็น 1 สามารถรับสัญญาณลอจิก 0 จากอุปกรณ์ภายนอกได้ง่าย สัญญาณข้อมูลจากอุปกรณ์ภายนอกจะถูกส่งเข้ามาแล้วเก็บไว้ในวงจรบัฟเฟอร์ภายในพอร์ต แล้วรอให้ซีพียูมาอ่านค่าเข้าไป เมื่อเป็นเช่นนี้อุปกรณ์

ภายนอกที่เชื่อมต่อกับพอร์ตอินพุทของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชควรกำหนดให้ทำงานในสภาวะลอจิก 0 จะดีและสะดวกที่สุด (ซึ่งในปัจจุบันอุปกรณ์อินพุทที่เชื่อมต่อไมโครคอนโทรลเลอร์แทบทั้งหมดทำงานที่ลอจิก 0 แล้ว

การใช้งานเป็นพอร์ตเอาท์พุท

โดยปกติแล้ว ขาพอร์ตจะกำนดให้มีลักษณะเป็นเอาท์พุทอยู่แล้ว ดังนั้นจึงสามารถส่งข้อมูลออกไปได้อย่างง่ายดายและตรงไปตรงมา กล่าวคือ เมื่อต้องการส่งข้อมูล 0 ออกไปทางเอาท์พุทก็ให้เขียนข้อมูล 0 ไปยังวงจรแลตช์ ซึ่งก็จะส่งต่อไปขับเฟต ทำให้เฟตทำงาน ที่ขาพอร์ตที่กำหนดให้ทำงานก็จะเกิดลอจิก 0 ขึ้น ในทางตรงข้ามหากต้องการส่งข้อมูล 1 ออกไป ก็ให้เขียนข้อมูล 1 ไปยังวงจรแลตช์ วงจรขับก็จะหยุดทำงาน ทำให้ที่ขาพอร์ตเชื่อมต่อกับวงจรพูลอัพภายในการเคลื่อนย้ายข้อมูล โดยถ้าเป็นอินพุทจะมีสัญญาณอ่านข้อมูลที่บัฟเฟอร์ แต่ถ้าเป็นเอาท์พุทจะไม่มีการอ่านข้อมูลที่บัฟเฟอร์แต่อย่างใด เว้นแต่ในกรณีที่ต้องการตรวจสอบข้อมูลที่ส่งออกมาทางเอาท์พุท

เมื่อใช้งานเป็นพอร์ตเอาท์พุท แต่ละขา (หรือแต่ละบิต) ของแต่ละพอร์ตมีความสามารถในการจ่ายกระแสหรือที่เรียกว่า กระแสซอร์ส (Source current) ได้สูงสุด 10mA และทุกขารวมกันในแต่ละพอร์ต (ทั้ง 8 บิต) สูงสุด 26 mAสำหรับพอร์ต 0 และ 15 mA สำหรับพอร์ต 1-3 ในกรณีที่ใช้งานทุกพอร์ตเอาท์พุทเพื่อไม่ให้เกิดปัญหาเกี่ยวกับความสามารถในการจ่ายกระแสจึงควรต่อวงจรบัฟเฟอร์ทางเอาท์พุทเพื่อช่วยในการขับกระแสอีกทางหนึ่ง

การอ่านค่าลอจิกจากพอร์ต

ในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชสามารถอ่านค่าลอจิกจากพอร์ตได้ 2 ลักษณะคือ อ่านจากขาพอร์ตโดยตรง และอ่านจากวงจรแลตช์ของแต่ละพอร์ต

ในกรณีที่พอร์ตต่อกับขาเบสทรานซิสเตอร์ชนิด NPN และขาอิมิตเตอร์ของทรานซิสเตอร์ตัวนั้นต่อลงกราวนด์ หากมีการส่งข้อมูล 1 ไปยังทรานซิสเตอร์ จะทำให้ทรานซิสเตอร์ทำงานสถานะลอจิกที่ขาพอร์ตจะเป็น 0 เนื่องจากเมื่อทรานซิสเตอร์ทำงาน จะเสมือนว่าขาพอร์ตนั้นถูกต่อลงกราวนด์ ทำให้หากอ่านค่าลอจิกที่ขาพอร์ตจะได้ผลตรงข้ามกับที่ส่งออกมา แต่ถ้าหากทำงานอ่านค่าลอจิกที่วงจรแลตช์ จะได้ค่าที่ตรงกับค่าที่ต้องการส่งจริง ดังนั้น ในการอ่านค่าลอจิกจากพอร์ตจึงต้องเลือกวิธีการให้เหมาะสมกับอุปกรณ์ที่นำมาต่อด้วย

จังหวะการทำงานของไมโครคอนโทรลเลอร์ MCS-51

ในการใช้งานไมโครคอนโทรลเลอร์ MCS-51 จะต้องทำความเข้าใจถึงจังหวะก่ารทำงานของซีพียูและลำดับขั้นตอนการประมวลผลคำสั่ง ในการประมวลผลคำสั่งของซีพียูจะมีขั้นตอนหลัก ๆ 2 ขั้นตอน คือ กระบวนการเฟตช์ (fetch) เป็นการเรียกคำสั่งออกจากหน่วยความจำโปรแกรมแล้วทำการแปลงรหัสคำสั่งนั้นเป็นภาษาเครื่องเพื่อเตรียมการประมวลผล จั้นตอนต่อมาคือ กระบวนการเอ็กซิคิวต์ (execute) เป็นการกระทำตามคำสั่งที่กำหนดหรือตามที่เฟตช์ขึ้นม โดยกระบวนการการก่อนหน้านี้ เมื่อทำการเอ็กซิคิวต์คำสั่งเรียบร้อยแล้ว ก็จะไปเริ่มกระบวนการเฟตช์คำสั่งใหม่ต่อไป

เมื่อเริ่มจ่ายไฟให้แก่ไมโครคอนโทรลเลอร์ จะเกิดการรีเซตในลักษณะที่เรียกว่า เพาเวอร์ออนรีเซต (power–on reset) ซีพียูเริ่มต้นการทำงานที่แอดเดรส 0000H ของหน่วยความจำโปรแกรม จังหวะการ

ทำงานของซีพียูจะเป็นไปตามรูปแบบ โดยได้รับการกำหนดมาจากกรอบการทำงานหรือแมชชีนไซเคิล (machine cycle) ของไมโครคอนโทรลเลอร์ MCS-51 1 รอบการทำงานหรือแมชชีนไซเคิลจะแบ่งย่อยออกเป็น 6 สเตท (state) กำหนดชื่อเป็น S1-S6 ในแต่ละสเตทมีค่าเวลาเท่ากับ 2 คาบเวลาของสัญญาณนาฬิกา ถ้าสัญญาณนาฬิกามีความถี่ 12 MHz จะมีคาบเวลาเท่ากับ 1 ms คาบเวลาทั้งสองภายในหนึ่งสเตทจะเรียกว่า เฟส 1 (phase 1) และเฟส 2 (phase 2)

การเอ็กซิคิวต์คำสั่งที่ใช้เวลา 1 ไซเคิล เริ่มต้นที่สเตท 1 จะเป็นการอ่านค่าออปโค้ด อันเป็นกระบวนการแลตช์ค่าของออปโค้ตส่งไปให้รีจิสเตอร์คำสั่ง (Instruction Register: IR) การเฟตช์ครั้งที่สองจะเกิดขึ้นที่สเตท 4 ภานในแมชชีนไซเคิลเดียวกัน ในกรณีที่เป็นคำสั่งไบต์เดียว การเฟตช์ครั้งที่ 2 ภายในแมชชีนไซเคิลเดียวกันจะถูกตัดทิ้งไป ในคำสั่งที่มีใช้เวลา 1 ไซเคิล จะสิ้นสุดการทำงานลงในสเตท 6 ของแมชีนไซเคิลเดียวกัน

ในกรณีที่คำสั่งใช้เวลา 2 ไซเคิล การทำงานของคำสั่งนั้นจะสิ้นสุดในสเตท 6 ของแมชีนไซเคิลที่สอง สำหรับในการกระทำคำสั่ง MOVX ซึ่งเป็นคำสั่งขนาด 1 ไบต์ 2 ไซเคิล จะไม่มีการเฟตช์เกิดขึ้นในไซเคิลที่สองของคำสั่ง MOVX นี้ เนื่องจากซีพียูจะไปทำการติดต่อกับหน่วยความจำภายนอก จะเห็นได้ว่าเวลาในการเอ็กซิคิวต์จะไม่ได้ขึ้นอยู่กับว่าทำการติดต่อหน่วยความจำโปรแกรมภายในหรือภายนอก

สรุปได้ว่าในการทำงาน 1 รอบหรือ 1 แมชชีนไซเคิล ซีพียูในไมโครคอนโทรลเลอร์ MCS-51 จะใช้เวลา 12 คาบเวลาของสัญญาณนาฬิกา นั่นคือ เวลาในการทำงาน 1 ไซเคิลมีค่าเท่ากับ 1ms หรือมีความเร็วในการทำงานภายใน 1 MHz ดังนั้นถ้าต้องการทราบความเร็วของการทำงานภายในของไมโครคอนโทรลเลอร์ MCS-51 สามารถหาได้จาก ค่าความถี่สัญญาณนาฬิกาหารด้วย 12 และถ้าต้องการหาค่าเวลาของ 1 รอบการทำงานหรือ 1 แมชชีนไซเคิล สามารถทำได้โดยการหาส่วนกลับของความเร็ว ในการทำงานภายในของไมโครคอนโทรลเลอร์ MCS-51 สามารถสรุปเป็นสูตรทางคณิตศาสตร์ได้ดังนี้

ความเร็วในการทำงานภายในของไมโครคอนโทรลเลอร์เท่ากับ

ความถี่ของสัญญาณนาฬิกา (ค่าของคริสตอบที่ต่ออยู่ที่ขา XTAL 1และ XTAL2)/12

เวลา 1 แมชชีนไซเคิล = 1/ความเร็วในการทำงานภายในของไมโครคอนโทรลเลอร์

6.2 ความรู้เบื้องตนเกี่ยวกับพอร์ตอนุกรม

6.2.1 ลักษณะทางกายภาพของพอร์ตอนุกรม

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 มีวงจรการสื่อสารแบบ Full Duplex 1 ชุด โดยใช้ขาสัญญาณของพอร์ต 3 คือ ขา P3.0 เป็นขารับข้อมูลเข้าหรือ RxD และ ขา P3.1 เป็นขอส่งข้อมูลออกหรือ TxD โดยวงจรสื่อสารอนุกรมของ ไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชเป็นแบบอะซิงโครนัส ปกติแล้วพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51 จะใช้ในการติดต่อสื่อสารกับพอร์ตอนุกรมของคอมพิวเตอร์ โดยใช้มาตรฐาน RS-232 RS-422 และ RS-485

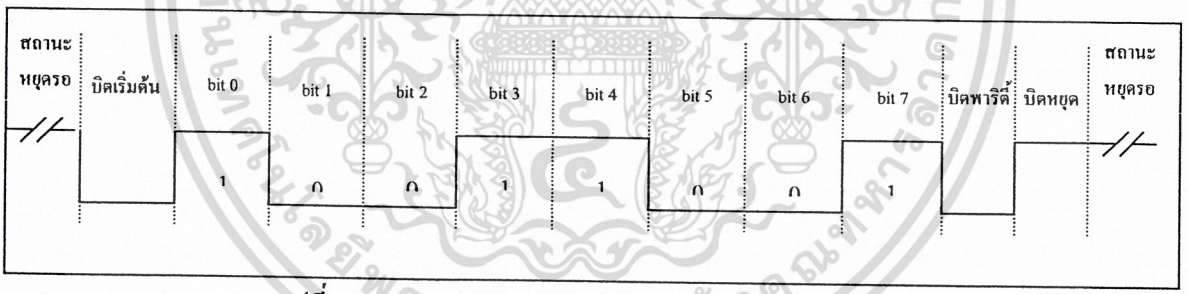
การสื่อสารข้อมูลแบบอะซิงโครนัส

การสื่อสารข้อมูลแบบอะซิงโครนัสคือการรับส่งข้อมูลโดยไม่จำเป็นต้องมีสัญญาณนาฬิการ่วมด้วยแต่จะใช้การกำหนดความเร็วในการรับส่งข้อมูลให้มีค่าเท่ากัน ซึ่งเรียกอัตราเร็วนี้ว่า อัตราบอด หรือ บอดเรต (Baud rate) มีหน่วยเป็น บิตต่อวินาที (bit per second: bps)

รูปแบบของข้อมูลที่ใช้ในการรับส่งข้อมูลแบบอะซิงโครนัสประกอบด้วย 4 ส่วนด้วยกันคือ

1. บิตเริ่มต้น (Start bit) มีขนาด 1 บิต
2. บิตข้อมูลแบบอนุกรมมีขนาด 8 บิต
3. บิตตรวจสอบพาร์ริตี้ (Parity bit) มีขนาด 1 บิตหรือไม่มี
4. บิตปิดท้ายหรือบิตหยุด (Stop bit) มีขนาด 1 บิต

รูปแบบของข้อมูลอนุกรมแบบอะซิงโครนัส เมื่อไม่มีการส่งข้อมูลขอ เริ่มต้นข้อมูลจะมีสถานะลอจิก 1 เรียกสถานะนี้ว่า สถานะหยุดรอ (waiting state) การเริ่มต้นส่งข้อมูลจะเริ่มจากการให้ขาข้อมูลมีลอจิก 0 เรียกบิตนี้ว่า บิตเริ่มต้น (start bit) จากนั้นบิตข้อมูลจะถูกส่งออกไป โดยเริ่มจากบิตที่มีความสำคัญต่ำสุดหรือ บิต LSB ก่อน ซึ่งข้อมูลที่ต้องการส่งมีจำนวน 8 บิต จากนั้นตามด้วย บิตพาร์ริตี้ (parity bit) ซึ่งใช้ในการตรวจสอบความผิดพลาดที่เกิดขึ้นจากการส่งข้อมูล บิตสุดท้ายที่จะส่งคือ บิตปิดท้าย หรือบิตหยุด (stop bit) โดยจะเป็นการทำให้ data มีสถานะลอจิกเป็น 1 อีกครั้งด้วยระยะเวลาอย่างน้อย 1บิต, 1.5 บิต หรือ 2 บิต เพื่อเป็นการแสดงว่าสิ้นสุดข้อมูลแล้ว โดยรูปแบบการส่งข้อมูลแบบอะซิงโครนัสจะแสดงได้ดังรูปที่ 6-3



รูปที่ 6-3 แสดงรูปแบบการส่งข้อมูลแบบอะซิงโครนัส

อัตราความเร็วในการรับและส่งข้อมูลของการรับส่งข้อมูลแบบอะซิงโครนัสหรืออัตราบอดหรือบอดเรตที่ใช้สำหรับอนุกรม RS-232 มีด้วยกันหลายค่าตั้งแต่ 110 ถึง 19200 บิตต่อวินาที สมมติว่า ข้อมูลอนุกรมมีขนาด 8 บิต ไม่มีการตรวจสอบพาริตี้ มีบิตเริ่มต้น 1 บิต และบิตปิดท้าย 1 บิต ความยาวของมูล 1 ไบต์ จะมีความยาวเท่ากับ 10 บิต ถ้าใช้อัตราบอด 9600 บิตต่อวินาที จะได้การส่งข้อมูลความเร็ว 960 ไบต์ต่อวินาที

รีจิสเตอร์ที่เกี่ยวข้องกับกับการทำงานของพอร์ตอนุกรมใน MCS-51 มี 2 ตัว ดังนี้

รีจิสเตอร์บัฟเฟอร์ของพอร์ตอนุกรมหรือ SBUF (Serial data buffer register)

มีแอดเดรสอยู่ที่ 99H ในพื้นที่ของรีจิสเตอร์ฟังก์ชันพิเศษหรือ SFR มีขนาด 8 บิต แบ่งเป็น 2 ส่วน คือ รีจิสเตอร์บัฟเฟอร์สำหรับส่งข้อมูล (transmit buffer register) และรับข้อมูล (receiver buffer register) เมื่อมีการเขียนข้อมูลมายังรีจิสเตอร์ SBUF ข้อมูลนั้นจะถูกส่งต่อไปยังบัฟเฟอร์สำหรับส่งขอมูล เพื่อส่งออกจากไมโครคอนโทรลเลอร์ผ่านทางขา TxD หรือ P3.1 ในกรณีที่มีการอ่านข้อมูลจากรีจิสเตอร์ SBUF ข้อมูลจะถูกส่งผ่านไปยังรีจิสเตอร์บัฟเฟอร์สำหรับรับข้อมูลเพื่อส่งต่อไปยังไมโครคอนโทรลเลอร์ต่อไป โดยการรับข้อมูลมานั้นรับจากขา RxD หรือขา P3.0 ของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลซ

รีจิสเตอร์ควบคุมการทำงานของพอร์ตอนุกรมหรือ SCON (Serial port control register)

มีแอดเดรสอยู่ที่ 98H ในพื้นที่ของรีจิสเตอร์ฟังก์ชันพิเศษหรือ SFR มีขนาด 8 บิต สามารถเข้าถึงระดับบิตมีรายระเอียดดังตารางที่ 6-3

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

ตารางที่ 6-3 แสดงรายละเอียดของรีจิสเตอร์ฟังก์ชันพิเศษ SFR ในระดับบิต

SM0-SM1 (Serial port mode bit) ใช้เลือกโหมดของพอร์ตอนุกรม

SM2 ใช้ในการเอ็นเอเบิลการสื่อสารในแบบมัลติโปรเซสเซอร์ ใช้ในการทำงานของโหมด 2 และ 3 เท่านั้น

REN (Enable serial reception) ใช้ในการเอ็นเอเบิลการรับข้อมูลของพอร์ตอนุกรมทำการเซตและเคลียร์ด้วยกระบวนการทางซอร์ฟแวร์ ถ้าตองการรับข้อมูลต้องเซตบิตนี้ให้เป็น 1

TB8 ใช้สำหรับเก็บข้อมูลบิตที่ 9 ที่ต้องการส่งออกไป ใช้ในการทำงานโหมดที่ 2 และ 3 เซตและเคลียร์ค่าด้วยกระบวนการทางซอร์ฟแวร์

RB8 ใช้สำหรับรับข้อมูลบิตที่ 9 ที่ต้องการส่งออกไป ใช้ในการทำงานโหมดที่ 2 และ 3 เซตและเคลียร์ค่าด้วยกระบวนการทางซอร์ฟแวร์

TI (Transmit interrupt flag) ใช้ในการแสดงการเกิดอินเตอร์รัปต์เมื่อมีการส่งข้อมูลออกจากพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51 สามารถเซตด้วยกระบวนการทางฮาร์แวร์ เมื่อทำการส่งข้อมูลบิตที่ 8 ไปเรียบร้อยแล้วในหาการทำงานโหมด 0 ส่วนในการทำงานโหมดอื่น บิตนี้จะเซตเมื่อมีการเริ่มต้นส่งบิตหยุดออกไป การเคลียร์บิตนีต้องใช้กระบวนการทางซอร์ฟแวร์เท่านั้น

RI (Receive interrupt flag) ใช้ในการแสดงการเกิดอินเตอร์รัปต์เมื่อมีการรับข้อมูลออกจากพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51 สามารถเซตด้วยกระบวนการทางฮาร์แวร์ เมื่อทำการรับข้อมูลบิตที่ 8 ไปเรียบร้อยแล้วในหาการทำงานโหมด 0 ส่วนในการทำงานโหมดอื่น บิตนี้จะเซตเมื่อสามารถรับบิตหยุดของขอมูลอนุกรมไปได้ครึ่งทางแล้ว ยกเว้นในกรณีที่บิต SM2 มีการเซต บิตนี้จะเซต

ได้ก็ต่อเมื่อการรับบิตหยุดหรือบิตที่9 เกิดขึนอย่างสมบูรณ์แล้ว การเคลียร์บิตนีต้องใช้กระบวนการทางซอร์ฟแวร์เท่านั้น

โหมดการทำงานของพอร์ตอนุกรมใน MCS-51

โหมด 0 กำหนดให้พอร์ตอนุกรมทำงานในลักษณะชิฟท์รีจิสเตอร์

ข้อมูลอนุกรมจะผ่านเข้าและออกทางขา RxD ส่วนขา TxD ทำหนาที่เป็นสัญญาณนาฬิกาของการเลื่อนข้อมูล (shift clock) ในโหมดนี้มีจำนวนข้อมูล 8 บิต โดยการทำรับและส่งขอมูล LSB ก่อน อัตราในการรับและส่งข้อมูลถูกกำหนดไว้คงที่ที่ 1/12 ของความถี่สัญญาณนาฬิกา

เริ่มต้นการส่งข้อมูลด้วยการเขียนข้อมูลที่ต้องการส่งมายังรีจิสเตอร์ SBUF สัญญาณเขียนข้อมูล SBUF แอกทีฟเป็น 1 ที่สเตท 6 เฟต 2 ของแมชชีนไซเคิลส่งมายังวงจรควบคุมการส่ง (TX control) ทำให้วงจรควบคุมเริ่มต้นส่งข้อมูล สัญญาณ SEND จะแอกทีฟเป็น 1 ตลอดการส่งข้อมูล

ข้อมูลจากรีจิสเตอร์ SBUF จะถูกเลื่อนออกที่ขา P3.0 หรือขา RxD ครั้งละบิต ตามจังหวะของสัญญาณนาฬิกาที่ส่งออกมาทาง P3.1 หรือ TxD โดยสัญญาณนาฬิการของการเลื่อนข้อมูลจะมีขอบบาลงของสัญญาณที่สเตท 3 เฟส และมีขอบขาขึ้นของสัญญาณที่ สเตท 6 เฟส 1 ของแต่ละแมชชีนไซเคิล ในกระบวนการส่งข้อมูลจนกระทั่งเมื่อส่งข้อมูลครบ 8 บิตแล้ว บิต TI ในรีจิสเตอร์ SCON จะเกิดการเซตเป็นการแจ้งให้ทราบว่าส่งขอมูลครบแล้ว หากการอินเตอร์รัปต์จากพอร์ตอนุกรมได้รับการเอ็นเอเบิลไว้ ก็จะเกิดการอินเตอร์รัปต์ขึ้นในระบบ เมื่อเสร็จสิ้นกระบวนการรับข้อมูล สัญญาณ SEND จะกลายเป็น 0 จนกว่าจะเริ่มต้นกระบวนการรับข้อมูลใหม่

ในกระบวนรับข้อมูล เริ่มต้นด้วยการเซต REN ให้เป็น 1 และเคลียร์บิต RI ในรีจิสเตอร์ SCON ก่อนที่สเตท 6 เฟส 2 ของแมชชีนไซเคิลถัดไป วงจรควบคุมการรับ (RX control) จะทำการเขียนข้อมูล 11111110 ไปยังชิฟท์รีจิสเตอร์สำหรับรับข้อมูลและทำการแอกทีฟสัญญาณ RECEIVE ให้เป็น 1 ในสัญญาณนาฬิกาลูกถัดไป

เมื่อสัญญาณ RECEIVE แอกทีฟ ก็จะเกิดการส่งสัญญาณนาฬิกาของการเลื่อนข้อมูล (shift clock) ขึ้นผ่านทางขา P3.1 หรือ TxD เพื่อทำการกำหนดจังหวะการรับข้อมูลที่ละบิต โดยสัญญาณนาฬิกานี้จะเกิดขึ้นในช่วง สเตท 3 เฟต 1 ถึง สเตท 6 เฟส 1 ของแต่ละแมชชีนไซเคิล การรับข้อมูลเข้ามาทางขา P3.0 หรือ RxD จะเกิดขึ้นที่สเตท 5 เฟส 2 ในแมชชีนไซเคิลเดียวกับสัญญาณนาฬิกาของการเลื่อนข้อมูล จนกระทั่งรับข้อมูลครบทั้ง 8 บิต บิต RI จะได้รับการเซ็ตเพื่อแจ้งการเสร็จสิ้นกระบวนการรับข้อมูล หากการอินเตอร์รัปต์จากพอร์ตอะนุกรมได้รับการเอ็นเอเบิลไว้ ก็จะเกิดการอินเตอร์รัปต์ขึ้นในระบบ เมื่อเสร็จสิ้นกระบวนการรับข้อมูลสัญญาณ RECEIVE จะกลายเป็น 0 จนกว่าจะเริ่มต้นกระบวนการรับต้อมูลใหม่

ในการทำงานโหมดนี้ของพอร์ตอนุกรมในไมโครคอนโทรลเลอร์ MCS-51 จะใช้ในการเชื่อมต่อกับ ไอซีรีจิสเตอร์ภายนอกเพื่อทำการขยายพอร์ตอินพุทหรือเอาท์พุท แต่ไม่เป็นที่นิยมใช้งานมากนัก เนื่องจากในไมโครคอนโทรลเลอร์ MCS-51 เองมีพอร์ตอยู่ค่อนข้างมาก และติดต่อกับพอร์ตเหล่านั้นได้ง่ายและรวดเร็วกว่ามาก

โหมด 1 กำหนดให้พอร์ตอนุกรมทำงาน UART ขนาด 8 บิตเลือกอัตราบอดได้

ใช้ในการรับส่งข้อมูลรวม 10 บิต โดยส่งข้อมูลออกทางขา P3.1 หรือ TxD และรับข้อมูลเขาทางขา P3.0 หรือ RxD ข้อมูลทั้ง 10 บิตประกอบด้วย บิตเริ่มต้น (มีค่าเป็น 0) 1 บิต ข้อมูล 8 บิต โดยรับหรือส่งข้อมูลในบิตก่อน และบิตหยุดหรือบิตปิดท้าย (มีค่าเป็น 1) ในการรับขอมูลบิตหยุดจะถูกเก็บไว้ในบิต RB8 ในรีจิสเตอร์ SCON อัตราบอดในโหมดนี้ได้รับการกำหนดโดยอัตราการเกิดโอเวอร์โฟลวของไทเมอร์ 1 ใน AT89C52

กระบวนการส่งขอมูลเริ่มต้นด้วยการแอกทีฟสัญญาณเขียนข้อมูลมายังรีจิสเตอร์ SBUF ส่งมายังวงจรควบคุมการส่ง (TX control) จากนั้นวงครควบคุมจะทำการแอกทีฟสัญญาณ SEND ที่สเตท1 เฟส 1 ของแมชชีนไซเคิลต่อมา โดยสัญญาณ SEND จะเป็น 0 ตลอดการส่งข้อมูล เมื่อสัญญาณ SEND แอกทีฟจะทำการส่งบิตเริ่มต้นก่อนเป็นบิตแรก โดยมีคาบเวลาของบิตเริ่มต้นเท่ากับ 1 แมชชีนไซเคิลจากนั้นตามด้วยการส่งบิตข้อมูล 8 บิต เรียงลำดับจากบิต LSB โดยขเอมูลที่ทำการส่งถูกเรียกออกมาจากรีจิสเตอร์บัฟเฟอร์สำหรับส่งข้อมูล ในทุกๆ บิตข้อมูลที่ถูกส่งออกไป จะเกิดสัญญาณพัลส์ SHIFT ขึ้น เพื่อให้เรียกข้อมูลในแต่ละบิตจากรีจิสเตอร์บัฟเฟอร์การกำหนดจังหวะการส่งข้อมูลใช้สัญญาณนาฬิกาการส่ง (TX control) เป็นตัวกำหนด โดยสัญญาณนาฬิกานี้มาจากการหารสัญญาณ TCLK จากไทเมอร์ 1 ด้วย 16 หลังจากการส่งบิตข้ อมูลก็จะเป็นการส่งบิตหยุดหรือบิตปิดท้าย 1 บิตดังนั้นการส่งข้อมูลจะใช้สัญญาณนาฬิกาทั้งหมด 10 ลูก เมื่อทำการส่งข้อมูลครบเรียบร้อยแล้ว จะทำการเซตบิต TI ในรีจิสเตอร์ SCON หารการอินเตอร์รัปต์จากพอร์ตอนุกรมได้รับการเอ็นเอเบิลไว้ ก็จะเกิดการอินเตอร์รัปต์ขึ้นในระบบ หลังจากที่ทำการอินเตอร์รัปต์หรือส่งข้อมูลเรียบร้อยแล้ว ต้องทำการเคลียร์บิต TI ก่อนเป็นอันดับแรก เพื่อให้การรับส่งข้อมูลทางพอร์ตอนุกรมดำเนินต่อไปได้

ด้านการรับข้อมูลจะทำการตรวจจับการเปลี่ยนแปลงระดับลอจิกจาก 1 เป็น 0 ที่ขา RxD โดยใช้อัตราการสุ่มเท่ากับ 1/16 เท่าของอัตราบอด เมื่อตรวจจับพบ ไทเมอร์/เคาน์เตอร์ที่ใช้ในการกำหนดอัตราบอดจะรีเซ็ตและทำการเขียนข้อมูล 1FFH ไปยังชิฟท์รีจิสเตอร์ ข้อมูลจะเริ่มเดินทางเข้าสู่พอร์ตอนุกรมของไมโครคอนโทรลเลอร์ผ่านทางขา RxD ในการดีความว่าบิตที่เข้ามาเป็น 0 หรือ 1 จะใช้ผลการสุ่มข้างมาก โดยบิตของข้อมูลที่เข้ามาจะถูกแบ่งเป็น 16 สเตท การสุ่มข้อมูลจะทำการสุ่มสเตทที่ 7 8 และ 9 หาก 2 ใน 3 ของการสุ่มพบว่าข้อมูลเป็นลอจิกใด จะตีความข้อมูลในบิตนั้นเป็นไปตามเสียงข้างมาก

สำหรับการรับข้อมูลจะเป็นลักษณะเดียวกับการส่งข้อมูลคือ เริ่มต้นด้วยบิตเริ่มต้นก่อน ตามด้วยบิตข้อมูล และบิตปิดท้าย ในทุกๆ การรับข้อมูลได้ 1 บิต จะมีพัลส์ SHIFT เกิดขึ้น เพื่อทำการเลื่อนข้อมูลเข้าสู้รีจิสเตอร์บัฟเฟอร์การรับข้อมูล การกำหนดจังหวะการรับข้อมูลใช้สัญญาณนาฬิกาการรับข้อมูล (RX clock) หลังจากสัญญาณนาฬิกาลูกสุดท้าย อันหมายถึงสามารถรับข้อมูลได้ครบแล้ว จงจรควบคุมการรับข้อมูลจะทำการส่งข้อมูลจากรีจิสเตอร์บัฟเฟอร์ไปยังรีจิสเตอร์ SBUF และบิต RB8 ในรีจิสเตอร์ SCON โดยข้อมูลในบิต RB8 ก็คือข้อมูลของบิตหยุดนั่นเอง พร้อมกันนั้นยังทำการเซตบิต RI ในรีจิสเตอร์ SCON ด้วย หากการอินเตอร์รัปต์จากพอร์ตอนากรมได้รับการเอ็นเอเบิลไว้ ก็จะเกิดการอินเตอร์รัปต์ขึ้นในระบบ หลังการบริการอินเตอร์รัปต์หรือรับข้อมูลเรียบร้อยแล้ว ต้องทำการเคลียร์บิต RI ก่อน เพื่อให้การรับส่งข้อมูลทางพอร์ตอนุกรมดำเนินต่อไปได้

ในการทำงานโหมดนี้ได้รับความนิยมสูงสุด เนื่องจากมีกระบวนการที่ไม่ซับซ้อนและสามารถทำการรับส่งข้อมูลกับคอมพิวเตอร์ได้อย่างมีประสิทธิภาพ

โหมด 2 และ 3 ของวงจรพอร์ตอนุกรม

ในทั้ง 2 โหมดนี้จะใช้รูปแบบข้อมูลรวม 11 บิต ประกอบด้วยบิตเริ่มต้น มีค่าเป็น 0 จำนวน 1 บิต บิตข้อมูล 8 บิตโดยทำการรับส่งขอมูล LSB ก่อน บิตข้อมูลบิตที่ 9 และบิตปิดท้ายมีค่าเป็น 1 จำนวน 1 บิต ในการส่งข้อมูลบิตที่ 9 จะได้รับการเก็บไว้ใน TB8 ในรีจิสเตอร์ SCON และในการรับข้อมูล ข้อมูลบิตที่ 9 จะถูกเก็บไว้ในบิต RB8 ในรีจิสเตอร์ SCON สำหรับอัตราบอดในโหมด 2 เลือกได้ 2 ค่า คือ 1/32 หรือ 1/64 ของความถี่สัญญาณนาฬิกา สำหรับในโหมด 3 อัตราบอดสามารถปรับได้เหมือนกับในโหมด 1

6.2.2 การนำพอร์ตอนุกรมไปใช้งาน

การเขียนหรือการส่งข้อมูลออกจากพอร์ตอนุกรม

ข้อมูลที่ต้องการส่งออกทุกค่าต้องนำไปไว้ที่รีจิสเตอร์บัฟเฟอร์ที่เกี่ยวข้องพอร์ตอนุกรม ซึ่งก็คือรีจิสเตอร์ SBUF ดังตัวอย่าง

```
MOV   SBUF, #'A'
```

จากคำสั่งข้างต้นเป็นการส่งข้อมูลด้วยอักษร A ออกไปยังพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ อย่างไรก็ตามก่อนทำการส่งข้อมูลทุกครั้ง ต้องแน่ใจว่า TI เคลียร์หรือมีค่าเป็น 0 และเมื่อทำการส่งข้อมูลเรียบร้อยแล้ว ก็จะทำการเซตบิต TI เพื่อให้ทราบ ดังตัวอย่างโปรแกรมดังต่อไปนี้

```
CLR   TI
MOV   SBUF, #'A'
JNB   TI, $
```

การอ่านหรือรับข้อมูลจากพอร์ตอนุกรม

การรับข้อมูลจากพอร์อนุกรมสามารถทำได้ง่ายมาก เพียงทำการตรวจสอบว่าบิต RI เกิดการเซตขึ้นหรือไม่ ถ้าพบว่ามีการเซตเกิดขึ้นแล้ว ให้ทำการอ่านค่าจากรีจิสเตอร์ SBUF โดยต้องทำการโอนย้ายข้อมูลผ่านทางแอกคิวมูเลเตอร์หรือรีจิสเตอร์ A ดังตัวอย่าง

```
CLR   RI
JNB   RI, $
MOV   SBUF, A
CLR   RI
```

การเชื่อมต่อกับพอร์ตอนุกรมของคอมพิวเตอร์

การใช้งานวงจรพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51 มักนิยมใช้ในการติดต่อเพื่อแลกเปลี่ยนข้อมูลกับคอมพิวเตอร์ผ่านทางพอร์ตอนุกรมในมาตรฐาน RS-232 มีตั้งแต่ระดับบวกลบ 3 ถึง 12V ในขณะที่ระดับสัญญาณของไมโครคอนโทรลเลอร์ MCS-51 อยู่ในระดับทีทีแอล ดังนั้นจึงไม่สามารถเชื่อมต่อพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51 เข้ากับพอร์ตอนุกรมของคอมพิวเตอร์ได้โดยตรง จึงต้องอาศัยการเชื่อมต่อผ่านไอซีพิเศษที่ทำหน้าที่ในการแปลงระดับสัญญาณ

ไอซีที่ทำหน้าที่ในการแปลงสัญญาณนี้ต้องทำการแปลงข้อมูลส่งของไมโครคอนโทรลเลอร์ MCS-51 จากระดับทีทีแอลไปเป็นระดับของ RS-232 และทำการแปลงข้อมูลรับจากคอมพิวเตอร์จากระดับของ RS-232 เป็นระดับทีทีแอลเพื่อให้ถ่ายทอดไปยังไมโครคอนโทรลเลอร์ MCS-51 ได้อย่างสมบูรณ์ ไอซีดังกล่างมีด้วยกันหลายเบอร์จากหลายผู้ผลิต อาทิ MAX232 จาก MAXIM ICL232 จาก HARRIS

6.3 ทฤษฎีการทำงานของเซ็นเซอร์

ในการทำงานของอุปกรณ์เซ็นเซอร์ใช้ในการควบคุมการป้อนกลับภายในและค่าจากภายนอก มนุษย์และสัตว์ต่างมีสัญชาตญาณการรับรู้เหมือนกันอย่างเช่น เมื่อตื่นนอน ก่อนที่จะลืมตาขึ้นนั้นจะรู้สึกและรับรู้ถึงแขนและขาคุณอยู่อย่างไร เพราะเส้นประสาทจะส่งสัญญาณไปยังสมอง และเช่นเดียวกันกับการเกร็งและคลายกล้ามเนื้อก็จะมีสัญญาณทีส่งไปยังสมองเช่นกัน

คล้ายกับหุ่นยนต์ที่แขนและข้อต่อจะมีการเคลื่อนไหวเซ็นเซอร์จำพวก Potentiometer ที่ทำการถอดรหัสและแปลงสัญญาณไปให้ส่วนควบคุมได้รับรู้ว่ามีการเคลื่อนไหวอย่างไร นอกจากนี้ในขณะที่มนุษย์และสัตว์รับรู้ได้ทางประสาทสัมผัสทั้ง 5 หุ่นยนต์ก็สามารถรับรู้ได้เช่นกัน เซ็นเซอร์ถูกออกแบบมาให้มีการทำงานเลียนแบบการรับรู้ของมนุษย์ และนอกจากนั้นยังสามารถรับรู้ได้มากกว่ามนุษย์อีกด้วย

6.3.1 ประเภทและคุณสมบัติทั่วไปของเซ็นเซอร์

คุณสมบัติทั่วไปของเซ็นเซอร์

ราคา เป็นสำคัญในการพิจารณาเลือกใช้ให้เหมาะสมกับอุปกรณ์ที่ต้องการ

ขนาด ขึ้นอยู่กับความสามารถของเซ็นเซอร์ ขนาดมีความสำคัญมาก อย่างเช่น อุปกรณ์ที่มีพื้นที่ขนาดเล็กจำเป็นต้องใช้เซ็นเซอร์ที่มีขนาดเล็กตามไปด้วย

น้ำหนัก จะต้องคำนึงถึงในอุปกรณ์ที่มีการเคลื่อนที่ได้ เพราะน้ำหนักจะมีผลเป็นการเพิ่มแรงเฉื่อยของการเคลื่อนที่

ชนิดของเอาท์พุท มีความสำคัญในการพิจารณาเอาท์พุทที่ได้จะนำไปใช้งานในรูปแบบต่าง ๆ กัน อย่างเช่น ดิจิตอลจะใช้เป็นเอาท์พุทใช้งานกับไมโครโพรเซสเซอร์

การเชื่อมต่อ จะต้องคำนึงถึงการเชื่อมต่อตัวอุปกรณ์และเซ็นเซอร์เช่น วงจรหรือสายสัญญาณต่าง ๆ

ความละเอียด เป็นการพิจารณาค่าเอาท์พุทของเซ็นเซอร์ว่าให้ความละเอียดได้มากเพียงใด เช่น เอาท์พุท 4 บิต ให้ความละเอียดได้ 16 ระดับ

ความอ่อนไหว เป็นอัตราการเปลี่ยนแปลงของเอาท์พุทที่มีผลมาจากการเปลี่ยนแปลงของอินพุทว่ามีผลมากน้อยเพียงใด

ความสัมพันธ์แบบเส้นตรง เป็นความสัมพันธ์ในการแปรผันตรงแบบเชิงเส้นระหว่างอินพุทและเอาท์พุท เมื่ออินพุทเข้าไปเท่าไรค่าก็จะเปลี่ยนเป็นเอาท์พุทออกมาในอัตราที่เท่าเดิมเสมอ

ช่วง เป็นช่วงที่แตกต่างกันระหว่างเอาท์พุทที่น้อยที่สุดและมากที่สุดที่เซ็นเซอร์สามารถจะรับได้

การตอบสนอง เป็นความรวดเร็วในการตรวจจับและแสดงค่าความต่างของเอาท์พุทเมื่อมีการเปลี่ยนแปลงของอินพุท

ความถี่ในการตอบสนอง ความสามารถในการรับความถี่ของสัญญาณอินพุทเนื่องจากบางสภาวะแวดล้อมที่เราต้องการตรวจจับ เช่น เสียง หรือคลื่นวิทยุจะต้องใช้ความถี่ในการจำแนกและตรวจจับ

ความน่าเชื่อถือ คืออัตราความถูกต้องของการทำงานต่อจำนวนครั้งที่ใช้งาน

ความแม่นยำ คือ ความถูกต้องของค่าเอาท์พุทที่ใกล้เคียงกับค่าที่คาดการณ์เอาไว้

การทำงานซ้ำ ความสามารถในการทำงานอย่างเดิมหลายครั้งและสามารถทำงานได้อย่างถูกต้องตามคุณสมบัติ

ประเภทของเซ็นเซอร์

เซ็นเซอร์วัดตำแหน่ง (Position sensors)

Potentiometer

Encoder

เซ็นเซอร์วัดความเร็ว (Velocity sensors)

Encoders

เซ็นเซอร์วัดแสงและอินฟราเรด (Light and Infrared sensors)

เซ็นเซอร์ประมาณระยะ (Proximity sensors)

Optical proximity sensors

6.3.2 ลักษณะการทำงานของตัวต้านทานปรับค่าได้

Potentiometer จำแปลงค่าตำแหน่งให้กลายเป็นการเปลี่ยนแปลงของค่าความต่างศักดิ์โดยมาจากค่าความต้านทานที่เปลี่ยนไป เช่น การเปลี่ยนแปลงค่าความต้านทานแคบก็จะเกิดจากการเปลี่ยนแปลงตำแหน่งในช่วงแคบเช่นกัน ซึ่งค่าความต้านทานก่อนและหลังที่เดิกการเปลี่ยนแปลงจะถูกกำหนดโดยช่วงทั้งหมดของตัวความต้านทานปรับค่าได้ โดยที่ความสามารถของ potentiometer จะทำงานแบบการแบ่งกระแสที่ไหลผ่านตัวมันเอง ทำให้สามารถคำนวณผลลัพธ์ได้จากความสัมพันธ์ของความต้านทาน คือ

$$V_{out} = Vcc*(R1/R)$$

Potentiometer มีทั้งแบบหมุนและแบบแถบเลื่อนขึ้นอยู่กับการใช้งานว่ามีการเคลื่อนไหวในแบบใด โดยในแบบหมุนได้นั้นมีแบบที่เป็นหมุนได้หลายรอบ ทำให้สามารถใช้ในอุปรณ์ที่มีการเคลื่อนแบบหมุนวนไหลรอบได้

Potentiometer มีทั้งแบบ wire wound และแบบ thin film deposit (ซึ่งก็ทำจากพลาสติก) โดยที่แบบ thin film deposit นั้นค่าความต้านทานที่ต่างกันจะอยู่ที่พื้นผิวหน้าสัมผัส โดยคุณลักษณะพิเศษของแบบ thin film deposit คือการให้ค่าเอาท์พุทที่ต่อเนื่องและมีสัญญาณรบกวนต่ำ ซึ่งก็เป็นผลให้สามารถ

นำไปใช้งานทำการหาอัตราความต่างของเอาท์พุท differentiate output ซึ่งจะสมารถนำไปหาอัตราเร็วต่อไป ซึ่งเอาท์พุทของแบบ wire wound เป็นแบบไม่ต่อเนื่องทำให้ไม่สามารถนำไปใช้งานแบบ หาอัตราความต่าเอาท์พุทได้

ทั่วไปแล้ว Potentiometer จะใช้ในการหาการป้อนกลับภายในเป็นลำดับเพื่อหาตำแหน่งปัจจุบันของแขนและจุดเชื่อมต่อของอุปกรณ์ Potentiometer นั้นสามารถใช้งานตามลำพังหรืออาจใช้งานร่วมกับตัวเซ็นเซอร์อื่นๆ ได้เช่น ตัวถอดรหัส ในกรณีนี้ ตัวถอดรหัสจะรายงานค่าตำแหน่งปัจจุบันของแขนและจุดเชื่อมต่อของอุปกรณ์ ดังเช่นการรายงานเมื่อตำแหน่งเริ่มต้นทำงาน ผลที่ได้จากการใช้งานร่วมกันหลายเซ็นเซอร์สิ่งที่ต้องการคือค่าของเอาท์พุทที่น้อยแต่มีความชัดเจนและเที่ยงตรงสูง

6.3.3 ลักษณะการทำงานของเซ็นเซอร์แสงและอินฟราเรด

เซ็นเซอร์ชนิดนี้จะทำงานโดยความเข้มของแสงที่ตกกระทบตัวมันเองจะเป็นตัวเปลี่ยนค่าความต้านทานทางไฟฟ้า ถ้าความเข้มแสงน้อยหรือต่ำจนเป็นศูนย์ค่าความต้านจะมีค่ามากที่สุด แต่เมื่อมีความเข้มแสงตกกระทบจะทำให้ค่าความต้านทางไฟฟ้าต่ำลงทำให้ เกิดกระแสไหลผ่านได้มาก ผลที่ได้คือ ค่าความค่าศักดิ์ที่ตกคร่อมจะมีค่าน้อย เซ็นเซอร์ชนิดนี้มีราคาไม่แพงและยังมีประโยชน์มากมายในการนำมาประยุกต์ใช้งาน ดดยสามารถนำมาสร้างเป็นตัวตรวจจับสัญญาณโดยแสงได้ และอุปกรณ์อื่นๆ อีกมากมาย ทั้งยังสมารถนำไปใช้งานในเซ็นเซอร์วัดพื้นผิวได้ด้วย

โดยตัวเซ็นเซอร์ที่ใช้กันนั้นคือ โฟโต้ทรานซิสเตอร์ ซึ่งจะทำงานเมื่อมีแสงที่มีความเข้มถึงขั้นมากระทบจะทำให้โฟโต้ทรานซิสเตอร์เปิด และเมื่อไม่มีแสงมากระทบหรือความเข้มน้อยกว่าที่ระบุก็จะทำให้อยู่ในสถานะปิดโดยปกติแล้วโฟโต้ทรานซิสเตอร์จะทำงานโดยใช้แหล่งกำเนิดแสงจำพวก LED

การประยุกต์ใช้งานโดยอาจมีการนำเซ็นเซอร์มาวางเรียงกันเพื่อใช้วัดระยะการเปลี่ยนตำแหน่งไปของตัวกำเนิดแสง ซึ่งจะนำไปใช้งานในการวัดค่าความผิดปกติหรือระยะการเคลื่อนที่น้อยๆ ของตัวเครื่องมือต่างๆ

เซ็นเซอร์แสงมีผลกระทบจากช่วงความถี่ของแสงปกติ เซ็นเซอร์อินฟราเรดก็จะได้รับผลกระทบจากช่วงของความถี่อินฟราเรดเช่นกัน เนื่องจากอินฟราเรดไม่สามารถมองเห็นได้โดยตาทำให้ไม้เดิกการรำคาญหากต้องใช้เป็นส่วนในอุปกรณ์สักอย่างที่มีการรับส่งแสงกัน เช่นว่า อุปกรณ์อย่างหนึ่งที่ต้องการแสงในการวัดระยะทางที่ไกลเพื่อนำมาสร้างเป็นระบบนำทาง ในที่นี้จะนิยมใช้อินฟราเรดมากกว่าเพราะว่าไม่เป็นที่สนใจและไม่รบกวนผู้อื่น หรือหลบเลี่ยงจากการก่อกวนได้

อุปกรณ์ที่ใช้อินฟราเรดเช่น รีโมตทีวี ซึ่งเป็นอุปกรณ์ที่มีประโยชน์มากในการควบคุมอุปกรณ์ทำงานต่างๆ ได้จากระยะไกล

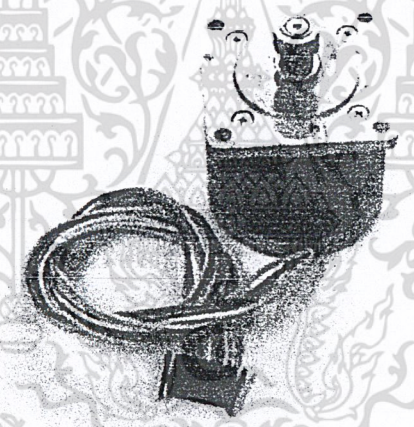
6.4 ทฤษฎีการทำงานของส่วนเคลื่อนไหว

มอเตอร์ไฟฟ้า (Electric Motors)

มอเตอร์เป็นอุปกรณ์ที่แปลงกระแสไฟฟ้าไปเป็นแรงบิดที่เกิดจากการหมุน โดยการทำงานโดยการผ่านกระแสไฟฟ้าไหลตัดสนามแม่เหล็กทำให้เกิดแรงบิด โดยมอเตอร์ไฟฟ้าจะมองว่าใช้ ไฟฟ้ากระแสตรง และกระแสสลับ การแบ่งมอเตอร์ไฟฟ้าตามคุณสมบัติการทำงานได้หลายแบบ ดังนี้

6.4.1 มอเตอร์กระแสตรง (DC Motors) เป็นมอเตอร์ที่ใช้กันอย่างแพร่หลายทั่วไปในโรงงานอุตสาหกรรมเป็ระยะเวลานาน เนื่องจากเชื่อถือได้ ทนทานและมีกำลังมากพอใช้งาน การทำงานโดยตัวโครงมอเตอร์ (stator) เป็นส่วนที่ถูกยึดกับที่และเป็นแม่เหล็กเพื่อสร้างสนามแม่เหล็กที่คงที่ และตัวหมุนจะอยู่ส่วนกลางสนามแม่เหล็ก โดยมีช่องอากาศโดยรอบ การทำงานโดยจ่ายกระแสไฟฟ้าตรงผ่านลวดตัวนำที่พันรอบตัวกลางทำให้เกิดแรงบิด

6.4.2 สเตปปิงมอเตอร์ (Stepping Motors)



รูปที่ 6-4 แสดงสเต็ปปิงมอเตอร์

ปกติแล้วสเต็ปปิงมอเตอร์ไม่ต้องใช้ระบบป้อนกลับ เนื่องจากการที่การเคลื่อนของมันในแต่ละสเต็ปมีความคลาดเคลื่อนน้อยมาก โดยจะรู้ว่าแต่ละสเต็ปจะหมุนไปกี่องศาเนื่องจากองศาที่แน่นอนไม่จำเป็นต้องใช้ตัวตรวจสอบ รูปแบบการใช้งานมีมากมายหลายแบบ โดยแต่ละแบบจะมีคุณลักษณะที่ต่างกันแล้วแต่จุดประสงค์การนำไปใช้งานโดยความแต้กต่างจะขึ้นกับการวางตัวของขดลวด

สเต็ปปิงมอเตอร์จะเหมือนกับมอเตอร์กระแสตรงที่ไม่มีแปรงคือว่าจะต้องได้รับการควบคุมจากไมโครคอนโทรลเลอร์เท่านั้น ไม่สามารถจ่ายกระแสควบคุมโดยตรงได้ เมื่อเทียบกับเซอร์โวมอเตอร์นั้นที่ต้องมีระบบป้อนกลับควบคุมซึ่งเป็นการยุ่งยากพอกัน ฉะนั้นในการทำงานในโรงงานผู้ออกแบบต้องออกแบบให้เหมาะกับการทำงานของมอเตอร์มากที่สุด เช่น โรงงานอาจมองว่าสเต็ปปิงมอเตอร์ใช้งานยาก

แต่สเต็ปปิงมอเตอร์จะถูกใช้งานมากภายนอกโรงงานอย่างเช่น หุ่นยนต์ หรืออุปกรณ์ต่างๆ ที่ใช้ควบคุมระบบอีกที

โครงสร้างของสเต็ปปิงมอเตอร์ จะมีแกนหมุนเป็นแม่เหล็กถาวรและมีตัวมอเตอร์ที่มีขดลวดพันอยู่หลายรอบและหลายขด การออกแบบทำให้ความร้อนถ่ายเทออกโดยง่ายทำให้ไม่ได้รับความเสียหายจากความร้อน การไม่มีแปรงและแหวนทำให้มีอายุการใช้งานยาวนาน ในต่างประเภทกันแกนที่เป็นแม่เหล็กจะมีลักษณะที่ต่างกันไป

การทำงานมอเตอร์ ที่มีจำนวนขดลวด 4 ขด และมีแกนหมุนเป็นแม่เหล็กถาวร ในตอนแรงการจ่ายไฟให้2 ขั้วที่อยู่ตรงข้ามกันทำให้เกิดสนามเหนียวนำแกนเหล็ก จากนั้นจ่ายกระแสให้ 2 ขั้วที่ถัดไปทำให้เกิดการเหนียวนำต่อเนื่องกันไป โดยการออกแบบต่างๆ มีผลต่อคุณสมบัติมอเตอร์ เช่น การออกแบบจำนวนของยักบนแกนหมุน และจำนวนหยักบนตัวโครงมอเตอร์ โดยเป็นไปตามสูตร

องศาของแต่ละสเต็ป = (360/จำนวนหยัก) – (360/จำนวนหยัก)

ตัวอย่างเช่น สเต็ปปิงมอเตอร์ตัวหนึ่งมีหยักที่แกน 50 หยัก ที่ตัวโครงมอเตอร์ 40 หยัก จะมีองศาการหมุนที่สเต็ปละ = 360/40 – 360/50 = 9 – 7.2 = 1.8 องศา

ชนิดสเต็ปมอเตอร์ ที่พบในปัจจุบันมี 3 ลักษณะดังนี้

1. แบบแม่เหล็กถาวร (PERMANENT MAGNET_PM) สเต็ปมอเตอร์แบบ PM จะมีสเตเตอร์ (STATOR) ที่พันขดลวดไว้หลายๆ โพล โดยมีโรเตอร์ (ROTOR) เป็นรูปทรง กระบอกฟันเลื่อย และโรเตอร์ทำด้วยแม่เหล็กถาวร เพื่อป้อนไฟกระแสตรง ให้กับขดสเตเตอร์ จะทำให้เกิดแรงแม่เหล็กไฟฟ้าผลักต่อโรเตอร์ ทำให้มอเตอร์หมุนมอเตอร์แบบ PM จะเกิดแรงฉุดยึดให้โรเตอร์หยุดอยู่กับที่ แม้จะไม่ได้ป้อนไฟเข้าขดลวด

2. แบบแปรค่ารีลักแตนซ์ (VARIABLE RELUCTANCE- VR) สเต็ปมอเตอร์แบบVR จะมีการหมุนโรเตอร์ได้อย่างอิสระ แม้จะไม่ได้จ่ายไฟให้โรเตอร์ทำจากสารเฟอร์โรแมกเนติก กำลังอ่อน มีลักษณะเป็นฟันเลื่อย รูปทรงกระบอกโดยจะมีความสัมพันธ์ โดยตรงกับจำนวนโพลในสเตเตอร์ แรงบิดที่เกิดขึ้นจะไปหมุนโรเตอร์ ไปในเส้นทางของอำนาจแม่เหล็กที่มีค่ารีลักแตนท์ต่ำที่สุด ตำแหน่งที่จะเกิดแน่นอนและมีเสถียรภาพแต่จะเกิดขึ้นได้หลายๆ จุดดังนั้นเมื่อป้อนไฟเข้าขดลวดต่างๆ ในมอเตอร์แตกต่างขดกันไป ก็ทำให้มอเตอร์ หมุนไปตำแหน่งต่างๆ กันโรเตอร์ของ VR จะมีความเฉื่อยของโรเตอร์น้อยจึงมีความเร็วรอบสูงกว่ามอเตอร์แบบ PM

3. แบบผสม (HYBRID-H) สเต็ปมอเตอร์แบบ H จะเป็นลูกผสมของ VR กับ PM โดยจะมีสเตเตอร์คล้ายกับที่ใช้ใน VR โรเตอร์มีหมวกหุ้ม ปลายซึ่งมีลักษณะของสารแม่เหล็กที่มีกำลังสูง โดยการควบคุมขนาดรูปร่างของหมวกแม่เหล็กอย่างดีทำให้ได้มุม การหมุนและครั้งน้อยและแม่นยำ ข้อดีก็คือ ให้แรงบิดสูงและมีขนาดกระทัดรัด และให้แรงฉุดยึดโรเตอร์นิ่งกับที่ตอนไม่จ่ายไฟ

การสั่งการทำงาน ของสเต็ปปิงมอเตอร์ เชื่อว่านักอิเล็กทรอนิกส์ทุกท่านคงเคยได้ยินหรือได้เคยใช้งานสเต็ปมอเตอร์กันมาบ้างแล้ว ชื่อของเจ้าสเต็ปมอเตอร์นั้นก็ค่อนข้างที่จะบ่งบอกถึงคุณสมบัติของตัวมันได้ดีอยู่แล้ว ว่าเป็นมอเตอร์ที่สามารถเคลื่อนที่ในลักษณะที่เป็นทีละสเต็ป (Step) ซึ่งต่างจากมอเตอร์ทั่วไปที่การเคลื่อนที่ของมันจะเป็นแบบต่อเนื่อง และสเต็ปมอเตอร์ยังสามารถควบคุมการทำงานที่ง่ายด้วย

การต่อร่วมกับอุปกรณ์ภายนอกเพียงเล็กน้อย ดั้งนั้นเราจะเห็นว่าสเต็ปมอเตอร์ถูกนำไปใช้งานอย่างกว้างขวางไม่ว่าจะเป็นงานด้าน robot แขนกล plotter หรืองานใดๆที่ต้องใช้การเคลื่อนที่ในลักษณะของแบบสเต็ปการเรียนรู้หลักการทำงานของสเต็ปมอเตอร์ก็ย่อมก่อให้เกิดประโยชน์แก่เราให้สามารถนำสเต็ปมอเตอร์ไปประยุกต์ใช้งานได้

ลักษณะของสเต็ปมอเตอร์ภายนอกก็จะประกอบไปด้วยสายไฟที่เราจะต้องป้อนสัญญาณพัลส์ (pulse) เข้าไปควบคุมมัน ถ้าหากสเต็ปมอเตอร์ของคุณมี 3 ขดลวด ก็จะได้ลักษณะการต่อดังรูป ซึ่งจะมีสายหนึ่งที่ต่อเป็นกราวนด์ร่วมของทั้ง 3 สายนั้น ลักษณะแบบนี้คือเป็นแบบ 3 เฟส โดยที่สเต็ปมอเตอร์ที่ขายทั่วๆไปก็อาจจะมีหลายแบบเช่นแบบ 4 เฟส 5 เฟส ซึ่งเมื่อเราทราบเฟสแล้วสิ่งที่จะต้องทำต่อไปคือการหาว่าเฟสใดเป็นเฟส 1 2 หรือ 3 ซึ่งที่ตัวของสเต็ปมอเตอร์อาจจะมีบอกอยู่แล้ว หรือถ้าไม่มีวิธีง่ายๆก็คือการใช้ไฟขนาดที่มอเตอร์ตัวนั้นใช้ป้อนเข้าไปที่เฟสแต่ละเฟส ถ้าหากเราป้อนเป็นเฟส 1-2-3 แล้วสเต็ปมอเตอร์จะต้องหมุนไปในทิศทางเดียวกันหรือถ้าหากป้อนเป็น 3-2-1 จะต้องหมุนกับทางกันซึ่งเป็นวิธีง่ายๆในการตรวจสอบเฟสของสเต็ปมอเตอร์

การป้อนพัลส์ วิธีการง่าย ๆ ที่จะให้สเต็ปมอเตอร์หมุนได้ในลักษณะของการหมุนทวนเข็มหรือการหมุนตามเข็มคือ การป้อนพัลส์เข้าที่เฟสโดยเรียงลำดับกันคือ

1-2-3-1-2-3-1-2-3

ซึ่งลักษณะของพัลส์ที่ป้อนก็คือ

เฟส 1 100100100100100100 Times------>

เฟส 2 010010010010010010 Times------>

เฟส 3 001001001001001001 Times------>

และถ้าต้องการให้สเต็ปมอเตอร์หมุนกับทางกันลักษณะการป้อนก็จะเป็นการป้อนโดยเรียงลำดับเฟสย้อนกลับ

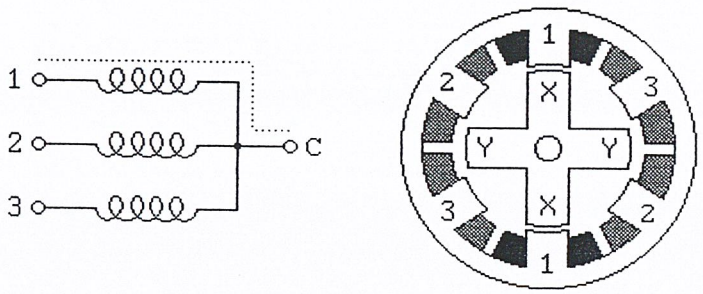
3-2-1-3-2-1-3-2-1

ซึ่งลักษณะของพัลส์ที่ป้อนคือ

เฟส 3 100100100100100 Times------>

เฟส 2 010010010010010 Times------>

เฟส 1 001001001001001 Times------>



รูปที่ 6-5 แสดงการป้อนพัลส์ให้กับสเต็ปมอเตอร์ และการหมุน

1. เมื่อ เฟส 1 ถูกป้อนพัลส์เข้ามาทำให้เกิดสนามแม่เหล็กที่ stator ที่ขดลวด 1 และจะผลักกับโรเตอร์ (Rotor) ให้เคลื่อนที่ไปคนละทาง

2. เมื่อ เฟส 2 ถูกป้อน ก็จะทำให้เกิดสนามแม่เหล็กขึ้นที่ขด 2 และก็จะผลักโรเตอร์ให้เกิดการเคลื่อนที่ไปอีก

3. เมื่อ เฟส 3 ถูกป้อน ก็จะทำให้เกิดสนามแม่เหล็กขึ้นที่ขด 3 และจะผลักให้โรเตอร์เกิดการเคลื่อนที่ไปอีก (สังเกตจากรูป การเคลื่อนที่ของสนามแม่เหล็กที่เกิดขึ้นและโรเตอร์จะหมุนไปทิศทางที่ตรงข้ามกัน)

6.4.3 ลักษณะการทำงานของ gear motor

จะเป็นมอเตอร์ชนิดหนึ่งที่เป็นการนำมอเตอร์กระแสตรงมาประยุกต์ต่อโดยการใส่ความสามารถในการควบคุมรอบความเร็ว โดยภายในตัวจะมีการติดตั้งตัวทดรอบไว้ การใช้งานคือ การป้อนกระแสไฟฟ้ากระแสตรงจะทำให้มอเตอร์หมุนโดยความเร็วรอบของมอเตอร์จะคงที่ และการควบคุมความเร็วรอบของมอเตอร์จากความต่างศักดิ์ที่จ่ายให้ โดยกระแสจะเป็นตัวกำหนดกำลังของมอเตอร์ ว่าสามารถให้แรงบิตมากเพียงใด

6.4.4 ลักษณะการทำงานของ เซอร์โวมอเตอร์ (Servo Motors)

เซอร์โวมอเตอร์เริ่มต้นมาจากการที่มีการศึกษาแรงในระบบมอเตอร์โดยแรงที่พิจารณาคือ แรงกลับทางไฟฟ้า Back electromotive force หรือ Back emf โดยทฤฎีเบื้องต้นเมื่อมีการจ่ายกระแสให้กับลวดตัวนำที่ตัดผ่านสนามแม่เหล็กจะทำให้เกิดแรงเคลื่อนตัว แต่ในทางกลับกันเมื่อทำให้ลวดตัวนำเคลื่อนที่ผ่านสนามแม่เหล็กทำให้เกิดกระแสไฟฟ้าขึ้นเป็นตัวกำเนิดไฟฟ้า อย่างไรก็ตามเมื่อมีกระแสไหลผ่านสนามแม่เหล็กมันจะทำให้เกิดกระแสในทางตรงกันข้ามเสมอ ซึ่งกระแสนี้เรียกว่า back electromotive force ซึ่งจะเป็นตัวลดประสิทธิภาพของมอเตอร์ มอเตอร์ที่หมุนด้วยความเร็วสูงจะมีแรง back emf มากตามไปด้วยโดยมีสูตรคือ

$$V_{cmf} = n \times Ke$$

โดยที่ค่า Ke เป็นค่าของโวลต์ที่ทำให้มอเตอร์หมุนที่ความเร็ว 1000 รอบต่อนาที โดยให้การหมุนของมอเตอร์ในขณะที่ไม่มีการแบกน้ำหนักเป็นปกติ ค่า back emf เป็นค่าที่มากพอที่จะทำให้ มอเตอร์ที่ไม่มีน้ำหนักถ่วงหมุนด้วยความเร็วที่คงที่ได้โดยจะทำให้ เกิดค่าเอาท์พุททอร์กของมอเตอร์มีค่าเป็น 0 โดยที่ค่าการหมุนของมอเตอร์จะมีสูตรความเร็วโดย

$$V_{in} = I \times R + V_{emf} = I \times R + n \times Ke$$

โดยที่ R เป็นค่าความต้านทานของช่องอากาศ เมื่อมีการแบกน้ำหนักทำให้มอเตอร์ช้าลงและยังทำให้ back emf ลดลงด้วย ทำให้การใช้งานประแสอย่างมีประสิทธิภาพและทำให้เกิดแรงบิด เมื่อแบกน้ำหนักมากขึ้นอีกมอเตอร์จะช้าลงและทำให้เกิดแรงบิดมหาศาล เมื่อทำให้น้ำหนักค่อยๆ เพิ่มขึ้นทำให้เกิดสภาวะที่ไม่มี back emf ทำให้การใช้งานกระแสได้สูงสุดและมีแรงบิดมากสุด แต่ข้อที่ไม่ดีก็คือเมื่อมอเตอร์เข้าสู่สถานะที่ให้แรงบิตมากสุด และมี back emf น้อยสุดก็ตาม มันจะนำมาซึ่งความร้อนที่สูงมากอาจจะทำให้เกิดความเสียหากับตัวมอเตอร์ได้

ในการสร้างแรงบิตที่มากขึ้นโดยไม่ให้ความเร็วลดลงอาจทำได้โดยการเพิ่มกระแส เช่น ในกรณีที่ความเร็วยังคงที่ และ back emf คงที่ การเพิ่มกระแสทำให้เกิดประสิทธิภาพที่เพิ่มขึ้นระหว่างกระแสที่ปอนกับแรงบิต โดยการจ่ายกระแสที่มีขนาดต่าง ๆ กัน ตามความเหมาะสมสมดุลระหว่างความเร็วกับแรงบิด ระบบนี้เรียกว่า เซอร์โวมอเตอร์

โดยมีสูตรหาค่าแรงบิดจาก

$$T = T_L + T_F + nK_D = IK_T$$

โดยที่ T เป็นเอาท์พุทแรงบิด T_L เป็นแรงบิดที่เกิดจากน้ำหนัก T_F เป็นแรงบิดที่เกิดจาดแรงเสียดทาน nK_D แรงบิดที่เกิดจากความเร็วของการหมุน และ K_T คือค่าคงที่ของแรงบิด

เซอร์โวมอเตอร์มีเป็นทั้ง มอเตอร์กระแสตรง มอเตอร์กระแสสลับ มอเตอร์ที่ไม่มีแปรง หรือแม้แต่สเตปปิงมอเตอร์ โดยสามารถมีป้อนกลับมาควบคุมความเร็วหรือแรงบิด การหมุนได้ โดยการออกแบบให้อุปกรณ์ป้อนกลับส่งสัญญาณมารายงานตำแหน่งและความเร็ว เช่นหากว่าน้ำหนักมากจนทำให้ความเร็วลดลงกว่าที่ออกแบบไว้ระบบจะทำการเพิ่มกระแสทำให้ความเร็วกลับมาเป็นตามที่กำหนดไว้ และถ้าหากว่ามอเตอร์หมุนเร็วกว่าที่กำหนดจากการที่น้ำหนักน้อยระบบจะลดกระแสที่ป้อนทำให้ความเร็วลดลง การวัดตำแหน่งก็เช่นกันเมื่อถึงเป้าหมายสัญญาณจะถูกส่งมาทำให้มอเตอร์หยุดการทำงาน

6.4.5 มอเตอร์กระแสสลับ (AC Motors) เป็นมอเตอร์ที่คล้ายกับมอเตอร์กระแสตรงมาก แต่จะต่างกันตรงที่ ตัวโครงมอเตอร์จะเป็นส่วนของขดลวด ตัวกลางหมุนจะเป็นแม่เหล็ก และส่วนที่เป็นวงแหวนและตัวแปลงจะไม่มีติดตั้ง

บทที่ 7
การพัฒนาในส่วนของฮาร์ดแวร์

7.1 ภาพรวมการออกแบบระบบ

จะแบ่งงานออกเป็น 4 ส่วน คือ

1) ส่วนส่งข้อมูลจากจักรยาน

2) ส่วน ตอบสนองกลับไปที่จักรยาน

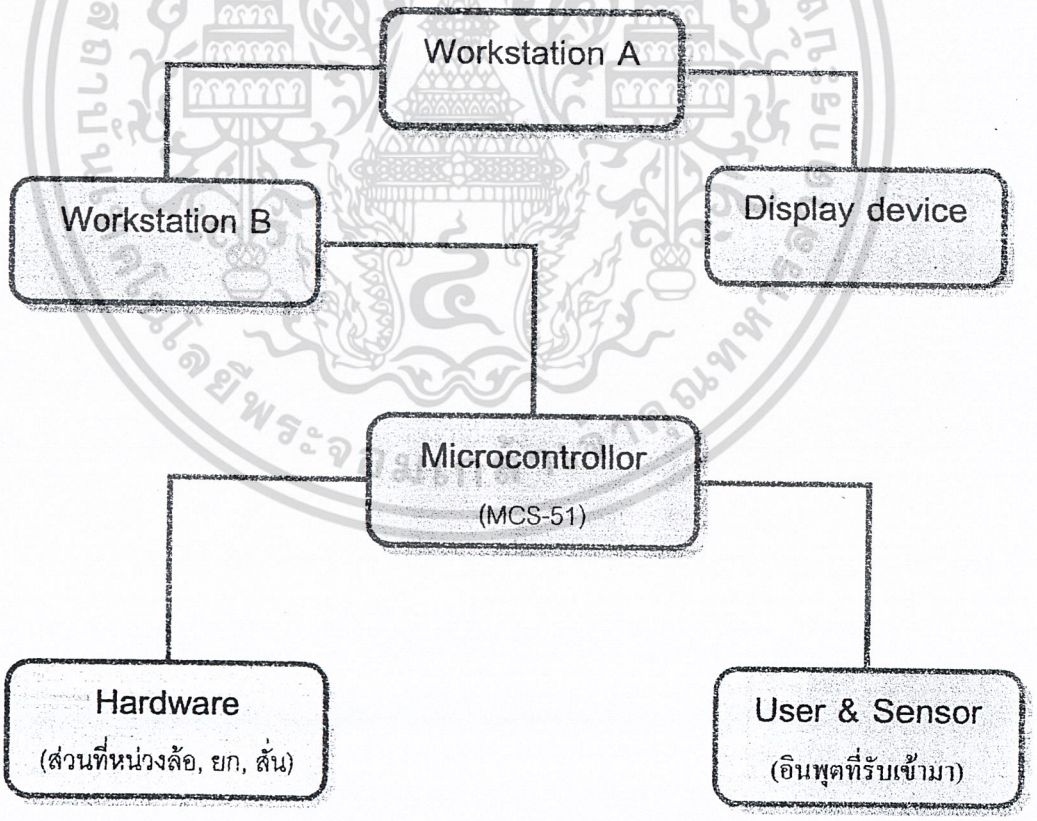
3) ส่วนสร้างภาพ 3 มิติ

4) ส่วนรับส่งข้อมูลกับจักรยาน

โดยที่จะแบ่งงานกันทำคนละส่วน และใช้คอมพิวเตอร์ 2 เครื่อง

เครื่องหนึ่งใช้สำหรับติดต่อรับส่งข้อมูลกับจักรยาน ประมวลผลข้อมูลที่ได้รับจากจักรยานเพื่อส่งให้คอมพิวเตอร์อีกเครื่อง และส่ง feedback จากโปรแกรมกลับไปหาจักรยาน

อีกเครื่องหนึ่งใช้สำหรับสร้างภาพ 3 มิติไปแสดงบนฉาก โดยคอมพิวเตอร์ทั้งสองจะรับส่งข้อมูลระหว่างกัน ดังรูปที่ 7.1



รูปที่ 7-1 การรับส่งข้อมูลระหว่างอุปกรณ์ต่าง ๆ ของระบบ

คำจำกัดความของรูปที่ 7.1

Workstation A เป็นเครื่องคอมพิวเตอร์ที่ ใช้ในการประมวลผลโปรแกรม Virtual Bicycle และประมวลผลภาพ รันโปรแกรมรับ-ส่งข้อมูล กับ Workstation B ใช้ control feedback ที่เกิดจากโปรแกรมไปยัง Workstation B

Workstation B เป็นเครื่องคอมพิวเตอร์ที่ใช้ในการประมวลผลสัญญาณในการรับ-ส่งข้อมูล จาก Workstation A ไปยัง Microcontrollor รันโปรแกรมรับ-ส่งข้อมูล กับ Workstation A และสร้างการส่งข้อมูลแบบอนุกรมกับ Microcontrollor

Display device อุปกรณ์ที่ใช้แสดงผลสัญญาณภาพต่างๆ ที่ได้จากการ Workstation A ซึ่งจะมีขอมูลที่สำคัญๆ คือ ภาพที่ต้องการให้แสดงออกเป็น Virtual, console ต่างๆ ที่ใช้ในการควบคุมโปรแกรม, ข้อมูลสถานะที่จำเป็นต้องแสดงต่างๆ เช่น การเต้นของหัวใจ, ความเร็ว และค่าพลังงานที่ใช้ไป

Microcontrollor คือตัวประมวลผลที่ทำหน้าที่ความคุมฮาร์ดแวร์ รับอินพุตจาก User และ Seneor ติดต่อกับ Workstation B โดยการส่งและรับข้อมูลแบบอนุกรม

User & Sensor ประกอบด้วยสัญญาณที่ได้จากการวัดการเลี้ยวซ้าย-ขวา ของผู้เล่น สัญญาณที่ได้จากการวัดความเร็วของจักรยานที่ผู้เล่นขี่โดยวัดจากล้อที่หมุน สัญญาณที่ส่งข้อมูลจากเครื่องมือที่ใช้วัดการเต้นของหัวใจของผู้ใช้งาน

Hardware ประกอบด้วยส่วนที่ควบคุมให้อุปกรณ์ที่จักรยานหน่วงการหมุนของล้อ ให้รู้สึกเหนือมเวลาเบรกจักรยานจริง ส่วนที่ควบคุมให้กลไกที่ควบคุมโครงจักรยานให้ยกในองศาต่าง เหมือนเวลาที่ขี่จักรยานขึ้นทางชัน และลงเนิน ส่วนที่ควบคุมให้กลไกทำการสั่นจักรยานเพื่อให้สมจริงในสถานการณ์ ที่ผู้ขี่จักรยานขี่จักรยานเข้าในสภาวะแวดล้อมที่มีพื้นผิวไม่เรียบ

ฮาร์ดแวร์แบ่งออกเป็น 2 ส่วนใหญ่ๆ ด้วนกันคือ ส่วนของอินพุท และส่วนของเอาท์พุท

ส่วนของอินพุทประกอบด้วย

1) ส่วนตรวจสอบการเลี้ยวของแฮนด์จักรยาน

2)ส่วนตรวจสอบการหมุนของล้อหลังของจักรยาน

3)ส่วนตรวจสอบการเต้นของชีพจร

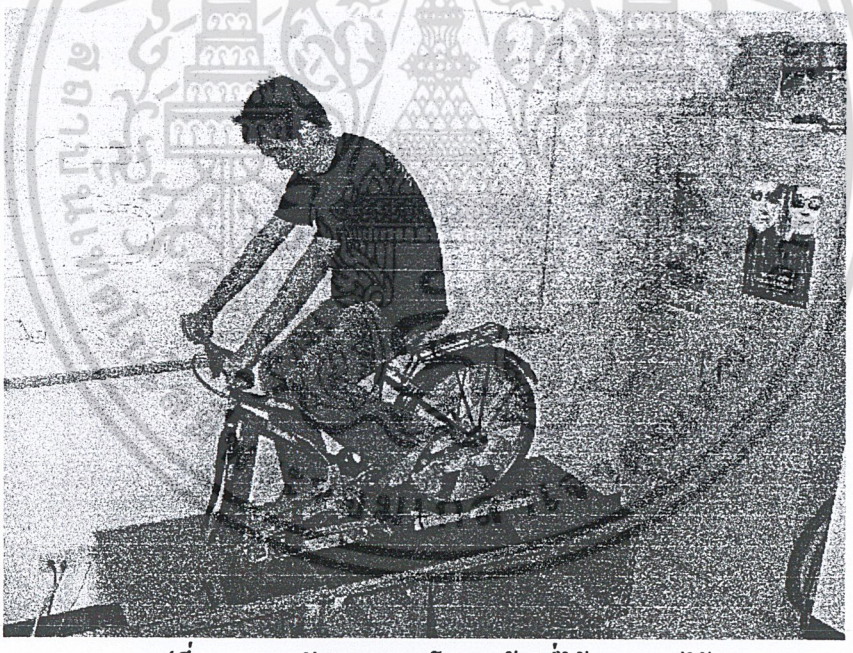
ส่วนของเอาท์พุทประกอบด้วย

1) ส่วนที่ควบคุมการหน่วงล้อ

2) ส่วนที่ควบคุมการยกจักรยาน

3) ส่วนที่ควบคุมการสั่นของจักรยาน

ฮาร์ดแวร์จะต้องตอบสนองระบบได้อย่างถูกต้องและรวดเร็ว เพื่อจะทำให้ระบบทำงานราบรื่นอย่างเป็นธรรมชาติ



รูปที่ 7-2 แสดงจักรยานและโครงสร้างที่ได้ออกแบบไว้

7.2 อุปกรณ์ที่ใช้ในการพัฒนาทั้งหมด

1) รถจักรยาน

2) ฐานจักรยาน

3) โซ่คจักรยานยนต์

4) บอร์ดไข่ปลา

5) หม้อแปลงไฟ 220V --> 24V, 12V, 6V

6) มอเตอร์กระแสตรง 24V (DC Motor)

7) มอเตอร์กระแสสลับ 220V (AC Motor)

8) สแต็ปปิงมอเตอร์ 24V (Stepping Motor)

9) แกนเหล็กเกลียว

10) สายพาน

11) ดัมพ์เบรกจักรยาน

12) ตัวสร้างความถี่ 12 MHz (Crystal)

13) รีเลย์ (Relay)

14) เซ็นเซอร์แสง (Photodiode)

15) ไมโครคอนโทรลเลอร์ MCS-51

16) สวิตท์

17) หัวต่อ (connector)

18) หัว DB – 9

19) ตัวแปลงข้อมูลในการส่งข้อมูลแบบอนุกรม MAX-232

20) ตัวแปลงสัญญาณอนาล็อกเป็นดิจิตอล (Analog to Digital Converter) ADC 0804

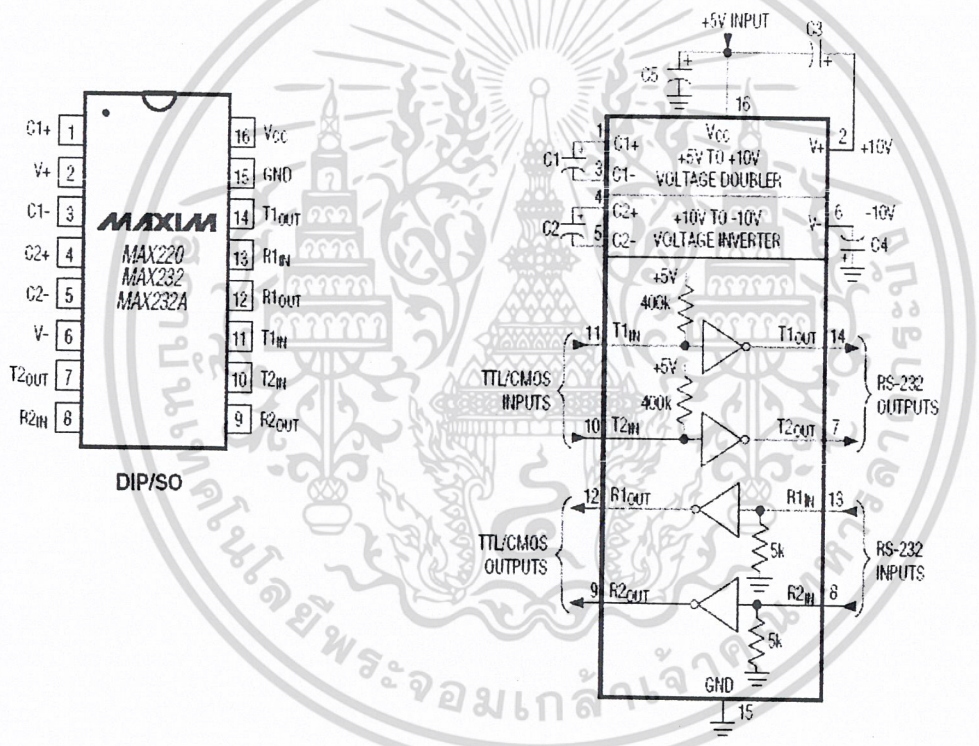
7.3 ส่วนตัวกลางรับ-ส่งของข้อมูลระหว่าง ตัวจักรยานกับคอมพิวเตอร์

7.3.1 พอร์ตอนุกรม

การรับส่งข้อมูลแบบอนุกรม ซึ่งมีข้อดีคือ ใช้สายสัญญาณน้อยและส่งได้เป็นระยะทางไกล สำหรับโครงงานนี้ใช้การติดต่อกับไมโครคอนโทรลเลอร์ รับส่งข้อมูลแบบอนุกรรมกับคอมพิวเตอร์ เพื่อทำการเปลี่ยนระดับสัญญาณไฟฟ้าได้มาตรฐานในการรับส่งข้อมูลผ่านทางพอร์อนุกรม (Serial Port)

การติดต่อระหว่างคอมพิวเตอร์กับฮาร์ดแวร์จะติดต่อโดยส่งข้อมูลแบบอนุกรมผ่านพอร์ตอนุกรม โดยใช้ MAX-232 เป็นตัวแปลงข้อมูลในการรับและส่งแบบอนุกรมทั้งขาเข้าและขาออกให้เป็นมาตรฐานการรับส่งแบบ RS-232 ขาเข้าจะรับข้อมูลมาจากไมโครคอนโทรลเลอร์ แล้วส่งไปพอร์ตอนุกรมเข้าคอมพิวเตอร์ ขาเข้า

MAX-232 จะแปลงข้อมูลแล้วส่งข้อมูลเข้าไมโครคอนโทรลเลอร์



รูปที่ 7-3 แสดงไอซี MAX-232

การกำหนดขา RS-232 สำหรับ DB-9

การกำหนดขาสัญญาณของคอนเน็กเตอร์ (Connector) อนุกรม 9 ขา (DB-9) แสดงในตารางที่ 7-1

ขา	ฟังก์ชั่น
1	Received Line Signal Detect
2	Received Data
3	Transmit Data
4	Data Terminal Ready
5	Signal Ground
6	Data Set Ready
7	Request to Send
8	Clear to Send
9	Ring Indicator

ตารางที่ 7-1 แสดงการกำหนดขาสัญญาณคอนเน็กเตอร์อนุกรม 9 ขา

คำอธิบายการทำงานของแต่ละขาที่ใช้งานของ DB-9

Transmit Data (TD) เป็นสัญญาณที่ส่งออกจาก DTE (หรือตัวไมโครคอมพิวเตอร์พิวเตอร์) ไปยังโมเด็ม หรือต่อกับไมโครคอมพิวเตอร์ตัวอื่น เมื่อไม่มีสัญญาณส่งออกสถานะภาพของลอกจิกที่ขานี้มีค่าเท่ากับ"1" หรือเที่ยบเท่า Stop Bit

Received Data (RD) เป็นทางสัญญาณเข้าไปยัง DTE หรือไมโครคอมพิวเตอร์ เมื่อไม่มีสัญญาณ รับเข้ามาขานี้จะมีสถานะภาพลอจิกเป็น "1"

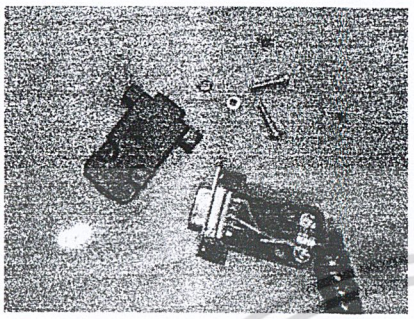
Request to Send (RTS) ใช้สำหรับส่งสัญญาณไปยังโมเด็มหรือเครื่องพิมพ์เป็นการเรียกร้องที่จะส่งสัญญาณมาทาง TD สัญญาณนี้จะใช้คู่กับ CTS ที่อุปกรณ์รับ หากได้รับ สัญญาณ RTS จะตรวจตัวเองว่าพร้อมจะรับสัญญาณได้หรือยัง หากพร้อมก็จะส่งสัญญาณออกไปที่สาย CTS

Clear to Send (CTS) เมื่อสายสัญญาณนี้อยู่ในสภาวะออฟ (Negative Voltage หรือลอจิก"1") หมายความว่า อุปกรณ์รับกำลังบอกว่า พร้อมจะรับข้อมูลแล้ว

Data Set Ready (DSR) เมื่อสายสัญญาณนี้อยู่ไนสภาวะออน (ลอกจิก"0") จะเป็นการบอกไมโครคอมพิวเตอร์ว่า พร้อมที่จะส่งได้แล้ว

Signal Ground (SG) ทำหน้าที่เป็นระดับแรงดันอ้างอิงสำหรับทุกๆสายสัญญาณจะมีแรงดันเป็น “0” เมื่อเทียบกับสายสัญญาณอื่นๆ

Data Terminal Ready (DTR) คอมพิวเตอร์เปิดสัญญาณนี้ให้ออน (ลอจิก”0”) เมื่อพร้อมที่จะติดต่อรับส่งข้อมูล



รูปที่ 7-4 แสดงการต่อสาย DB-9

7.3.2 ไมโครคอนโทรลเลอร์ MCS-51

Controller คือ ตัวควบคุมการทำงานของอุปกรณ์หรือขบวนการต่างๆ ซึ่งอาจทำขึ้นมาจากวงจรไฟฟ้ากลไก PLC ฯลฯ Micro-Controller ก็คือ อุปกรณ์ประเภทสารกึ่งตัวนำที่รวบรวมฟังก์ชั่นการทำงานต่างๆ ไว้ภายในตัวของมันเอง มีขนาดเล็ก และสามารถเขียนโปรแกรมควบคุมการทำงานของอุปกรณ์ต่างๆ ที่เชื่อมต่อกับตัวมัน โดยเน้นความสมบูรณ์ภายในตัวของมันเองและง่ายต่อการนำไปใช้งานหรือแก้ไขดัดแปลง

ไมโครคอนโทรลเลอร์ที่ใช้คือ AT89C51

คุณสมบัติของ MCS51

-เป็นชิปขนาด 40 ขา มีคุณลักษณะพิเศษหลายอย่างดังนี้

- มีหน่วยความจำโปรแกรมชนิดแฟลชเมโมรี (Flash Memory) หรือชนิดที่เขียนและลบได้รวดเร็ว ขนาด 4 กิโลไบต์

-ทนต่อการเขียนลบได้ 1000 ครั้ง และคงค่าข้อมูลไว้ได้นาน 10 ปี

- ทำงานที่ความถี่นาฬิกา 0-24 เมกะเฮิรตซ์

- ป้องกันการอ่านข้อมูลจากหน่วยความจำโปรแกรม (Program Memory Lock) ได้ 3 ระดับ

- มีหน่วยความจำแรม (Ram) ภายในตัวชิปอยู่ 128 ไบต์

- มีขาอินพุท/เอาท์พุท ที่สามารถโปรแกรมได้จำนวน 32 ขา

- มีตัวตั้ง/ตัวนับเวลาขนาด 16 ไบต์จำนวน 2 ตัว

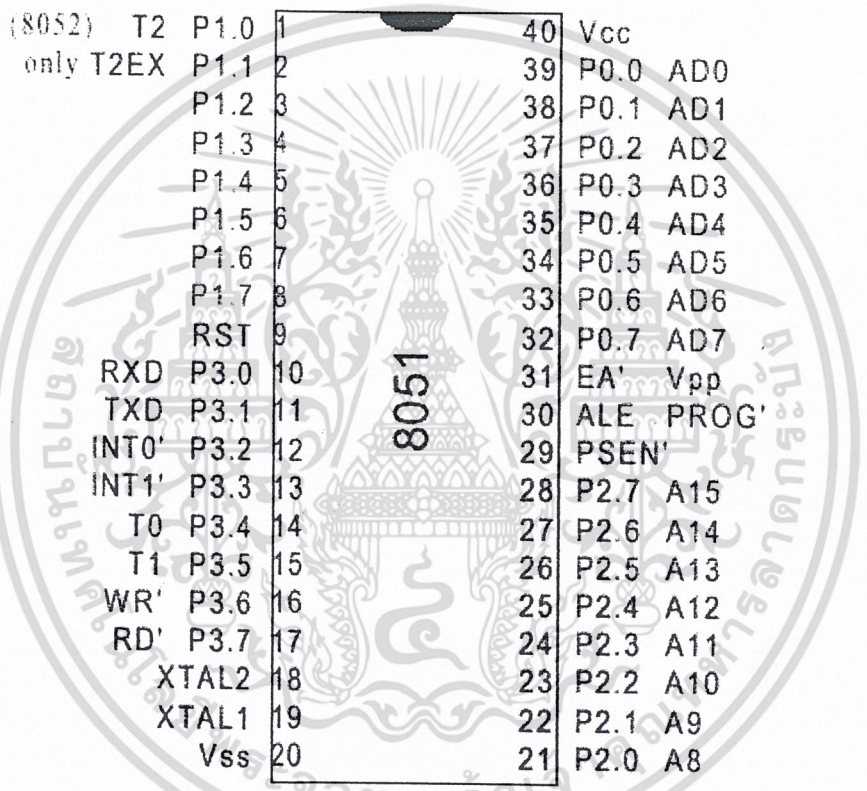
- สามารถรับการอินเตอร์รับ (Interrupt) หรือสัญญาณขัดจังหวะให้มีการทำงานก่อนได้จาก 5 แหล่ง

- มีช่องรับส่งข้อมูลแบบอนุกรมที่สามารถโปรแกรมได้

โครงสร้าง 8051

ลักษณะการจัดขาภายนอกของ MCS-51 การจัดขาตามลักษณะภายนอกของชิป MCS-51 จะมีการแบ่งกลุ่มการจัดขาออกเป็น 4 กลุ่มด้วยกัน คือ

- กลุ่มขาแหล่งจ่ายไฟเลี้ยง และสัญญาณนาฬิกา
- กลุ่มขาสำหรับการอ้างแอดเดรสและรับส่งข้อมูล
- กลุ่มขาที่ใช้ในการควบคุม
- กลุ่มขาพอร์ตใช้งานแบบขนานและอนุกรม



รูปที่ 7-5 แสดงไอซี 8051 ไมโครคอนโทรลเลอร์

ขาที่สำคัญของไมโครคอนโทรลเลอร์

1. ขา Vcc เป็นขารับแรงดันไฟกระแสตรง +5 Vdc
2. ขา GND เป็นขากราวนด์
3. พอร์ต 0 (P0.0-P0.7) เป็นพอร์ตอินพุทเอาท์พุทแบบ 2 ทิศทาง สำหรับใช้งานทั่วไป
4. พอร์ต 1 (P1.0-P1.7) เป็นพอร์ตอินพุทเอาท์พุทแบบ 2 ทิศทาง สำหรับใช้งานทั่วไป
5. พอร์ต 2 (P2.0-P2.7) เป็นพอร์ตอินพุทเอาท์พุทแบบ 2 ทิศทาง สำหรับใช้งานทั่วไป

6. พอร์ต 3 (P3.0-P3.7) เป็นพอร์ตอินพุทเอาท์พุทแบบ 2 ทิศทาง สำหรับใช้งานทั่วไป

7. ขารีเซต (RST) ใช้สำหรับการรีเซตการทำงานของไมโครคอนโทรลเลอร์ โดยการรีเซตจะต้องคงสภาวะ high อย่างน้อยนาน 2 Machine cycle ในขณะที่ออสซิลเลเตอร์กำลังทำงานอยู่

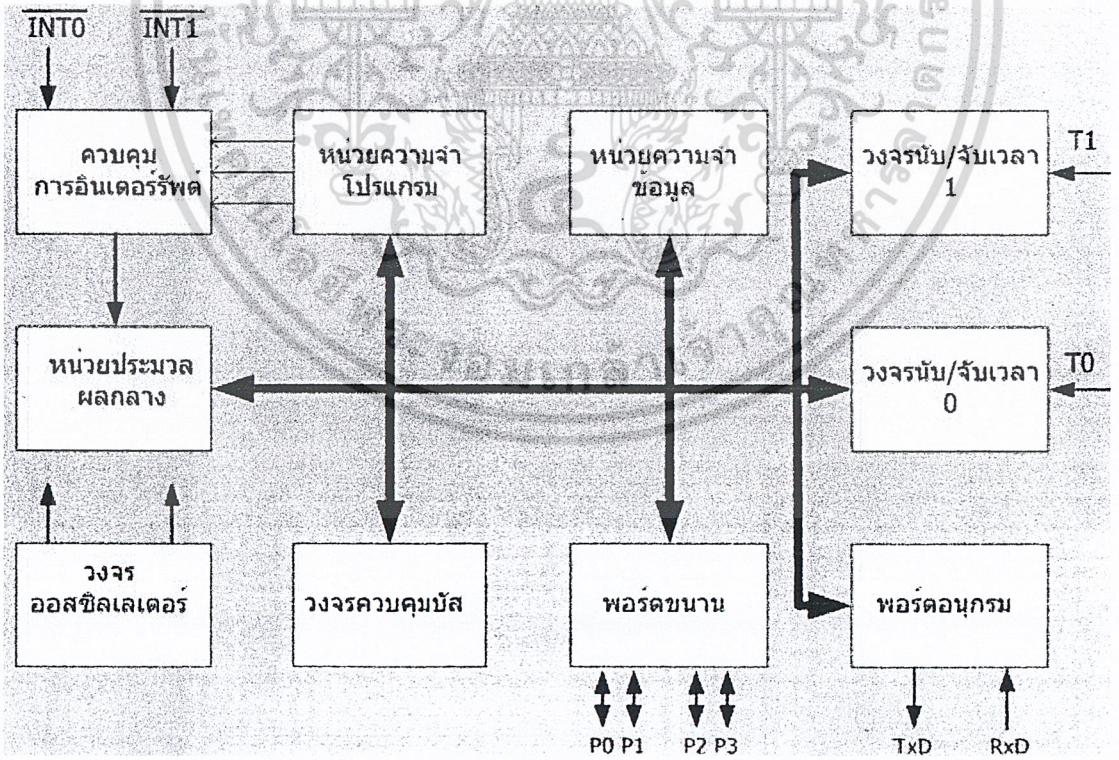
8. ขา ALE/PROG เป็นขาสัญญาณเพื่อทำหน้าที่ควบคุมการแล็ช (Latch) ค่าตำแหน่งแอดเดรสไบต์ต่ำ (Address Latch Enable)

9. ขา PSEN (Program Store Enable) ทำหน้าที่เป็นสัญญาณสโตปเพื่ออ่านคำสั่งจากหน่วยความจำภายนอก เมื่อไมโครคอนโทรลเลอร์ประมวลผลคำสั่ง จากหน่วยความจำภายนอก ขานี้จะส่งสัญญาณสโตปจำนวน 2 ครั้งในแต่ละ Machine cycle แต่ในขณะที่ติดต่อกับหน่วยความจำข้อมูลภายนอกจะไม่มีการส่ง สัญญาณสโตปแต่อย่างใด

10. ขา EA/Vcc (External Access Enable/Vcc) เป็นขาสำหรับการเลือกใช้หน่วยความจำโปรแกรมจากภายในหรือภายนอก โดยมีสถานะเป็น 0 และ 1 และขานี้ยังทำหน้าที่รับแรงดันไฟสำหรับ โปรแกรม (Vcc) ขนาด 12 โวลต์ เพื่อใช้ในระหว่างการโปรแกรมหน่วยความจำโปรแกรม (EPROM)

11. ขา XTAL1 และขา XTAL2 เป็นขาใช้งานของวงจรอินเวอร์ติ้งออสซิลเลเตอร์แอมปลิไฟเออร์ (Inverting Oscillator Amplifier) สำหรับใช้ต่อร่วมกับคริสตอลภายนอก

โครงสร้างภายในของไมโครคอนโทรลเลอร์ MCS-51

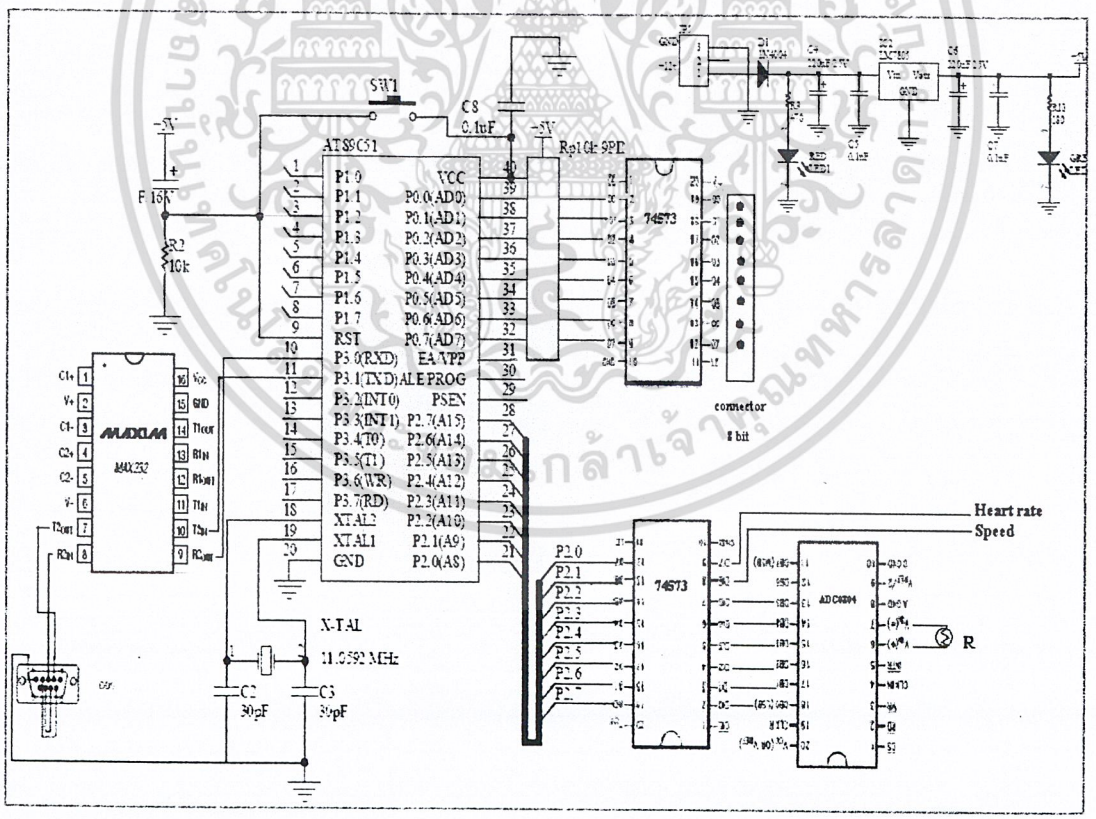


รูปที่ 7-6 แสดงโครงสร้างภายในของไมโครคอนโทรลเลอร์ MCS-51

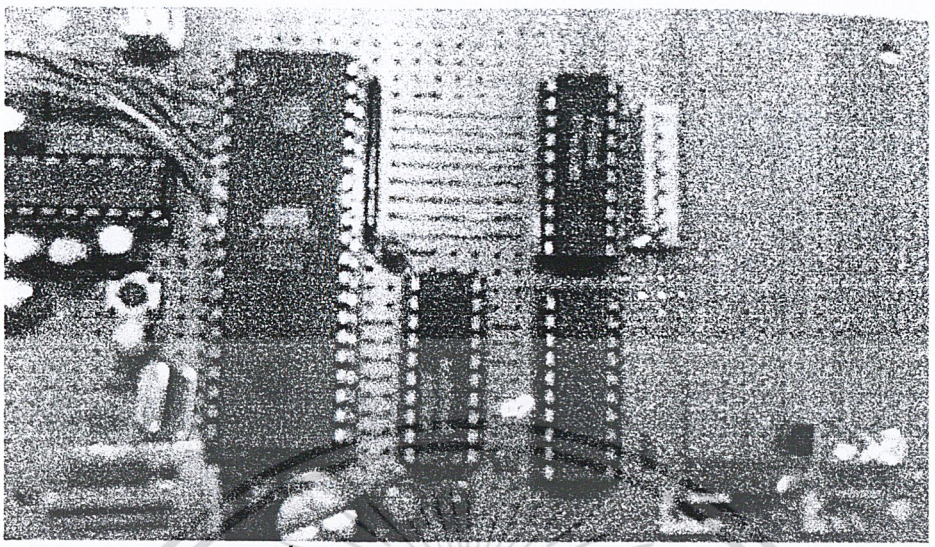
โครงสร้างภายในพื้นฐานของไมโครคอนโทรลเลอร์ตระกูล MCS-51 เบอร์ 8051 ประกอบด้วยอุปกรณ์ต่างๆดังนี้ ส่วนของหน่วยความจำภายในสำหรับเก็บข้อมูลขนาด 128 ไบท์ (Internal Data Memory) ส่วนของหน่วยความจำภายในสำหรับเก็บโปรแกรมที่มีขนาด 4 กิโลไบท์ (Internal Program Memory) อุปกรณ์ควบคุมการอินเตอร์รัพท์ (Interrupt Control Unit) ตัวตั้งเวลาและตัวนับเวลาขนาด 16 บิท 2 ชุด (Timer/Counter 0 and Timer/Counter 1) พอร์ตควบคุมการสื่อสารอนุกรมแบบ Full Duplex ซึ่งสามารถรับส่งข้อมูลพร้อมกันได้ พอร์ตขนานสำหรับติดต่อกับอุปกรณ์ภายนอกจำนวน 4 พอร์ต พอร์ตละ 8 บิท วงจรผลิตสัญญาณนาฬิกาภายใน

การทำงานของไมโครคอนโทรลเลอร์ในระบบ

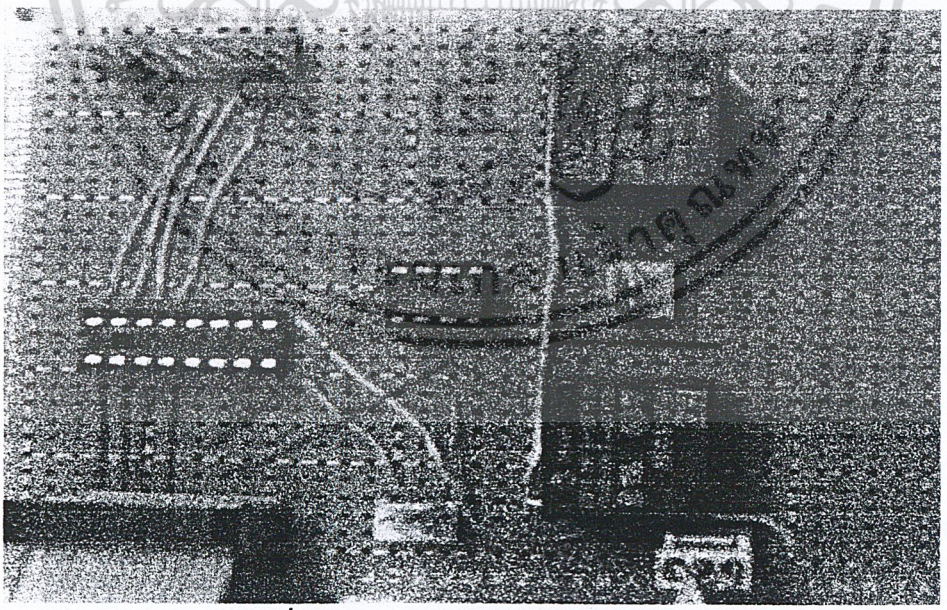
ไมโครคอนโทรลเลอร์ จะทำหน้าที่รับข้อมูลอนุกรมจากคอมพิวเตอร์เข้ามาทางขา 10 แล้วส่งออกมาทางพอร์ต 0 นำมาควบคุมมอเตอร์ยกจักรยาน, มอเตอร์สั่นจักรยาน และมอเตอร์หน่วงล้อจักรยาน และไมโครคอนโทรลเลอร์ จะรับข้อมูลจากเซ็นเซอร์ทางพอร์ต 2 แล้วส่งข้อมูลจากอินพุทที่เข้ามาไปยังคอมพิวเตอร์ โดยติดต่อผ่านทางพอร์ตอนุกรมมาตรฐาน RS-232 และได้กำหนดอัตราความเร็วในการรับ และส่งข้อมูลไว้ที่ 9600 บิตต่อวินาที (baud rate)



รูปที่ 7-7 แสดงวงจรหลักของระบบ



รูปที่ 7-8 แสดงรูปถ่ายวงจรหลักของระบบ

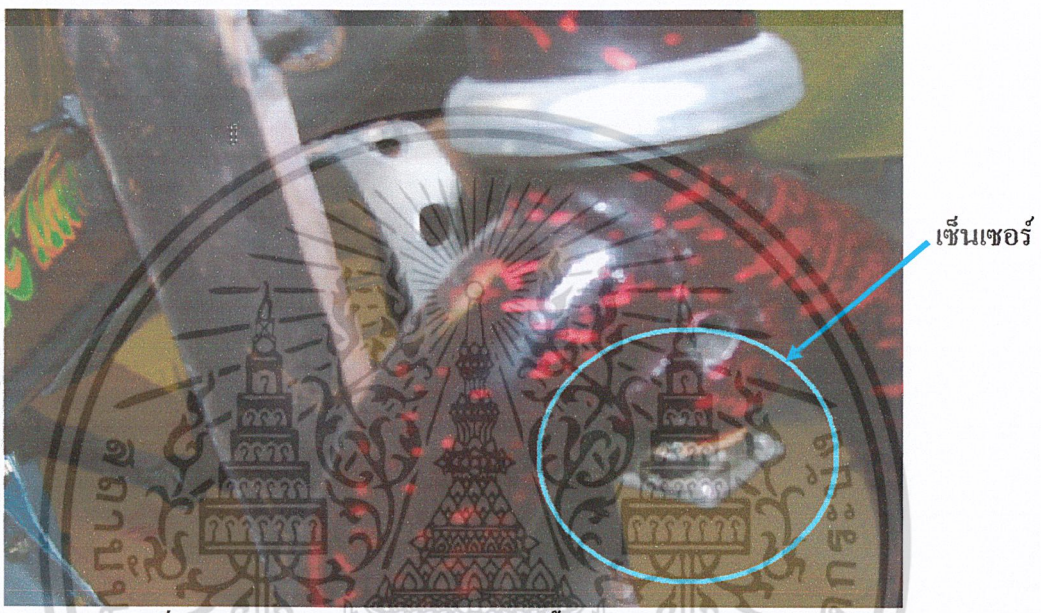


รูปที่ 7-9 แสดงรูปถ่ายวงจรควบคุมมอเตอร์

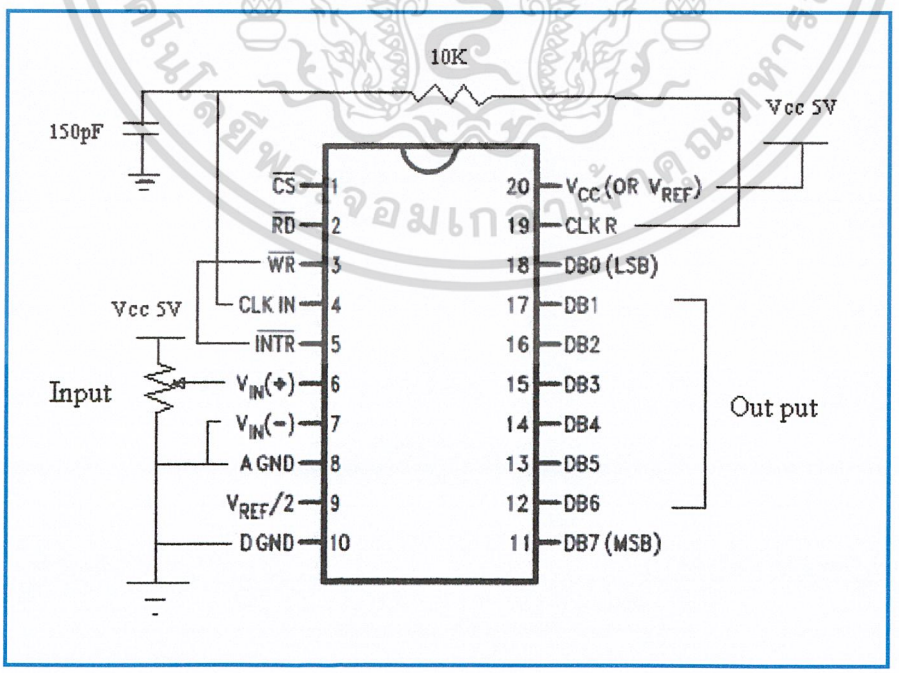
7.4 ส่วนตรวจสอบการควบคุมการทำงานจากรถจักรยาน

7.4.1 ส่วนตรวจสอบการเลี้ยวของแฮนด์จักรยาน

ใช้ความต้านทานปรับค่าได้ติดเข้ากับคอจักรยาน ดังรูปที่ 7-9 ซึ่งความต้านทานตัวนี้ต่อกับวงจรวัดความต่างศักย์ เมื่อแฮนด์จักรยานมีการหมุนจะทำให้ค่าความต้านทานจะเปลี่ยนแปลง ความต่างศักย์ก็จะเปลี่ยนแปลงด้วย IC แปลงสัญญาณอนาลอกเป็นสัญญาณดิจิตอล (Analog to Digital Converter) ก็จะแปลงความต่างศักย์เป็นสัญญาณ digital ขนาด 6 บิต ส่งเข้าไมโครคอนโทรลเลอร์



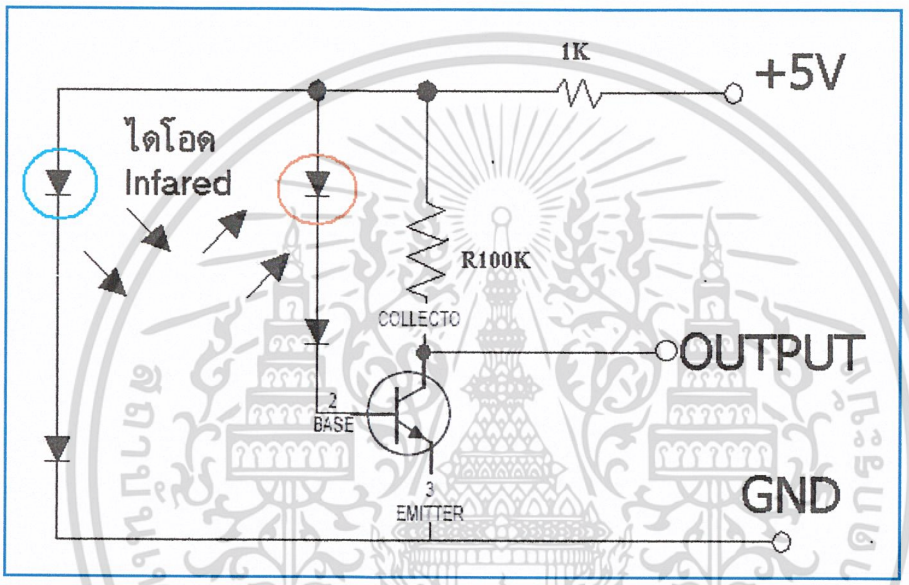
รูปที่ 7-10 แสดงเซ็นเซอร์ตรวจสอบการเลี้ยวของแฮนด์จักรยาน



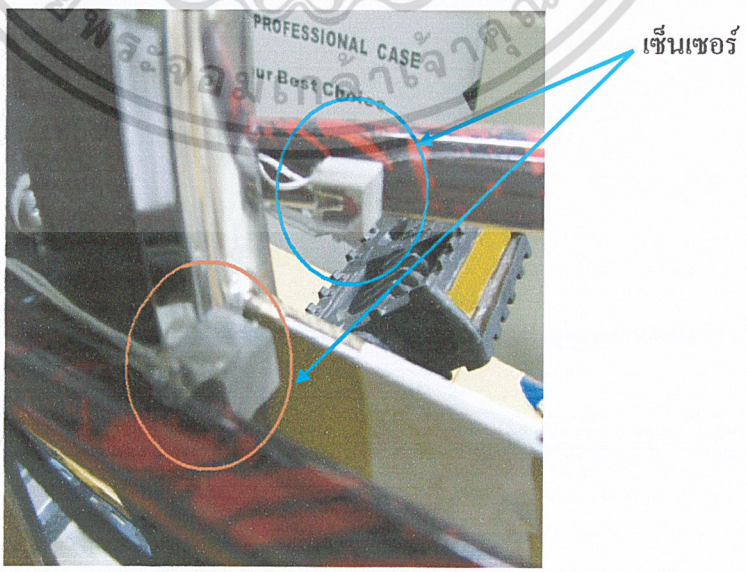
รูปที่ 7-11 แสดง ADC 0804 ซึ่งใช้แปลงค่าอนาลอกเป็นดิจิตอล

7.4.2 ส่วนตรวจสอบการหมุนของล้อหลังของจักรยาน

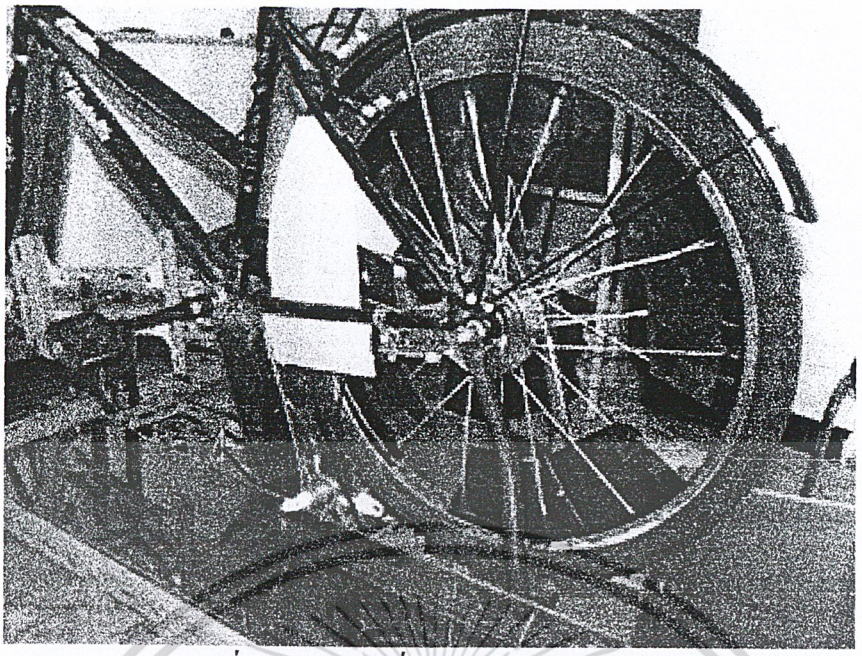
ในส่วนของการตรวจสอบการหมุนของล้อหลังของจักรยานใช้เพื่อวัดความเร็วและการเคลื่อนที่ของจักรยาน จะใช้วงจรที่ใช้โฟโต้ไดโอดจับความเร็วโดยวางเซ็นเซอร์ไว้ระหว่างล้อที่มีช่องที่มีความห่างเท่าๆกันดังรูป การทำงานของโฟโต้ไดโอดจะทำงานคล้ายกับไดโอดธรรมดาแต่จะแตกต่างที่โฟโต้ไดโอดจะยอมให้กระแสไฟฟ้าผ่านได้เมื่อมีแสงอินฟราเรดมาไบแอสเท่านั้น ปริมาณของกระแสที่โฟโต้ไดโอดยอมให้ผ่านขึ้นอยู่กับความเข้มของแสงด้วย ดังนั้น เมื่อล้อหมุนแผ่นกั้นช่องจะหมุนไปตัดลำแสงของโฟโต้ไดโอดทำให้ได้สัญญาณ ติด-ดับ ที่เปลี่ยนแปลง แล้วส่งเป็นข้อมูลขนาด 1 บิตไปยังไมโครคอนโทรลเลอร์



รูปที่ 7-12 แสดงวงจรจับความเร็วจักรยาน



รูปที่ 7-13 แสดงเซ็นเซอร์ตรวจสอบการหมุนของล้อหลังของจักรยาน



รูปที่ 7-14 แสดงล้อที่แบ่งออกเป็นช่องเท่าๆกัน

7.5 ส่วนสร้างปฏิกิริยาตอบสนอง

ส่วนสร้างปฏิกิริยาตอบสนองนั้นจะเป็นวงจรที่รับสัญญาณจากพอร์ต 0 ของไมโครคอนโทรลเลอร์ โดยมีการแบ่งความหมายของแต่ละบิตในการควบคุมอุปกรณ์ต่างๆ โดยจะแบ่ง 8 บิตดังตารางที่ 7-2

P0.7							P0.0
ลม	สั่น	ยก1	ยก0	เบรก3	เบรก2	เบรก1	เบรก0

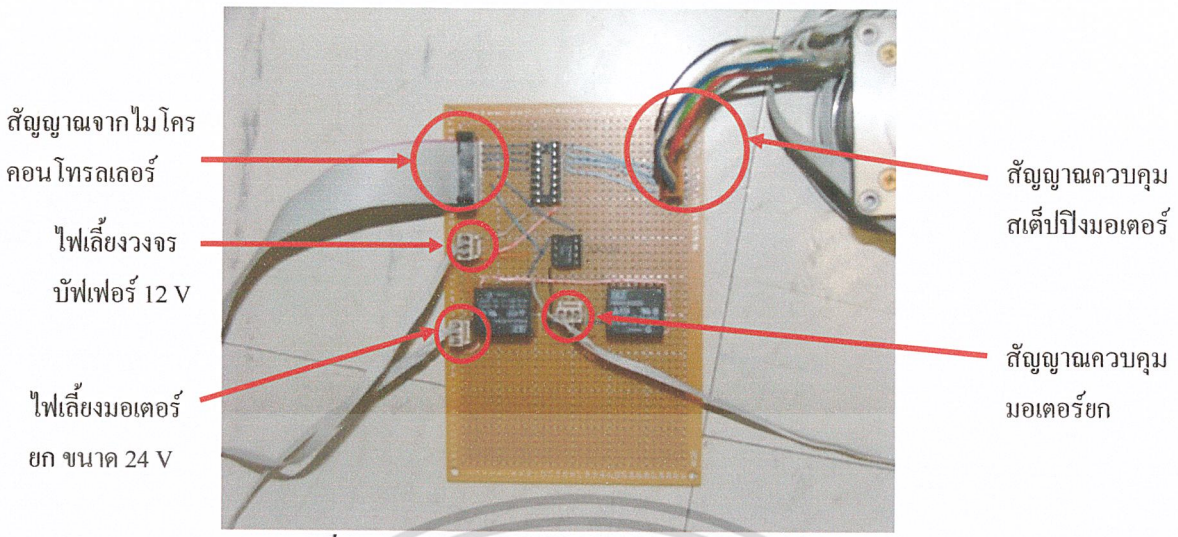
ตารางที่ 7-2 แสดงการแบ่งบิตข้อมูลของส่วนสร้างปฏิกิริยาตอบสนอง

7.5.1 ส่วนสร้างการหน่วงล้อ

ส่วนวงจรการหน่วงล้อนั้นมิได้ถูกออกแบบมาเพื่อใช้ในการตอบสนองการเบรก แต่เป็นการสร้างความฝืด หรือความรู้สึกหน่วงหนักเมื่อเวลาขี่จักรยานขึ้นทางชัน ทำให้จำเป็นต้องมีการติดตั้งเบรกทำงาน 2 รูปแบบ โดยส่วนใช้หน่วงล้อนี้จะเป็นตัวควบคุมเบรกแบบดรัมเบรก

ส่วนของอุปกรณ์ที่ใช้ในการหน่วงล้อคือ ใช้สเต็ปปิงมอเตอร์ต่อเข้ากับสายเบรกจักรยานที่ติดตั้งและแก้ไขบางส่วน

การควบคุมโดยใช้ 4 บิต ที่ออกมาจากไมโครคอนโทรลเลอร์ เพื่อเป็นตัวสร้างสัญญาณควบคุมให้เป็นรูปแบบต่างๆ เมื่อสเต็ปปิงมอเตอร์หมุนตามทิศทางที่กำหนดจะดึงสายเบรกทำให้เกิดการดึงเป็นระยะเวลาสั้นๆ และคลายออก โดยมีการเชื่อมต่อดังนี้



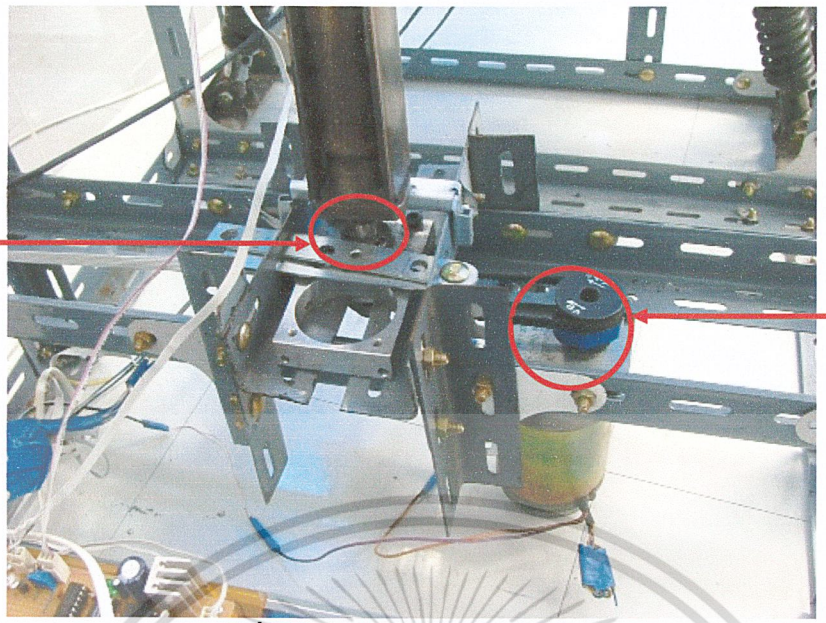
รูปที่ 7-15 แสดงรูปถ่ายวงจรควบคุมส่วนตอบสนอง

7.5.2 ส่วนสร้างระดับการยกองศาตัวจักรยาน

ส่วนการยกนั้นจะหลักการของคานในการคำนวณแรงที่ใช้ในการยก โดยการยกนั้นจะยกผ่านจุดหมุนที่คำนวณไว้และเหมาะสม แต่ในที่นี้จุดหมุนที่มีความเหมาะสมให้เลือกมีน้อยมาก ทำให้ต้องใช้การคิดหาจุดที่ใช้ยกที่เหมาะสมต่างหาก

อุปกรณ์ที่ใช้ในการยกจะมี มอเตอร์ที่มีขนาดใหญ่พอสมควรที่มีแรงมากพอ และแกนเหล็กเกลียวที่มีความฝืดต่ำเพื่อใช้เป็นกลไกในการยก โดยมีการติดตั้งดังรูป

การควบคุมโดยสั่งการมาจากไมโครคอนโทรลเลอร์ โดยใช่สัญญาณ 2 บิตโดยการทำงานเป็นแบบถ้า 2 บิตที่ส่งมาต่างกันจะเป็นการสั่งให้มีการยกในทิศทางที่กำหนด แต่ถ้า 2 บิตมีค่าเท่ากัน คือ 00 หรือ 11 จะเป็นการหยุดการทำงานของมอเตอร์ โดยสัญญาณที่ได้จากไมโครคอนโทรลเลอร์จะมีค่าน้อยกว่าที่จะเป็นเกิดจากการสูญหาย ทำให้ต้องมีวงจรที่ทำหน้าที่ช่วยเพิ่มความต่างศักดิ์เป็นวงจรออปแอมป์เพื่อรักษาระดับสัญญาณให้สามารถนำไปใช้ควบคุมรีเลย์ได้ ปัญหาที่สำคัญที่พบคือระดับสัญญาณที่ควบคุมกับระดับความต่างศักดิ์ที่ใช้งานมีความต่างกันมาก คือ ที่ไมโครคอนโทรลเลอร์จะมีขนาด 0-5 V และที่มอเตอร์ที่ทำงานที่ความต่างศักดิ์ขนาด 12 - 24 V ทำให้ต้องใช้รีเลย์เป็นตัวควบคุมอีกขั้นหนึ่ง

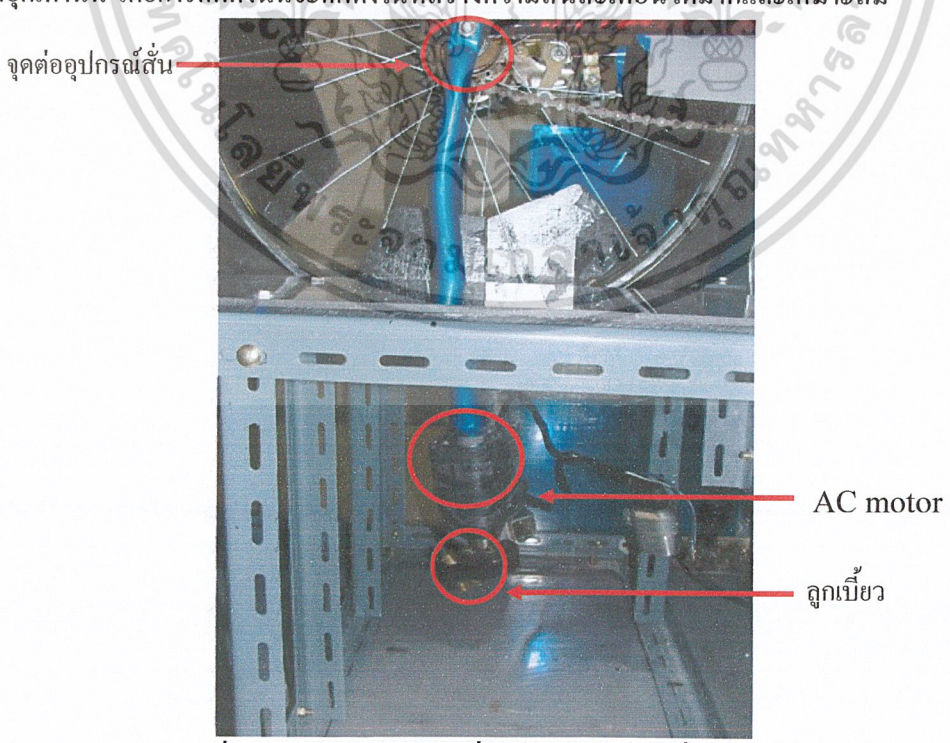


รูปที่ 7-16แสดงรูปอุปกรณ์ยกจักรยาน

7.5.3 ส่วนการสร้างแรงสั่นสะเทือน

วงจรการสร้างแรงสั่นสะเทือนจะใช้ในการตอบสนองเมื่อผู้ขี่จักรยานขี่จักรยานเข้าไปในพื้นที่ที่ขรุขระก็จะมีแรงสันสะเทื่อนเป็นจังหวะๆ

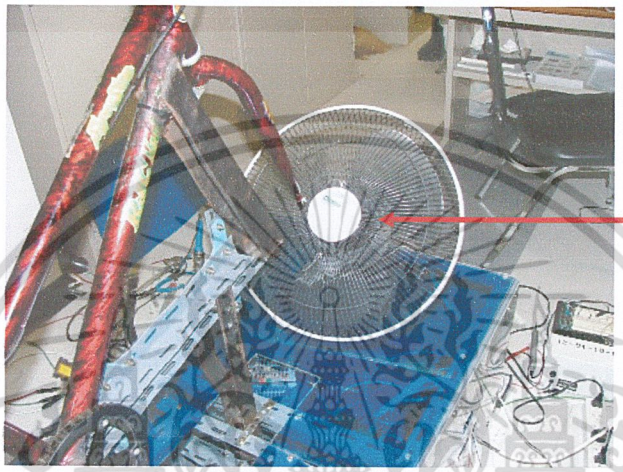
มอเตอร์ที่ใช้นั้นเป็นมอเตอร์กระแสสลับธรรมดาที่มีความเร็วรอบสูง และดัดแปลงโดยการทำเป็นลูกเบี้ยว โดยมอเตอร์ที่ใช้ในโครงงานนี้มีความเร็วสูงสุดที่ 1300 รอบต่อนาที และใช้ไฟฟ้าขนาด 220 โวลต์ ดังนั้นจึงต้องใช้รีเลย์ โดยการส่งสัญญาณควบคุมนั้นจะใช้เพียง 1 บิตเท่านั้น คือควบคุมให้ทำงานหรือหยุดเท่านั้น โดยการติดตั้งนั้นจะติดตั้งในที่สร้างความสั่นสะเทื่อนได้มากและเหมาะสม



รูปที่ 7-17 แสดงรูปมอเตอร์ที่ใช้ในการสร้างแรงสั่นสะเทือน

7.5.4 ส่วนการสร้างแรงลม

เป็นการสร้างความน่าสนใจยิ่งขึ้นเวลาใช้งานโดยได้นำเอามอเตอร์ที่ขนาดพอเหมาะมาติดตั้งเมื่อสร้างแรงลมเมื่อผู้ขี่ขี่จักรยานด้วยความเร็ว โดยจะดัดแปลงนำพัดลมมาต่อเข้ากับวงจรควบคุม สัญญาณควบคุมใช้เพียง 1 บิต และมีการใช้รีเลย์ซึ่งโดยรวมคล้ายกับการสร้างแรงสั่นสะเทือนมาก



รูปที่ 7-18 แสดงรูปมอเตอร์ที่ใช้ในการสร้างลม

บทที่ 8
การพัฒนาระบบทางด้านซอฟท์แวร์

สำหรับการพัฒนาระบบทางด้านซอฟท์แวร์นั้น จะแบ่งเป็น 2 ส่วน คือ

1) ส่วนสร้างภาพ 3 มิติ

2) ส่วนรับส่งข้อมูลกับจักรยาน

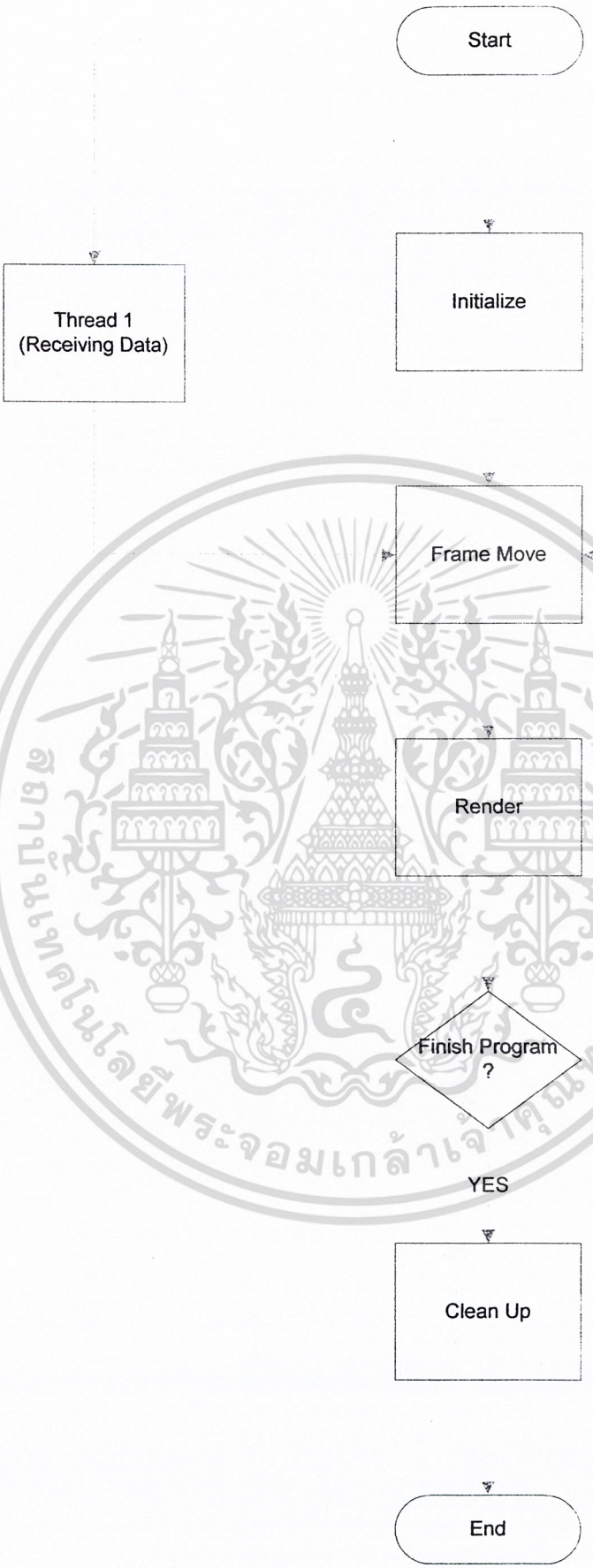
โดยแต่ละส่วนจะอยู่ในคอมพิวเตอร์คนละเครื่อง เครื่องหนึ่งใช้สำหรับติดต่อและรับส่งข้อมูลกับจักรยาน จากนั้นจะนำข้อมูลที่ได้รับมาประมวลผลข้อมูลเพื่อส่งให้คอมพิวเตอร์อีกเครื่อง รวมทั้งส่งฟีดแบ็คจาก (feedback) โปรแกรมสามมิติกลับไปหาจักรยาน และจะเรียกคอมพิวเตอร์เครื่องแรกนี้ว่า ฮาร์ดแวร์เซิร์ฟเวอร์ (Hardware Server)

คอมพิวเตอร์อีกเครื่องหนึ่งจะใช้ในการสร้างภาพ 3 มิติไปแสดงบนฉาก โดยคอมพิวเตอร์ทั้ง 2 เครื่องจะรับส่งข้อมูลระหว่างกันผ่านทางวินซอกส์ และจะเรียกคอมพิวเตอร์เครื่องที่ 2 นี้ว่า กราฟฟิกเซิร์ฟเวอร์ (Graphic Server)

8.1 ส่วนสร้างภาพ 3 มิติ

ในการพัฒนาซอฟท์แวร์ส่วนนี้ จะใช้โปรแกรมไมโครซอฟท์ วิชวล ซีพลัสพลัส 6.0 (Microsoft Visual C++ 6.0) โดยเป็นแอพพลิเคชันในรูปแบบของ Win32 และมีการเพิ่มโค้ดในส่วนของ DirectX 9.0 ที่ใช้สำหรับสร้างภาพ 3 มิติเข้าไปด้วย ซึ่งก่อนที่จะสามารถเรียกใช้ไลบรารีต่าง ๆ ของไดเร็กซ์เอ็กซ์ได้นั้น จะต้องลง DirectX SDK ก่อน

ขั้นตอนการทำงานของแอพพลิเคชันที่ใช้สำหรับสร้างภาพ 3 มิติ สามารถแสดงได้ด้วยโฟลวชาร์ต (flow chart) ในรูปที่ 8-1



รูปที่ 8-1 แสดงโฟลวชาร์ตของโปรแกรมสร้างภาพ 3 มิติ

อธิบาย

Start: เริ่มโปรแกรม

Initialize: ใช้ในการกำหนดค่าเริ่มต้นต่างๆ ดังนี้

- InitAudio

จะมีการกำหนดค่าของไฟล์เสียงเก็บไว้เพื่อที่จะนำไปใช้งานเสียง

```
m_pMusicManager->CreateSegmentFromResource(&m_pClashSound,
_T("CLASH"), _T("WAVE") );
```

ตารางที่ 8-1 แสดงการกำหนดไฟล์เสียงไปเก็บไว้ในตัวแปร

โค้ดในตารางที่ 8-1 เป็นการกำหนดเสียง CLASH.WAV ให้กับ m_pClashSound เพื่อที่จะนำไปใช้ในโปรแกรม

- InitDeviceObjects

1) ทำการจองพื้นที่ในหน่วยความจำไว้ให้กับวัตถุทั้งที่สร้างขึ้นเองภายในโปรแกรม และที่นำเข้ามาจากไฟล์ .x

ตัวอย่างของการสร้างวัตถุขึ้นเองภายในโปรแกรม Microsoft Visual C++ 6.0 แสดงในตารางที่ 8-2

```
m_bridge = new Bridge(m_pd3dDevice);
hr = m_bridge->Setup();
if( FAILED( hr ) )
return DXTRACE_ERR( "m_bridge->Setup", hr );
```

ตารางที่ 8-2 แสดงการสร้างวัตถุขึ้นเองภายในโปรแกรม

โค้ดในตารางที่ 8-2 จะเป็นการเรียกใช้ฟังก์ชัน Setup ของวัตถุที่ต้องการสร้าง โดยโครงสร้างภายในฟังก์ชัน Setup จะเป็นดังตารางที่ 8-3

```
m_pDevice->CreateVertexBuffer(
        32 * sizeof(Vertex),
        D3DUSAGE_WRITEONLY,
        FVF_VERTEX,
        D3DPOOL_MANAGED,
        &m_pVB,
        0);
Vertex* v;
m_pVB->Lock(0, 0, (void**)&v, 0);
v[0] =  Vertex(-30.0f, 0.0f,-20.0f,  0.0f, 1.0f,  0.0f, 0.0f, 0.0f);
v[1] =  Vertex(-30.0f, 0.0f,  0.0f,  0.0f, 1.0f,  0.0f, 0.0f, 1.0f);
v[2] =  Vertex(-50.0f, 3.0f,-20.0f,  0.0f, 1.0f,  0.0f, 1.0f, 1.0f);
m_pVB->Unlock();
```

ตารางที่ 8-3 แสดงการสร้างเวอร์เท็กซ์บัฟเฟอร์

โค้ดในตารางที่ 8-3 จะใช้ในการกำหนดจุดยอดต่างๆ แล้วนำค่าไปเก็บไว้ที่ m_pVB ซึ่งจะใช้เป็นบัฟเฟอร์สำหรับเก็บจุดต่าง ๆ เหล่านั้นไว้ โดยจะเรียกจุดยอดนี้ว่า เวอร์เท็กซ์ โดยจะสามารถระบุรายละเอียดของเวอร์เท็กซ์ได้ เช่น กำหนดตำแหน่งของเวอร์เท็กซ์ในเวิร์ล กำหนดเส้นปกติของเวอร์เท็กซ์ กำหนดค่าของแกน u และค่าของแกน v ที่ใช้ในการทำ texturing เป็นต้น

```
m_pDevice->CreateIndexBuffer(
        54 * sizeof(WORD),
        D3DUSAGE_WRITEONLY,
        D3DFMT_INDEX16,
        D3DPOOL_MANAGED,
        &m_pIB,
        0);
WORD* i = 0;
m_pIB->Lock(0, 0, (void**)&i, 0);
i[0] = 0; i[1] = 2; i[2] = 1;
i[3] = 1; i[4] = 2; i[5] = 3;
i[6] = 4; i[7] = 6; i[8] = 5;
m_pIB->Unlock();
```

ตารางที่ 8-4 แสดงการสร้างอินเด็กซ์บัฟเฟอร์

หลังจากที่ได้กำหนดจุดเวอร์เท็กซ์แล้ว ก็จะต้องกำหนดอินเด็กซ์ที่จะชี้ไปยังเวอร์เท็กซ์เหล่านั้น เพื่อกำหนดรูปร่างลักษณะของวัตถุที่จะวาด โดยค่าอินเด็กซ์ต่าง ๆ จะถูกเก็บไว้ในบัฟเฟอร์ที่มีชื่อว่า m_pIB หลังจากนั้นจึงนำค่าเวอร์เท็กซ์ที่ถูกชี้ด้วยข้อมูลในอินเด็กซ์บัฟเฟอร์ไปวาดเป็นรูปทรง 3 มิติต่าง ๆ ในขั้นตอนของการ Render

```
D3DXCreateTextureFromFile(
        m_pDevice,
        "cityroad.jpg",
        &m_pTex);
```

ตารางที่ 8-5 แสดงการนำข้อมูลของ texture เข้ามาใช้ในโปรแกรม

โค้ดในตารางที่ 8-5 เป็นการนำข้อมูลของ texture "cityroad.jpg" เข้ามาเก็บไว้ที่ m_pTex เพื่อที่จะนำไปแปะติดกับวัตถุที่สร้างขึ้นในขั้นตอนก่อนหน้านี้

ตัวอย่างการนำวัตถุเข้ามาใช้จากไฟล์ .x แสดงในตารางที่ 8-6

```
m_bicycle = new Bicycle(m_pd3dDevice);
hr = m_bicycle->Setup();
if( FAILED( hr ) )
return DXTRACE_ERR( "m_bicycle->Setup", hr );
```

ตารางที่ 8-6 แสดงนำวัตถุเข้ามาใช้ในโปรแกรมจากไฟล์ .x

โค้ดในตารางที่ 8-6 จะเป็นการเรียกใช้ฟังก์ชัน Setup ของวัตถุที่ต้องการนำเข้ามาใช้งานในโปรแกรมจากไฟล์ .x โดยโครงสร้างภายในฟังก์ชัน Setup จะเป็นดังตารางที่ 8-7

```
D3DXLoadMeshFromX(
        "Scooter.x",
        D3DXMESH_MANAGED,
        m_pDevice,
        &adjBuffer,
        &mtrlBuffer,
        0,
        &numMtrls,
        &m_pSourceMesh);
```

ตารางที่ 8-7 แสดงการนำข้อมูลของวัตถุมาจากไฟล์ .x

โค้ดในตารางที่ 8-7 จะเป็นตัวอย่างการนำเข้าข้อมูลของวัตถุมาจากไฟล์ "Scooter.x" โดยค่า material และ texture ของวัตถุที่อยู่ในไฟล์ .x จะถูกเก็บอยู่ในตัวแปร mtrlBuffer และจำนวนของส่วนประกอบ (subset) ต่าง ๆ ของวัตถุหรือจำนวน material จะอยู่ในตัวแปร numMtrls ซึ่งสามารถนำค่าเหล่านี้ไปใช้ในการวาดภาพในขั้นตอนของการ Render ได้

```
m_pSourceMesh->OptimizeInplace(
        D3DXMESHOPT_ATTRSORT |
        D3DXMESHOPT_COMPACT |
        D3DXMESHOPT_VERTEXCACHE,
        (DWORD*)adjBuffer->GetBufferPointer(),
        (DWORD*)adjBuffer->GetBufferPointer(),
        0, 0);
```

ตารางที่ 8-8 แสดงการใช้งานฟังก์ชัน OptimizeInplace

สำหรับโค้ดในตารางที่ 8-8 นี้จะเป็นการใช้งานฟังก์ชัน OptimizeInplace ซึ่งจะใช้เพื่อปรับปรุงคุณภาพของรูปที่นำเข้ามาจากไฟล์ .x ให้ดีขึ้น

2) ทำการเก็บค่าของ texture ต่าง ๆ เพื่อเตรียมที่จะนำใช้ในขั้นตอนของการ Render ภาพ 2 มิติ

```
hr=D3DXCreateTextureFromFile(m_pd3dDevice,"green.jpg",&m_pSpeedG );
if( FAILED( hr ) )
return DXTRACE_ERR( "m_pSpeedG->InitDeviceObjects", hr );
```

ตารางที่ 8-9 แสดงการสร้าง texture เพื่อนำมาใช้ในโปรแกรม

โค้ดในตารางที่ 8-9 นี้เป็นการสร้าง texture แล้วนำไปเก็บไว้ที่ m_pSpeedG

- RestoreDeviceObjects

เป็นขั้นตอนของการกำหนดค่าเริ่มต้นต่าง ๆ ให้กับแอพพลิเคชัน แต่จะเรียกใช้หลังจากทำฟังก์ชัน InitDeviceObjects เสร็จแล้ว ซึ่งมีการกำหนดค่าต่าง ๆ ดังนี้

1) ทำการกำหนดค่าเริ่มต้นต่าง ๆ ให้กับกล้อง

```
D3DXVECTOR3 vEyePt = m_camera->GetEyePt();
D3DXVECTOR3 vLookatPt = m_camera->GetLookatPt();
D3DXVECTOR3 vUpVec = m_camera->GetUpVec();
vEyePt[0] =  0;
vEyePt[1] =  5;
vEyePt[2] =  -20;
m_vRightVec[0] = 1.0;
m_vRightVec[1] = 0.0;
m_vRightVec[2] = 0.0;
vLookatPt[1] = -0.35f;
m_camera->SetViewParams(vEyePt, vLookatPt, vUpVec);
D3DXMatrixLookAtLH( &matView, &vEyePt, &vLookatPt, &vUpVec );
m_pd3dDevice->SetTransform( D3DTS_VIEW, &matView );
```

ตารางที่ 8-10 แสดงการกำหนดค่าเริ่มต้นให้กับกล้อง

โค้ดในตารางที่ 8-10 เป็นการตั้งค่าเริ่มต้นให้กับกล้อง โดยจะกำหนดค่าตำแหน่งกล้องเป็น (0,5,-20) ตามค่าของเวกเตอร์ vEyePt กำหนดค่าเวกเตอร์ที่ระบุทิศขวาของกล้องเป็น (1,0,0) ตามค่าของ m_vRightVec และกำหนดค่าตำแหน่งที่กล้องมองเป็น (ค่าเดิม,-0.35,ค่าเดิม) หลังจากนั้นก็เรียกใช้ฟังก์ชัน D3DXMatrixLookAtLH(&matView,...) เพื่อทำการสร้างเมทริกซ์ matView และจะต้องเรียกใช้ฟังก์ชัน SetTransform(D3DTS_VIEW, &matView) เพื่อกำหนดให้เมทริกซ์ matView เป็นเมทริกซ์ที่ใช้ที่ใช้สำหรับการเปลี่ยนจากระบบแกนโคออร์ดิเนตของเวิร์ลมาเป็นระบบแกนโคออร์ดิเนตของกล้อง

2) ทำการกำหนดค่าเริ่มต้นต่าง ๆ ที่ใช้ในการจัดการเรื่องแสง

```
D3DLIGHT9 light;
D3DUtil_InitLight( light, D3DLIGHT_SPOT, -35.0f, 10.0f,-20.0f );
light.Theta = 3.1416/4;
light.Phi = 3.1416/2;
 m_pd3dDevice->SetLight( 1, &light );
 m_pd3dDevice->LightEnable( 1, true);
```

ตารางที่ 8-11 แสดงการกำหนดค่าเริ่มต้นให้กับการจัดการเรื่องแสง

สำหรับโค้ดในตารางที่ 8-11 จะใช้สำหรับการสร้างแหล่งกำเนิดแสงแบบสปอตไลท์ โดยให้แหล่งกำเนิดแสงนี้อยู่ในตำแหน่ง (-35, 10, -20) ของระบบแกนโคออร์ดิเนตของเวิร์ล รวมทั้งกำหนดค่ามุม θ และค่ามุม φ ให้กับแหล่งกำเนิดแสงดวงนี้ด้วย

Frame Move : เป็นส่วนของการนำค่าต่าง ๆ มาใช้คำนวณเพื่อใช้ในการเปลี่ยนตำแหน่งของกล้องและวัตถุอื่น ๆ ดังนี้

1) ความเร็ว

```
vEyePt[0] += m_fElapsedTime * m_speed * vLookatPt[0];
vEyePt[2] += m_fElapsedTime * m_speed * vLookatPt[2];
```

ตารางที่ 8-12 แสดงส่วนของการคำนวณความเร็วให้เป็นการเคลื่อนที่ในโปรแกรม 3 มิติ

โค้ดในตารางที่ 8-12 เป็นการนำค่าของความเร็วมาคูณกับทิศทางแล้วนำไปบวกเข้ากับค่าเดิมเพื่อเปลี่ยนตำแหน่งของกล้อง โดยตัวอย่างจะคิดความเร็วเฉพาะในแนวแกน x และ z เพราะเป็นการเคลื่อนที่ในแนวระนาบ

2) องศาการเลี้ยว

```
m_fWorldRotY = m_fElapsedTime *D3DXToRadian(m_Rotangel);
D3DXMatrixRotationY(&T, m_fWorldRotY);
D3DXVec3TransformCoord(&m_vRightVec, &m_vRightVec, &T);
D3DXVec3TransformCoord(&vLookatPt, &vLookatPt, &T);
```

ตารางที่ 8-13 แสดงการนำค่าองศาการเลี้ยวไปเปลี่ยนทิศทางการเคลื่อนที่ของจักรยาน

โค้ดในตารางที่ 8-13 เป็นการนำค่าของมุมมาเปลี่ยนทิศทางการเคลื่อนที่ของกล้องตามมุม m_Rotangel

Render : เป็นขั้นตอนของการแสดงภาพต่าง ๆ โดยจะนำภาพที่เราได้สร้างขึ้นมาจากขั้นตอนก่อนหน้านี้ไปแสดงในเวิร์ลที่เราได้เตรียมไว้ให้ เพื่อจะได้เห็นภาพออกมาที่หน้าจอ

1) การวาดภาพ 3 มิติ

```
m_ground->Draw()
```

ตารางที่ 8-14 แสดงการใช้ฟังก์ชัน Draw ในการวาดภาพ 3 มิติ

โค้ดในตารางที่ 8-14 จะเป็นการเรียกใช้ฟังก์ชัน Draw ของวัตถุ m_ground เพื่อที่จะวาดภาพที่เราเตรียมไว้ในหน่วยความจำตอน Initialize โดยการวาดวัตถุนั้นจะมีอยู่ 2 แบบ คือ การวาดวัตถุที่สร้างขึ้นภายในโปรแกรมโดยค่าเวอร์เท็กซ์และอินเด็กเอง กับการวาดวัตถุที่นำเข้ามาจากไฟล์ .x

- ตัวอย่างการวาดภาพวัตถุที่สร้างขึ้นภายในโปรแกรม

สำหรับรายละเอียดภายในฟังก์ชัน Draw ของวัตถุที่สร้างขึ้นภายในโปรแกรมนั้น สามารถแสดงได้ตามโค้ดในตารางที่ 8-15

```
m_pDevice->SetTransform(D3DTS_WORLD, &m_world);
m_pDevice->SetMaterial(&m_mtrl);
m_pDevice->SetTexture(0, m_pTex);
m_pDevice->SetSamplerState(0,D3DSAMP_ADDRESSU, D3DTADDRESS_WRAP);
m_pDevice->SetSamplerState(0,D3DSAMP_ADDRESSV, D3DTADDRESS_WRAP);
m_pDevice->SetStreamSource(0, m_pVB, 0, sizeof(Vertex));
m_pDevice->SetIndices(m_pIB);
m_pDevice->SetFVF(FVF_VERTEX);
m_pDevice->DrawIndexedPrimitive(
        D3DPT_TRIANGLELIST,
        0,
        0,
        32,
        0,
        18);
```

ตารางที่ 8-15 แสดงวิธีการวาดภาพวัตถุที่สร้างขึ้นภายในโปรแกรม

ขั้นแรกก่อนการวาดภาพจะต้องทำการแปลง (transform) ค่าตำแหน่งของวัตถุให้ไปอยู่ในจุดที่ต้องการก่อน หลังจากนั้นจึงทำการกำหนด material และ texture ให้กับวัตถุนั้น แล้วจึงทำการเตรียมข้อมูลก่อนทำการวาด เช่น กำหนดรูปแบบของการวาด (SetSamplerState), กำหนดบัฟเฟอร์ของเวอร์เท็กซ์และอินเด็กซ์ที่จะใช้วาด รวมทั้งกำหนดรูปแบบของเวอร์เท็กซ์ที่จะทำการวาด จนถึงขั้นตอนสุดท้ายจึงทำการเรียกใช้ฟังก์ชัน DrawIndexedPrimitive เพื่อทำการวาดภาพนั้นออกไป

- ตัวอย่างการวาดภาพวัตถุที่นำเข้ามาจากไฟล์ .x

รายละเอียดของการวาดวัตถุที่นำเข้ามาจากไฟล์ .x ถูกแสดงตามโค้ดในตารางที่ 8-16

```
D3DXMATRIX      m_tran;
D3DXMATRIX      m_scale;
D3DXMATRIX      m_rotY;
D3DXMatrixTranslation(&m_tran, m_posX, m_posY, m_posZ);
D3DXMatrixScaling(&m_scale,0.035f,0.035f,0.035f);
D3DXMatrixRotationY(&m_rotY,D3DXToRadian(m_angleY));
D3DXMatrixMultiply(&m_world,&m_scale,&m_rotY);
D3DXMatrixMultiply(&m_world,&m_world,&m_tran);
m_pDevice->SetTransform(D3DTS_WORLD, &m_world);
for(int i = 0; i < int(m_mtrls.size()); i++)
{
        m_pDevice->SetMaterial( &m_mtrls[i] );
        m_pDevice->SetTexture(0, m_pTextures[i]);
        m_pSourceMesh->DrawSubset(i);
}
```

ตารางที่ 8-16 แสดงวิธีการวาดภาพวัตถุที่นำเข้ามาจากไฟล์ .x

ก่อนการวาดภาพวัตถุที่นำเข้ามาจากไฟล์ .x นั้น จะต้องมีการเลื่อนตำแหน่ง (translate), ย่อหรือขยายขนาด (scaling) และหมุนวัตถุ (rotation) ให้มีขนาด และอยู่ในตำแหน่งและทิศทางที่ต้องการเสียก่อน หลังจากนั้นจึงทำการแปลง (transform) วัตถุให้ไปอยู่ในตำแหน่งที่เหมาะสมในระบบแกนโคออร์ดิเนตของเวิร์ล สุดท้ายจึงทำการกำหนด material และ texture ขององค์ประกอบ (subset) ของวัตถุที่ต้องการทีละองค์ประกอบ พร้อมทั้งวาดองค์ประกอบนั้นออกไปทีละชิ้นจนครบทุกองค์ประกอบของวัตถุที่ต้องการวาด

2) การวาดภาพ 2 มิติ

```
D3DXMATRIX scale, trans, final, rotate;
RECT m_rect;
m_rect.left = 0;
m_rect.top = 0;
m_rect.right = 1020;
m_rect.bottom = 250;
   D3DXMatrixScaling( &scale, 1.0f, 0.65f, 0.0f );
D3DXMatrixTranslation( &trans, 0, 530, 0 );
D3DXMatrixMultiply( &final, &scale, &trans);
m_pSprite->SetTransform( &final );
if(FAILED(m_pSprite->Draw(
        m_pConsole,
        &m_rect,
        0,
        0,
        0xFFFFFFFF)))
{
        return FALSE;
}
```

ตารางที่ 8-17 แสดงวิธีการวาดภาพ 2 มิติ

โค้ดในตารางที่ 8-17 เป็นการนำ texture ที่เราเตรียมจะ Render ภาพ 2 มิติมาวาด โดยเราจะกำหนดขนาดให้กับพื้นที่ที่จะใช้วาด ดังตารางที่ 8-18

```
m_rect.left = 0;
m_rect.top = 0;
m_rect.right = 1020;
m_rect.bottom = 250;
```

ตารางที่ 8-18 แสดงการกำหนดขนาดพื้นที่ที่จะใช้วาดภาพ 2 มิติ

หลังจากนั้น จะกำหนดตำแหน่งที่ต้องการวาดภาพ 2 มิติภายในพื้นที่ที่กำหนดไว้ และทำการเลื่อนตำแหน่งของรูปที่ต้องการวาดไปไว้ในที่ ๆ ต้องการ สุดท้ายจึงใช้คำสั่ง Draw เพื่อวาดลงไป

Clean Up : เป็นการลบค่าต่าง ๆ ที่เก็บลงไว้ในหน่วยความจำก่อนที่จบโปรแกรม ดังในตารางที่ 8-19

```
m_ground->Cleanup();
```

ตารางที่ 8-19 แสดงการใช้ฟังก์ชัน Cleanup

โค้ดด้านบนเป็นตัวอย่างการเรียกใช้ฟังก์ชัน Cleanup ของวัตถุ m_ground ซึ่งภายในฟังก์ชันนี้จะทำการเรียกใช้ฟังก์ชัน Release เพื่อทำการยกเลิกการหน่วยความจำให้กับ COM ออบเจ็กท์ที่ใช้ในการสร้างวัตถุ m_ground

End : จบการทำงานของโปรแกรม

8.2 ส่วนรับส่งข้อมูลกับจักรยาน

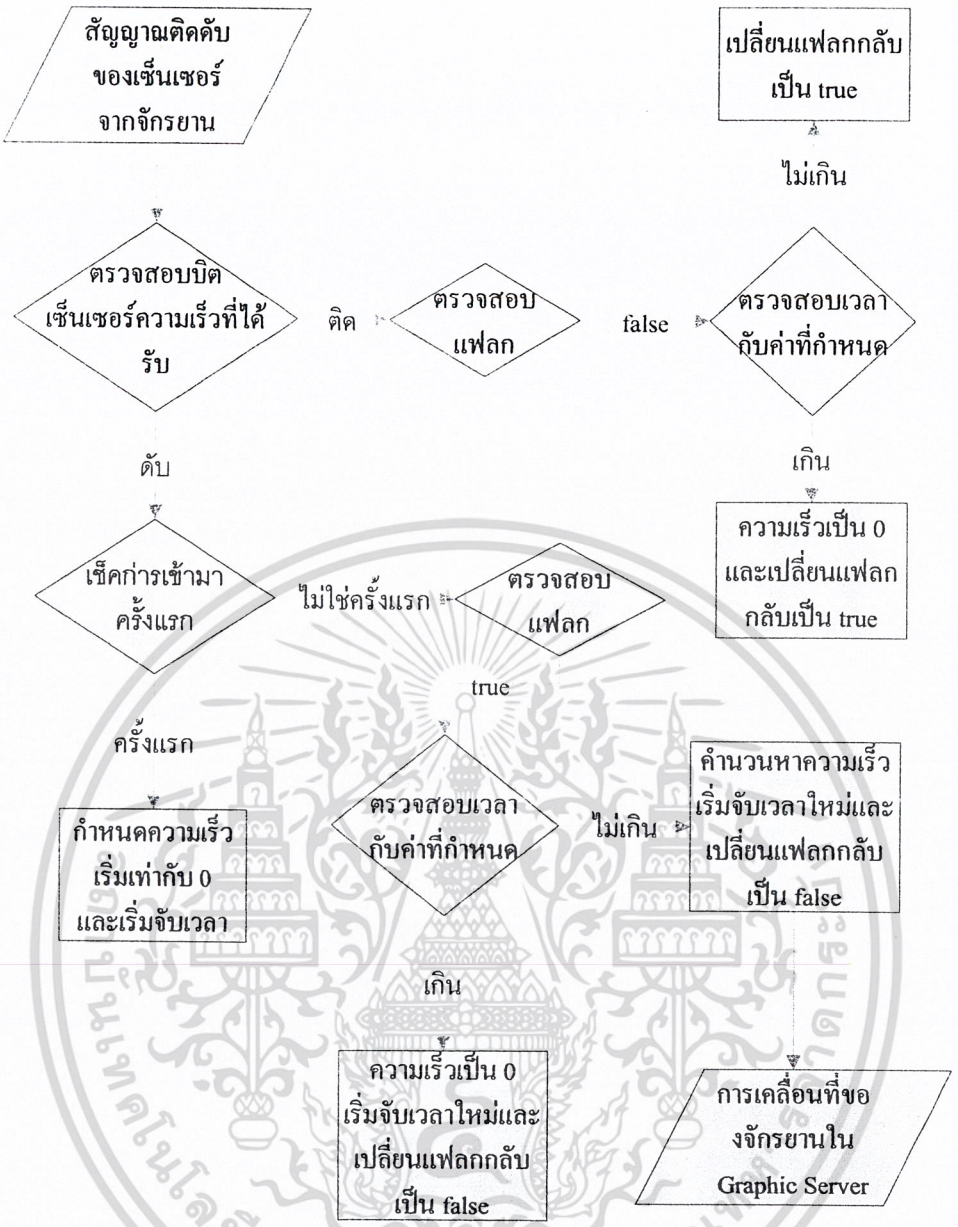
สำหรับการพัฒนาซอฟท์แวร์ในส่วนนี้ จะใช้โปรแกรมไมโครซอฟท์ วิชวล ซีพลัสพลัส 6.0 (Microsoft Visual C++ 6.0) โดยจะเป็นแอพพลิเคชันในรูปแบบของ MFC (Microsoft Foundation Class) ชนิดไดอะล็อกเบส ซึ่งจะแบ่งการพัฒนาออกเป็นส่วนสำคัญต่าง ๆ 6 ส่วนดังนี้

1) ส่วนคำนวณความเร็ว
2) ส่วนคำนวณองศาของการเลี้ยว
3) ส่วนคำนวณแคลอรี
4) ส่วนส่ง feedback กลับไปให้จักรยาน
5) ส่วนเชื่อมต่อกับพอร์ตอนุกรม (Serial port)
6) ส่วนเชื่อมต่อกับ กราฟฟิกเซิร์ฟเวอร์

โดยรายละเอียดการพัฒนาส่วนต่าง ๆ มีดังนี้

1) ส่วนคำนวณความเร็ว

ส่วนนี้จะเป็นส่วนที่รับค่าสัญญาณติด-ดับของบิตที่ 7 มาจากพอร์ตอนุกรม ซึ่งสัญญาณติด-ดับนี้จะเป็นสัญญาณของโฟโต้ไดโอดที่ติดอยู่กับล้อจักรยาน ซึ่งจะใช้วิธีการจับเวลาในช่วงที่สัญญาณเริ่มดับไปจนติดแล้วกลับมาดับอีกครั้ง จากนั้นจึงนำช่วงเวลาที่ได้ไปหาระยะทางซึ่งก็คือส่วนของล้อจักรยานที่มีความยาวตั้งแต่ตำแหน่งที่กั้นแสงของไดโอดตำแหน่งแรกไปจนถึงตำแหน่งที่กั้นแสงของไดโอดตำแหน่งต่อไป โดยโฟลวชาร์ตที่แสดงการคำนวณความเร็วสามารถแสดงได้ตามรูปที่ 8-2



รูปที่ 8-2 แสดงโฟลวชาร์ตที่แสดงการคำนวณความเร็ว

อธิบายอัลกอริทึมของส่วนคำนวณความเร็ว

```
speedPulse = sRxBuf[0] & 0x80;
speedPulse = speedPulse >> 7;
```

ตารางที่ 8-20 แสดงการรับข้อมูลที่ใช้คำนวณความเร็วมาจากพอร์ตอนุกรม

โค้ดในตารางที่ 8-20 นี้ เป็นส่วนที่รับข้อมูลมาจากพอร์ตอนุกรม แล้วนำมา AND กับ 10000000b หลังจากนั้นจึงทำการเลื่อนบิตข้อมูลไปทางขวา 7 บิต ทั้งนี้เพื่อให้เหลือแต่ค่าของบิตที่ 7 ซึ่งก็คือค่าสัญญาณติด-ดับของโฟโต้ไดโอดนั่นเอง

```
if (speedPulse == 0x01)                    //off
{
        if (firsttime == false)
        {
                if (chk)
                {
                        _ftime(&timebuffer);
                        timeperiod = timebuffer.time - m_speedTimer.time;
                        if (timeperiod > 1)
                                m_speed = 0;
                        else
                        {
                                milliperiod =timebuffer.millitm- m_speedTimer.millitm;
                                if (milliperiod < 0)
                                {
                                        timeperiod--;
                                        milliperiod += 1000;
                                }
                                m_speed=30*10/double(timeperiod*1000 +milliperiod);
                                if (m_speed > 5)
                                        m_speed = 5;
                        }
                        m_speedTimer = timebuffer;
                        chk=false;
                }
                else
                {
                        _ftime(&timebuffer);
                        timeperiod = timebuffer.time - m_speedTimer.time;
                        if (timeperiod > 1)
                                m_speed = 0;
                }
        }
        else
        {
                _ftime(&timebuffer);
                m_speedTimer=timebuffer;
                firsttime = false;
                chk = false;
        }
}
```

ตารางที่ 8-21 แสดงอัลกอริทึมที่ใช้ในการคำนวณความเร็ว

```
else
{
        if (firsttime == false)
        {
                if (!chk)
                {
                        _ftime(&timebuffer);
                        timeperiod = timebuffer.time - m_speedTimer.time;
                        if (timeperiod > 1)
                        {
                                m_speed = 0;
                        }
                        chk = true;
                }
                else
                {
                        _ftime(&timebuffer);
                        timeperiod = timebuffer.time - m_speedTimer.time;
                        if (timeperiod > 1)
                        {
                                m_speed = 0;
                        }
                }
        }
        else
        {
                //do nothing
        }
}
```

ตารางที่ 8-22 แสดงอัลกอริทึมที่ใช้ในการคำนวณความเร็ว

โค้ดในตารางที่ 8-21 และตารางที่ 8-22 คือ อัลกอริทึมที่ใช้ในการคำนวณความเร็ว โดยจะใช้ตัวแปรบูลลีน chk เป็นตัวตรวจสอบว่าได้ผ่านสัญญาณดับมายังสัญญาณติดแล้วกลับไปสัญญาณดับครบ 1 รอบของการคำนวณแล้วหรือยัง ซึ่งถ้าครบ 1 รอบแล้ว ก็จะนำเวลาที่ได้จากการจับ m_speedTimer มาคำนวณหาความเร็ว m_speed โดยที่ m_speed สามารถหาได้จากสูตร

m_speed = 30*10/double(timeperiod*1000 + milliperiod)

โดยที่ m_speed เป็นค่าความเร็วของจักรยาน ซึ่งมีชนิดข้อมูลเป็น double มีหน่วยเป็น เมตรต่อวินาที
timeperiod เป็นช่วงเวลาตั้งแต่ดับครั้งแรกไปยังดับครั้งต่อไป หน่วยเป็นวินาที
milliperiod เป็นช่วงเวลาตั้งแต่ดับครั้งแรกไปยังดับครั้งต่อไป หน่วยเป็นมิลลิวินาที

ซึ่งจะต้องมีการแปลงค่า timeperiod เป็นมิลลิวินาทีก่อนเพื่อจะนำไปรวมกับ milliperiod จากนั้นก็จะนำมาหารกับระยะทางของล้อจักรยานที่หมุนไปซึ่งจะมีค่าเท่ากับ 20 เซนติเมตร ส่วนค่า 10 ที่คูณเข้าไปนั้นเป็นการแปลงหน่วยจาก เซนติเมตรต่อมิลลิวินาที ให้เป็นหน่วย เมตรต่อวินาที

2) ส่วนคำนวณองศาการเลี้ยว

ส่วนนี้จะรับและใช้ข้อมูลจากพอร์ตอนุกรมบิตที่ 0 ถึงบิตที่ 6 โดยค่าที่เปลี่ยนแปลงไปนี้จะเกิดจากการที่แฮนด์ของจักรยานบิดไปมาจนทำให้ตัวต้านทานปรับค่าได้เปลี่ยนแปลง ซึ่งการที่ค่าความต้านทานเปลี่ยนแปลงนี้จะทำให้ค่าความต่างศักย์เปลี่ยน จากนั้นจะนำค่าความต่างศักย์ที่เปลี่ยนแปลงไปแปลงจากสัญญาณอนาลอกให้เป็นสัญญาณดิจิตอลเพื่อใช้ระบุองศาการเลี้ยวต่อไป โดยโฟลวชาร์ตที่แสดงการคำนวณองศาการเลี้ยวสามารถแสดงได้ตามรูปที่ 8-3

รูปที่ 8-3 แสดงโฟลวชาร์ตที่แสดงการคำนวณองศาการเลี้ยว

อธิบายอัลกอริทึมของส่วนคำนวณองศาการเลี้ยว

```
turn = sRxBuf[0] & 0x7f;
```

ตารางที่ 8-23 แสดงการรับข้อมูลองศาการเลี้ยวมาจากพอร์ตอนุกรม

ตารางที่ 8-23 จะเป็นการรับค่ามาจากพอร์ตอนุกรมแล้วใช้การ AND ด้วยค่า 01111111 เพื่อนำเอาเฉพาะบิตที่ 0 ถึงบิตที่ 6 มาใช้ในการคำนวณองศาการเลี้ยว

```
m_turn = (turn * 90)/80;
m_turn = m_turn * (-1);
m_turn = m_turn + 45;
```

ตารางที่ 8-24 แสดงการคำนวณองศาการเลี้ยว

ส่วนตารางที่ 8-24 จะเป็นการปรับช่วงของข้อมูลจาก 0 ถึง 80 ให้อยู่ในช่วง 0 ถึง 90 แล้วคูณด้วย -1 เพื่อให้อยู่ในช่วง -90 ถึง 0 หลังจากนั้นนำค่าที่ได้ไปบวกด้วย 45 เพื่อให้ได้ช่วงของข้อมูลสุดท้ายอยู่ระหว่าง -45 ถึง 45 ซึ่งค่าที่ได้จะเป็นองศาการเลี้ยวที่จะส่งไปให้กับ กราฟฟิกเซิร์ฟเวอร์ ต่อไป

3) ส่วนคำนวณแคลอรี

ส่วนนี้จะใช้วิธีการคำนวณแคลอรีที่สูญเสียไปจากสูตร โดยสูตรที่ใช้จะเกี่ยวข้องกับตัวแปรต่าง ๆ ได้แก่ ความเร็วในการขี่, น้ำหนักที่กดทับล้อจักรยาน และเวลาที่ใช้ในการขี่ โดยหน่วยของตัวแปรทั้ง 3 ตัว จะมีหน่วยเป็น ไมล์ต่อชั่วโมง, กิโลกรัม และชั่วโมง ตามลำดับ โดยสูตรจะเปลี่ยนแปลงไปตามความเร็วในการขี่ ดังตาราง ที่ 8-25

ความเร็วในการขี่	สูตร
1-10 mph	4 * ความเร็ว * น้ำหนัก * เวลา
10-12 mph	6 * ความเร็ว * น้ำหนัก * เวลา
12-14 mph	8 * ความเร็ว * น้ำหนัก * เวลา
14-16 mph	10 * ความเร็ว * น้ำหนัก * เวลา
16-20 mph	12 * ความเร็ว * น้ำหนัก * เวลา
20 mph ขึ้นไป	16 * ความเร็ว * น้ำหนัก * เวลา

ตารางที่ 8-25 แสดงตารางสูตรในการคำนวณแคลอรี

อธิบายอัลกอริทึมของส่วนคำนวณแคลอรี

```
_ftime(&timebuffer);
timeperiod = timebuffer.time - m_calorieTimer.time;
milliperiod = timebuffer.millitm - m_calorieTimer.millitm;
if (milliperiod < 0)
{
        timeperiod--;
        milliperiod += 1000;
}
m_calorieTimer = timebuffer;
m_calorieTime = (double(timeperiod * 1000 + milliperiod)/3600)/1000;
```

ตารางที่ 8-26 แสดงแสดงการจับเวลาเพื่อนำเวลาที่ได้มาใช้คำนวณแคลอรี

โค้ดในตารางที่ 8-26 นี้ เป็นส่วนจับเวลาสำหรับการคำนวณแคลอรีซึ่งจะนำค่าเวลาใหม่ที่อยู่ใน timebuffer มาลบด้วยค่าเวลาเก่าที่เก็บอยู่ใน m_calorieTimer ซึ่งค่าของช่วงเวลาที่แตกต่างกันจะเก็บอยู่ใน timeperiod และ milliperiod หลังจากนั้นก็จะนำค่าดังกล่าวมารวมกันและแปลงหน่วยจากมิลลิวินาที ให้กลายเป็นชั่วโมง โดยหารด้วย 3,600,000 แล้วจึงนำค่าที่ได้ไปเก็บไว้ในตัวแปร m_calorieTime

```
double speedTemp = 1.5*1000/3600;
if (m_speed > 0 && m_speed <= (10 * speedTemp))
{
        m_heart += 4 * (((m_speed / 1.5)/ 1000) * 3600) * m_weight * m_calorieTime;
}
else if (m_speed > (10 * speedTemp) && m_speed <= (12 * speedTemp))
{
        m_heart += 6 * (((m_speed / 1.5)/ 1000) * 3600) * m_weight * m_calorieTime;
}
else if (m_speed > (12 * speedTemp) && m_speed <= (14 * speedTemp))
{
        m_heart += 8 * (((m_speed / 1.5)/ 1000) * 3600) * m_weight * m_calorieTime;
}
else if (m_speed > (14 * speedTemp) && m_speed <= (16 * speedTemp))
{
        m_heart += 10 * (((m_speed / 1.5)/ 1000) * 3600) * m_weight * m_calorieTime;
}
else if (m_speed > (16 * speedTemp) && m_speed <= (20 * speedTemp))
{
        m_heart += 12 * (((m_speed / 1.5)/ 1000) * 3600) * m_weight * m_calorieTime;
}
else if (m_speed > (20 * speedTemp))
{
        m_heart += 16 * (((m_speed / 1.5)/ 1000) * 3600) * m_weight * m_calorieTime;
}
```

ตารางที่ 8-27 แสดงวิธีการในการคำนวณค่าแคลอรีจากสูตร

สำหรับโค้ดส่วนสุดท้ายที่อยู่ในตารางที่ 8-27 จะใช้สำหรับการหาค่าแคลอรีที่สูญเสียไปโดยใช้สูตรที่ได้มาจากตารางด้านบน โดยค่าแคลอรีที่สูญเสียไปจะเก็บอยู่ในตัวแปร m_heart ส่วนตัวแปรตัวอื่น ๆ ที่นำมาใช้คำนวณหาซึ่งได้แก่ m_weight จะเป็นค่าน้ำหนักที่กดทับล้อจักรยานซึ่งก็คือ น้ำหนักของผู้ขี่นั่นเอง และตัวแปร m_calorieTime จะเป็นช่วงเวลาที่จับ

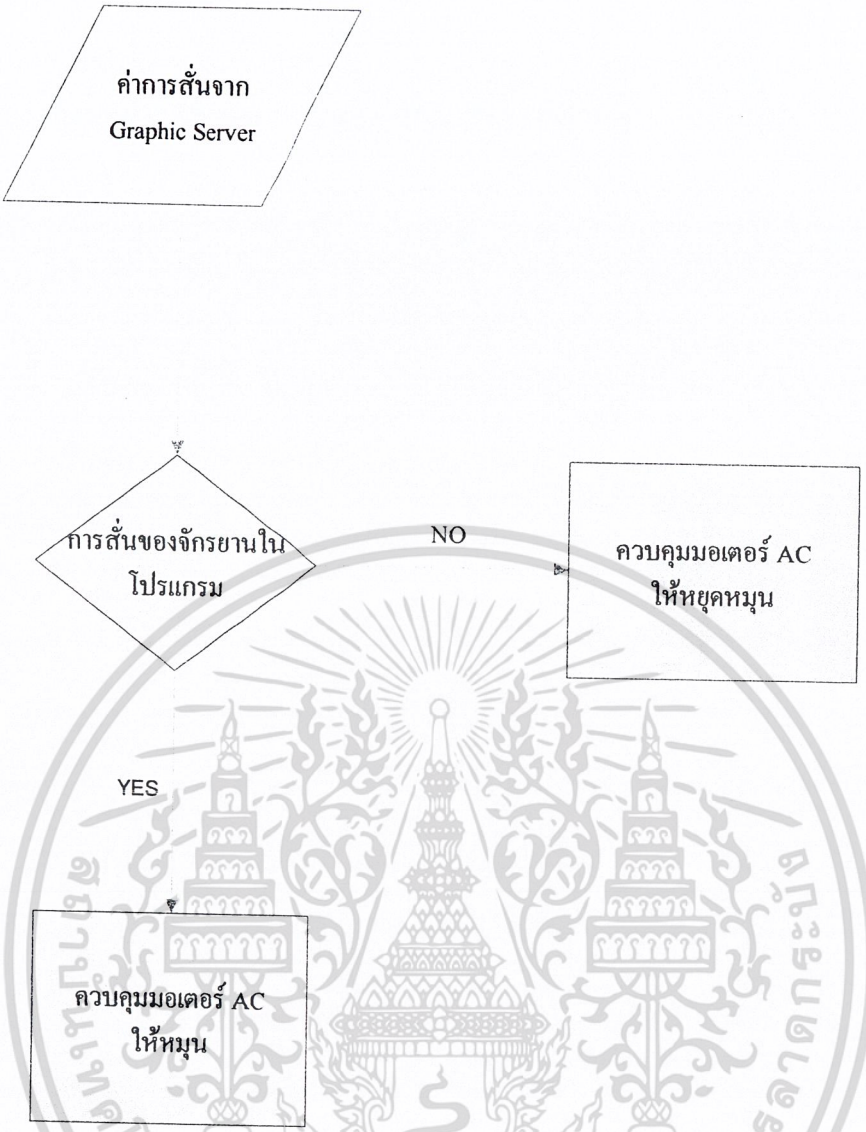
4) ส่วนส่ง feedback กลับไปให้จักรยาน

สำหรับการออกแบบในส่วนนี้ จะใช้ค่าที่รับมาจาก กราฟฟิกเซิร์ฟเวอร์ ซึ่งมี 2 ค่า ได้แก่

- ค่าที่บอกว่าได้เข้าไปในทางลูกระนาดหรือยัง
- ค่าที่บอกว่าจักรยานในโปรแกรม 3 มิติกำลังอยู่ในสถานะใด เงยขึ้นหรือก้มลง

โดยจะนำค่า 2 ค่านี้ไปคำนวณการสั่งงานให้มอเตอร์ต่าง ๆ ที่ใช้เป็นฟีดแบ็คของระบบหมุน ทั้งที่เป็นมอเตอร์ที่ใช้ยกรถจักรยาน และมอเตอร์ที่ใช้สร้างแรงสั่นสะเทือน ซึ่งโฟลวชาร์ตที่แสดงการคำนวณการสั่งงานให้มอเตอร์ทั้ง 2 ตัวหมุน สามารถแสดงได้ตามรูปที่ 8-4 และรูปที่ 8-5 ตามลำดับ

รูปที่ 8-4 แสดงการคำนวณการสั่งงานมอเตอร์ที่ใช้ยกรถจักรยาน



รูปที่ 8-5 แสดงการคำนวณการสั่งงานมอเตอร์ที่ใช้สร้างแรงสั่นสะเทือน

5) ส่วนเชื่อมต่อกับพอร์ตอนุกรม

ในการเชื่อมต่อกับพอร์ตอนุกรมนั้น จะใช้คลาส CSerialPort ซึ่งเป็นคลาสที่ห่อหุ้มการกำหนดค่าต่าง ๆ รวมไปถึงการติดต่อไปยังพอร์ตอนุกรมไว้ ซึ่งจะช่วยให้การติดต่อไปยังพอร์ตอนุกรมทำได้ง่ายขึ้น

อธิบายวิธีการเชื่อมต่อกับพอร์ตอนุกรม

```
try
{
        int comPort = 1;
        port.Open(comPort, 9600, CSerialPort::NoParity, 8,
CSerialPort::OneStopBit,CSerialPort::NoFlowControl);
        DCB dcb;
        port.GetState(dcb);
        dcb.BaudRate = 9600;
        port.SetState(dcb);
        DWORD dwErrors;
        port.ClearError(dwErrors);
        port.SetBreak();
        port.ClearBreak();
        COMSTAT stat;
        port.GetStatus(stat);
        COMMTIMEOUTS timeouts;
        port.GetTimeouts(timeouts);
        port.Setup(10000, 10000);
        port.GetConfig(config);
        config.dcb.BaudRate = 9600;
        port.SetConfig(config);
        port.Set0WriteTimeout();
        port.Set0ReadTimeout();
        serialflag = true;
        m_mshwlist.AddString("Successfully Connected");
        UpdateData(FALSE);
}
 catch (CSerialException* pEx)
{
        TRACE(_T("Handle Exception, Message:%s\n"), pEx-
>GetErrorMessage());
        pEx->Delete();
 }
```

ตารางที่ 8-28 แสดงวิธีการเชื่อมต่อพอร์ตอนุกรม (Connect)

โค้ดในตารางที่ 8-28 เป็นส่วนที่ใช้ในเชื่อมต่อกับพอร์ตอนุกรม โดยในขั้นแรกจะต้องทำการเปิดพอร์ตเสียก่อน ดังตารางที่ 8-29

```
        int comPort = 1;
        port.Open(comPort, 9600, CSerialPort::NoParity, 8,
CSerialPort::OneStopBit,CSerialPort::NoFlowControl);
```

ตารางที่ 8-29 แสดงการเปิดพอร์ตอนุกรมตามค่าที่ระบุ

โค้ดที่อยู่ในตารางที่ 8-29 นี้เป็นการเปิดคอมพอร์ต 1 (ซึ่งกำหนดโดยตัวแปร comport), ใช้บอดเรดเท่ากับ 9600, ไม่มีการใช้พาริตี้บิต, มีบิตข้อมูลเท่ากับ 8 บิต, ใช้สต็อปบิต 1 บิต และไม่มีการใช้การตรวจสอบความถูกต้องของข้อมูลที่ส่ง (flow control)

ส่วนโค้ดที่เหลือในตารางที่ 8-28 จะเป็นการกำหนดรายละเอียดอื่น ๆ ของการติดต่อกับพอร์ตอนุกรม เช่น การกำหนดค่าให้กับ dcb ซึ่งเป็นสตรัคที่ใช้ในการเก็บรายละเอียดของการติดต่อ, การกำหนดไทม์เอาท์สำหรับการรับค่าและการส่งค่า

```
m_nReadTimer = SetTimer(1, 10, 0);
```

ตารางที่ 8-30 แสดงการสร้างไทม์เมอร์ (Timer) ขึ้นมารับข้อมูล

สำหรับโค้ดในตารางที่ 8-30 จะเป็นการเรียกใช้ฟังก์ชัน SetTimer ซึ่งเป็นฟังก์ชันที่ใช้ในการสร้างไทม์เมอร์ (Timer) ขึ้นมาเพื่อรอรับข้อมูลที่ส่งมาจากจักรยาน

อธิบายวิธีการรับข้อมูลจากพอร์ตอนุกรม

ในการรับข้อมูลจากพอร์ตอนุกรมจะใช้วิธีการโพล (Poll) หรือการรอรับค่านั่นเอง โดยจะมีการใช้ไทม์เมอร์นับเวลา ซึ่งถ้าครบตามเวลาที่ไทม์เมอร์จับก็จะเรียกใช้ฟังก์ชันที่ทำหน้าที่รับข้อมูลขึ้นมาทำงาน โดยฟังก์ชันที่จะใช้รับข้อมูลมีชื่อว่า OnTimer มีรายละเอียดของโค้ดดังตารางที่ 8-31

```
void CHardwareServerView::OnTimer(UINT m_nIDEvent)
{
    if (port.Read(sRxBuf,1))
    {
        sRxBuf[1] = NULL;
        ...
        ...
        ...
        port.ClearReadBuffer();
    }
    CWnd::OnTimer(this);
}
```

ตารางที่ 8-31 แสดงโครงสร้างของอัลกอริทึมที่ใช้รับข้อมูลในฟังก์ชัน OnTimer

ภายในฟังก์ชันจะมีการรับพารามิเตอร์ m_nIDEvent เพื่อใช้ระบุว่าเป็นไทม์เมอร์ตัวไหน แต่เนื่องจากโปรแกรมนี้มีการใช้ไทม์เมอร์แค่ตัวเดียว จึงไม่จำเป็นต้องใช้ตัวแปรนี้ หลังจากนั้นก็จะทำการอ่านค่ามาจากพอร์ตอนุกรม 1 ไบต์ ซึ่งถ้ามีค่าอยู่ในบัฟเฟอร์ ก็จะนำค่านั้นมาใช้งานต่อไป และเมื่อใช้งาน

จบแล้วจะต้องเรียกใช้ฟังก์ชัน ClearReadBuffer ด้วย เพื่อลบค่าที่มีอยู่ในบัฟเฟอร์ออกไป เป็นการป้องกันปัญหาบัฟเฟอร์ล้น (Buffer overflow)

อธิบายวิธีการส่งข้อมูลออกไปยังพอร์ตอนุกรม

ในการส่งข้อมูลออกไปพอร์ตอนุกรมนั้น สามารถทำได้อย่างง่าย ๆ โดยการเรียกใช้ฟังก์ชัน Write ของคลาส CSerialPort ดังโค้ดตัวอย่างในตารางที่ 8-32

```
char m_feedback[10];
m_feedback[0] = 0x0f;
m_feedback[1] = NULL;
port.Write(m_feedback,1);
```

ตารางที่ 8-32 แสดงการส่งข้อมูลออกไปยังพอร์ตอนุกรม

โค้ดในตารางที่ 8-32 เป็นการส่งข้อมูลที่มีค่า 00001111 ไปยังพอร์ตอนุกรม

อธิบายวิธีการส่งยกเลิกการติดต่อกับพอร์ตอนุกรม

ในการยกเลิกการติดต่อกับพอร์ตอนุกรมนั้น จะใช้ฟังก์ชัน Close ของคลาส CSerialPort ตามตัวอย่างโค้ดด้านล่าง

```
port.Close();
```

ตารางที่ 8-33 แสดงการปิดพอร์ตอนุกรม

6) ส่วนเชื่อมต่อกับ กราฟฟิกเซิร์ฟเวอร์

สำหรับการติดต่อกับ กราฟฟิกเซิร์ฟเวอร์ นั้น จะใช้ วินซอกส์ ในการติดต่อ โดยมีโพรโตคอล 2 โพรโตคอลให้เลือกใช้ คือ TCP และ UDP ซึ่งโครงงานนี้ได้เลือกใช้โพรโตคอล UDP เพราะเป็นโพรโตคอลที่ส่งได้อย่างรวดเร็ว เนื่องจากไม่ต้องตรวจสอบความถูกต้องของข้อมูลที่ส่งออกไป และเหตุผลที่เลือกใช้โพรโตคอลนี้ก็เพราะโครงงานนี้ต้องการความรวดเร็วในการส่งมากกว่าความถูกต้องของข้อมูลที่ส่งออกไป

การแอพพลิเคชันที่รันบน ฮาร์ดแวร์เซิร์ฟเวอร์ นั้นจะติดต่อไปหาแอพพลิเคชันบน กราฟฟิกเซิร์ฟเวอร์ ได้นั้น แอพพลิเคชันที่รันบน กราฟฟิกเซิร์ฟเวอร์ จะต้องสร้างซอกเก็ต (socket) แล้วทำการผูก (bind) ซอกเก็ตที่สร้างกับพอร์ตหนึ่งพอร์ตใดก่อนเพื่อทำให้พอร์ตนั้นเป็นพอร์ตที่เครื่องอื่นรู้จักและมองเห็นได้ (well-known port) ก่อน

หลังจากนั้นแอพพลิเคชันบน ฮาร์ดแวร์เซิร์ฟเวอร์ ก็จะสามารถส่งข้อมูลไปให้ กราฟฟิกเซิร์ฟเวอร์ ได้โดยระบุแค่ไอพีแอดเดรสและพอร์ตของ กราฟฟิกเซิร์ฟเวอร์ เท่านั้น

การติดต่อไปหา กราฟฟิกเซิร์ฟเวอร์ ผ่านทาง วินซอกส์ โดยใช้โพรโตคอล UDP มีวิธีการทำดังนี้

1) ใช้คลาส CASock ซึ่งเป็นคลาสที่ใช้สำหรับการเชื่อมต่อระหว่างคอมพิวเตอร์ที่มีอยู่แล้วในการพัฒนาโปรแกรมแบบ MFC

2) สร้างการเชื่อมต่อขึ้นมาก่อน โดยตัวอย่างจะเป็นการกำหนดให้ติดต่อไปยังพอร์ต 4000 และใช้โพรโตคอล UDP (กำหนดจาก SOCK_DGRAM) ดังโค้ดในตารางที่ 8-34

```
int errorcode = m_connectSock.Create(4000, SOCK_DGRAM);
```

ตารางที่ 8-34 แสดงการสร้างการเชื่อมต่อโดยใช้โพรโตคอล UDP

โดยที่ errorcode จะเป็นค่าที่รีเทิร์นมาจากฟังก์ชัน Create ซึ่งถ้า errorcode เป็น 0 จะหมายความว่าเกิดปัญหาขึ้นระหว่างการพยายามสร้างการเชื่อมต่อ

3) หากต้องการส่งข้อมูลไปยัง กราฟฟิกเซิร์ฟเวอร์ จะใช้ฟังก์ชัน SendTo โดยระบุแค่แอดเดรสและพอร์ตปลายทางเท่านั้น ดังตัวอย่างโค้ดในตารางที่ 8-35

```
    dwBytes = parent-
>m_connectSock.SendTo((LPCTSTR)parent->m_stringToGP,
BytesSent, 4000, “161.246.6.7”);

    if (dwBytes == SOCKET_ERROR)
    {
        if (parent->m_connectSock.GetLastError() !=
WSAEWOULDBLOCK)
        {
            wsprintf(strError, "Socket failed to send,
        Error Code: %d", parent-
        >m_connectSock.GetLastError());
            AfxMessageBox (strError);
            return 0;
        }
    }
    else
    {
    }
```

ตารางที่ 8-35 แสดงการส่งข้อมูลไปให้ กราฟฟิกเซิร์ฟเวอร์ โดยใช้ฟังก์ชัน SendTo

โดยค่าที่รีเทิร์นมาจากฟังก์ชัน SendTo จะเป็นค่าที่บอกจำนวนไบต์ข้อมูลที่ได้ส่งออกไป และถ้าค่านี้เป็น 0 จะหมายความว่า เกิดปัญหาขึ้นระหว่างการส่งข้อมูลซึ่งจะต้องนำมาตรวจสอบและแก้ไขปัญหาต่อไป

4) หลังจากใช้งาน วินซอกส์ เสร็จแล้ว ก็จะต้องยกเลิกการเชื่อมต่อโดยเรียกใช้ฟังก์ชัน Close ของคลาส CASock ตามตัวอย่างในตารางที่ 8-36

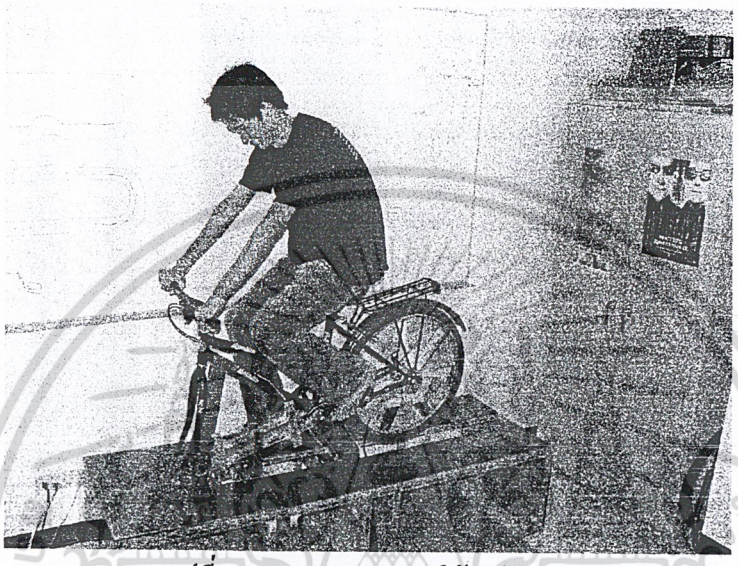
```
m_connectSock.Close();
```

ตารางที่ 8-36 แสดงการยกเลิกการเชื่อมต่อ

บทที่ 9
การทดลองทำงานโดยรวมทั้งหมด

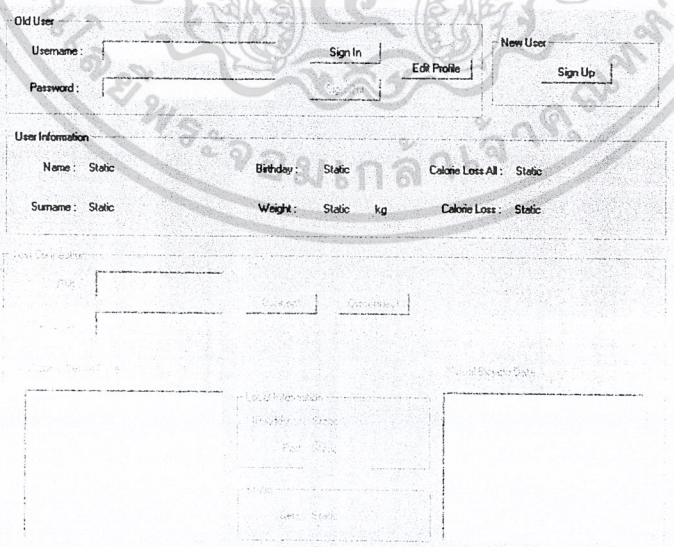
การทำงานโดยรวมจะมีส่วนหลัก ๆ ทั้งหมด 3 ส่วน

1) ส่วนของจักรยาน จะมีการส่งข้อมูลความเร็ว การเลี้ยว และรับฟีดแบ็คที่ใช้ในการยกจักรยานให้ขึ้นและลง สร้างแรงสั่นสะเทือนให้กับจักรยาน รวมทั้งระบบหน่วงล้อรถจักรยาน



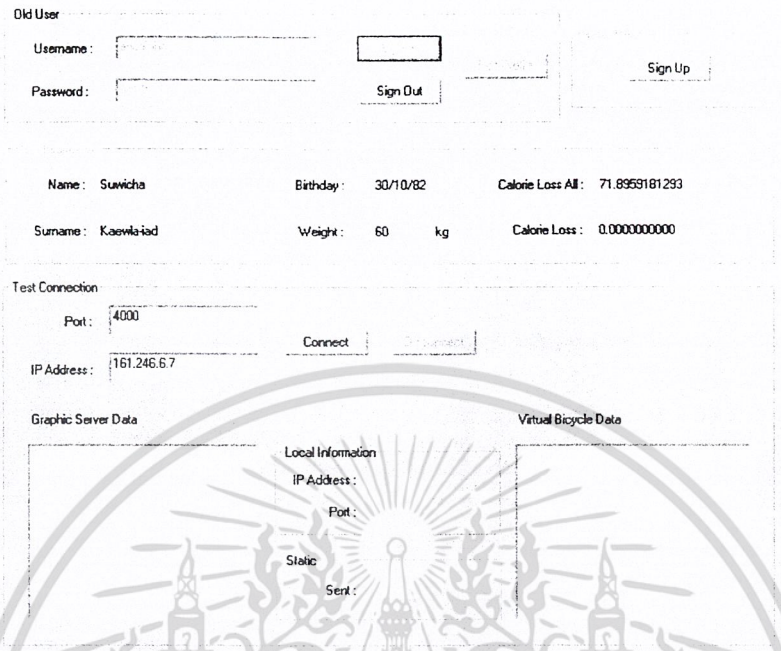
รูปที่ 9-1 แสดงการทดลองใช้งานระบบ

2) ส่วนประกอบภายในโปรแกรมรับส่งข้อมูลระหว่างตัวจักรยานและส่วนแสดงผลภาพ 3 มิติ ที่อยู่ในฮาร์ดแวร์เซิร์ฟเวอร์ นั้นจะมีอินเตอร์เฟสกับผู้ใช้ดังรูปที่ 9-2



รูปที่ 9-2 แสดงอินเตอร์เฟสของโปรแกรมที่ ฮาร์ดแวร์เซิร์ฟเวอร์

ขั้นแรกเราต้องใส่ชื่อและน้ำหนักแล้วทำการ Log in การใส่ชื่อและน้ำหนักตรงนี้เพื่อที่จะนำไปใช้คำนวณหาแคลอรีที่ได้ใช้ไปในการมาใช้งานเครื่องนี้ ดังแสดงในรูปที่ 9-3

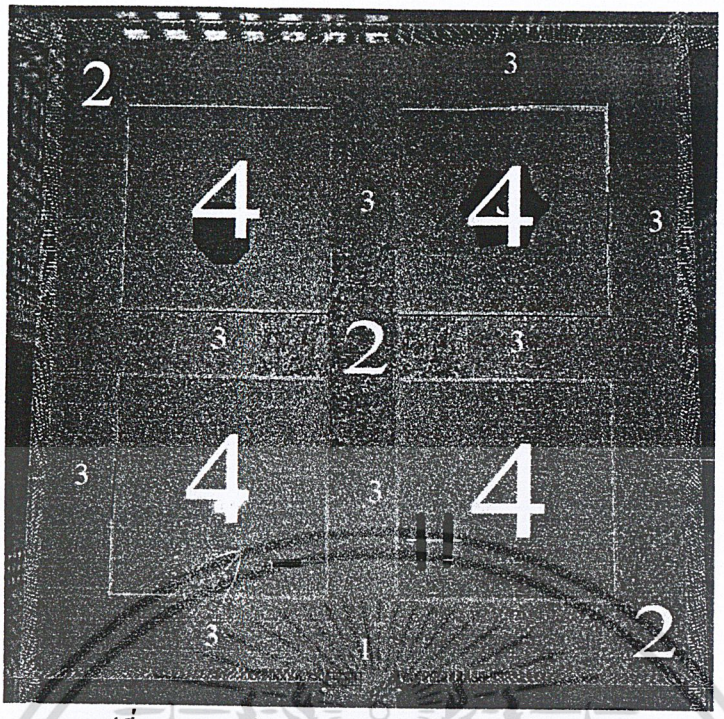


รูปที่ 9-3 แสดงวิธีการ Log in เข้าใช้งานระบบ

หลังจาก Log In มาแล้วจะต้องติดต่อกับส่วนที่ใช้แสดงผลภาพ 3 มิติ และรถจักรยาน โดยใส่ IP ของเครื่องที่จะใช้แสดงผลจากนั้น ก็กดปุ่ม connect เพื่อจะให้ติดต่อกัน

ในระหว่างที่มีการส่งข้อมูลระหว่างส่วนแสดงผลภาพ 3 มิติ และ ส่วนตัวจักรยาน ข้อมูลที่ส่งไปมากับตัวจักรยานจะมาแสดงให้เห็นที่ Message To HW และข้อมูลที่ส่งไปมาระหว่างส่วนแสดงผลภาพ 3 มิติจะมาแสดงให้เห็นที่ Message To GPServer ทั้งนี้เพื่อใช้ในการตรวจสอบ (Debug) หาความผิดพลาดที่เกิดขึ้น

3) ส่วนของการแสดงผลภาพ 3 มิติ



รูปที่ 9-4 แสดงส่วนประกอบภายในโลกเสมือน 3 มิติ

ส่วนประกอบภายในโลกเสมือน 3 มิติ ในรูปที่ 9-5 ประกอบด้วย

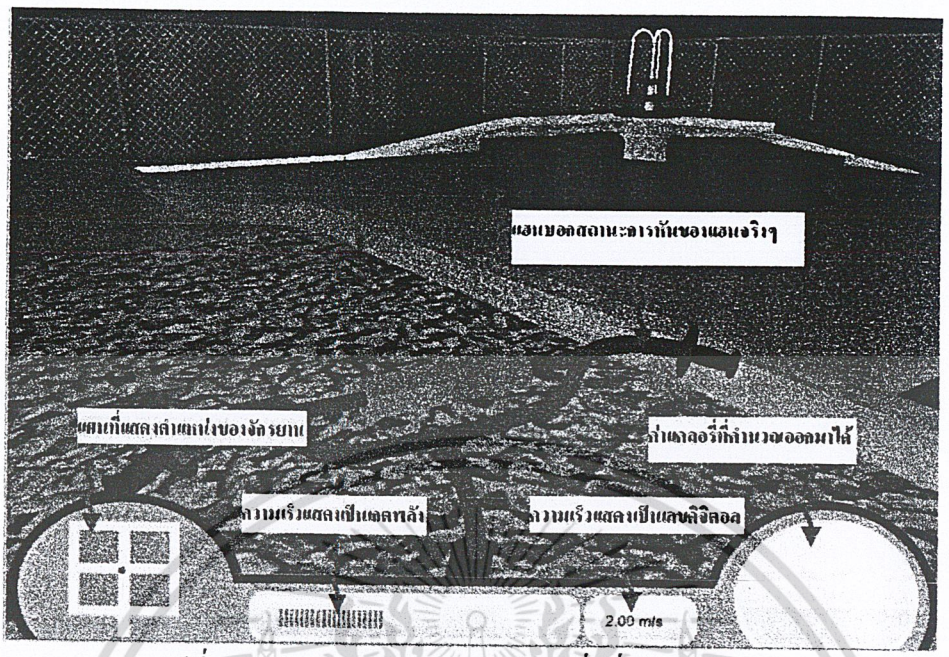
1. ทางที่รถจักรยานจะวิ่ง
2. ทางขรุขระ
3. ทาง ขึ้น-ลง เนิน
4. สวนหย่อม

และก็จะมีประตูทางเข้า รั้วกั้นรอบๆ ทางวิ่งของจักรยาน และโมเดล (model) ของวัตถุต่าง ๆ ที่จะช่วยให้เสมือนจริงขึ้นและรอบ ๆ จะมี Skybox หุ้มรอบ ๆ แผนที่นี้เพื่อให้ผู้ใช้รู้สึกว่าตัวเองอยู่ในสิ่งแวดล้อมเสมือน

ลักษณะของโลกเสมือน 3 มิติ

1. แผนที่จะเป็นรูปสี่เหลี่ยม
2. ผู้ใช้ไม่สามารถออกจากแผนที่นี้ได้ คือ ไม่สามารถไปไกลกว่าแผนที่ที่เตรียมไว้ให้ได้
3. ภายในโลกเสมือนจะไม่มีการขี่ถอยหลัง แผนที่ทุกทางจะสามารถขี่วนไปได้ทุกที่

ลักษณะการแสดงภาพที่ผู้ใช้จะเห็น จะเป็นมุมมองของบุคคลที่ 1 (First Person View)



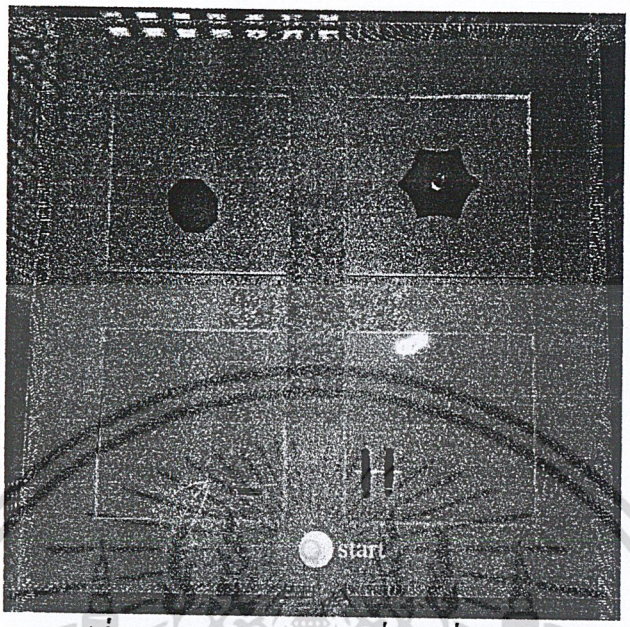
รูปที่ 9-5 แสดงการแสดงภาพและข้อมูลที่ผู้ขี่จักรยานจะเห็น

ในหน้าต่างที่ผู้ใช้เห็นนี้จะบอกสิ่งต่างๆให้กับผู้ใช้ได้รู้ตามรูปที่ 9-6 ดังนี้

1. ตำแหน่งของผู้เล่นที่อยู่ใน แผนที่ โดยจะแสดงไว้ที่ console ทางด้านซ้าย โดยจุดสี่แดงจะเป็นตำแหน่งของผู้ใช้
2. ความเร็วจะแสดงออกมาได้ 2 ลักษณะ คือในแบบของเกจแสดงความเร็ว (Speed Gauge) และในแบบของตัวเลขดิจิตอล
3. ที่แฮนด์เมื่อเราได้ทำการเปลี่ยนองศาก็จะนำมาแสดงในรูปแบบของแฮนด์ที่อยู่โลก 3 มิติ
4. เมื่อผู้ใช้ป้อนค่าต่างๆลงไปโปรแกรมจะคำนวณหาค่าแคลอรี่ที่ได้ใช้ไปและแสดงออกมาที่ console ทางด้านขวามือ

การเดินทางภายในโลกเสมือน 3 มิติ

1. เริ่มต้น ผู้ใช้จะอยู่ตำแหน่งตรงกลางด้านล่างของแผนที่ ตามรูปที่ 9-7



รูปที่ 9-6 แสดงตำแหน่งของผู้ขี่ตอนเริ่มเกม

2. หากผู้เล่นขับชนขอบทางจะเกิดเสียงดังขึ้นและรถจะถอยกลับมาระยะหนึ่ง เพื่อผู้เล่นได้ไม่ต้องถอยเองหากเกิดการชนเข้า
3. เมื่อผ่านทางบรุขระจะเกิดเสียงและภาพจะเกิดการสั่นแสดงให้รู้ว่าผู้ใช้ได้ผ่านทางบรุขระอยู่

บทที่ 10
วิจารณ์และสรุป

10.1 สรุปการทำงาน

โครงงานนี้พัฒนาได้สำเร็จลุล่วงตามเป้าหมาย โดยได้ทำการออกแบบและสร้างแอพพลิเคชันที่ใช้จำลองโลกเสมือน 3 มิติขึ้นมาที่คอมพิวเตอร์ที่เรียกว่า กราฟฟิกเซิร์ฟเวอร์ ซึ่งภายในแอพพลิเคชันจะมีการใช้ความรู้ที่ได้ศึกษามาทั้งเรื่องของการเขียนโปรแกรมโดยใช้ภาษา C++ ในรูปแบบของ win32 แอพพลิเคชันที่มีการใช้คอมโพเนนท์ไดเร็กซ์เอ็กซ์ร่วมด้วย สำหรับคอมโพเนนท์ไดเร็กซ์เอ็กซ์ที่ใช้ ได้แก่ การสร้างและแสดงภาพวัตถุ 3 มิติ, การสร้างกล้องเสมือนขึ้นมาใช้, การสร้างและแสดงภาพ 2 มิติ, การสร้างเสียง เป็นต้น

ในส่วนของคอมพิวเตอร์ที่เป็นตัวประมวลผล ที่เรียกว่า ฮาร์ดแวร์เซิร์ฟเวอร์ นั้นได้มีการออกแบบ สร้าง และพัฒนาอัลกอริทึมที่ใช้ในการคำนวณค่าต่าง ๆ เช่น ความเร็ว, องศาการเลี้ยว, แคลอรี, วิธีการสร้าง feedback รวมไปถึงสร้างรูปแบบของการเชื่อมต่อและรับส่งข้อมูลระหว่างตัวจักรยานกับแอพพลิเคชัน 3 มิติให้ดีที่สุด

และในส่วนของการทำงานทางด้านฮาร์ดแวร์ ก็ได้ทำการออกแบบและพัฒนาในส่วนของอินพุทจากจักรยาน ซึ่งมีทั้ง ระบบเซ็นเซอร์แสงที่นำโฟโต้ไดโอดไปติดกับล้อหลังของรถจักรยานเพื่อเอาไว้ตรวจวัดความเร็วของการปั่นจักรยาน, ระบบตรวจวัดองศาของการเลี้ยวโดยใช้ตัวต้านทานปรับค่าได้ซึ่งได้นำไปติดไว้กับแฮนด์ของจักรยาน

สำหรับส่วนของเอาท์พุทหรือฟีดแบ็ค (feedback) ของจักรยาน ก็ได้ทำการออกแบบและพัฒนาดังนี้ ระบบสร้างแรงสั่นสะเทือนที่ใช้มอเตอร์กระแสสลับหมุนลูกเบี้ยวให้เกิดแรงเหวี่ยงจนทำให้รถจักรยานสั่นเมื่อถึงทางที่เป็นลูกระนาด, ระบบสร้างแรงต้านการปั่นหรือระบบหน่วงล้อรถจักรยานที่ใช้สเตปปิ้งมอเตอร์, ระบบยกรถจักรยานขึ้นลงตามลักษณะพื้นถนนในแอพพลิเคชัน 3 มิติที่ใช้มอเตอร์กระแสตรงกำลังสูงในการยก

ลักษณะการทำงานโดยรวมคือ เมื่อผู้ใช้ปั่นจักรยาน ภาพในแอพพลิเคชัน 3 มิติก็จะเคลื่อนที่ตามความเร็วที่ปั่น และเมื่อผู้ใช้มีการเลี้ยวซ้ายหรือขวา แฮนด์ของจักรยานในแอพพลิเคชัน 3 มิติก็จะบิดไปตามองศาการเลี้ยว

เมื่อผู้ใช้ปั่นไปชนวัตถุที่อยู่ในแอพพลิเคชันก็จะเกิดเสียงของการชนขึ้น เมื่อผู้ใช้ปั่นเข้าไปในทางลูกระนาดก็จะเกิดเสียงของการสั่นของจักรยาน รวมทั้งมีการส่งฟีดแบ็คกลับไปเพื่อให้ตัวรถจักรยานสั่น และเมื่อผู้ใช้ปั่นจักรยานขึ้นหรือลงสะพาน ระบบจะเพิ่มแรงต้านการปั่นรวมไปถึงยกรถจักรยานขึ้นในขณะปั่นขึ้นสะพาน และลดแรงต้านการปั่นรวมไปถึงยกรถจักรยานลงในขณะปั่นลงสะพาน

10.2 วิจารณ์สิ่งที่ได้จากโครงงาน

สิ่งที่ได้รับจากการพัฒนาโครงงานระบบจำลองการขี่รถจักรยานในสภาพแวดล้อมเสมือน มีดังนี้

1) ความรู้ที่ใช้ในโครงงานซึ่งเป็นความรู้ที่คนทั่วไปไม่อาจหาอ่านได้ง่ายนัก เช่น ความรู้ในการเขียนโปรแกรม 3 มิติด้วยไดเร็กซ์เอ็กซ์ซึ่งน้อยคนในประเทศไทยที่จะมีความรู้ความเชี่ยวชาญทางด้านนี้ ทั้งนี้เพราะงานทางด้านมัลติมีเดียในประเทศไทยยังมีอยู่น้อยเมื่อเทียบกับงานโปรแกรมเมอร์ด้านอื่น ๆ

2) วิธีการแก้ปัญหาที่อาจเกิดขึ้นได้ตลอดการทำงาน เนื่องจากการทำงานทุกอย่างอาจเกิดปัญหาได้เสมอ โครงงานนี้ก็เช่นเดียวกัน ก็มีปัญหาต่าง ๆ เกิดขึ้นมากมาย ซึ่งสิ่งที่ได้รับจากปัญหานั้น คือ ตัวปัญหา, สาเหตุที่ทำให้เกิดปัญหานั้น, ความรุนแรงของปัญหา, วิธีในการแก้ปัญหา และระยะเวลาที่ใช้ในการแก้ปัญหา ซึ่งสิ่งเหล่านี้จะช่วยให้คนที่เคยมีประสบการณ์ในการเผชิญกับปัญหาเหล่านี้มาก่อน มีวิธีการในการหลีกเลี่ยงปัญหา และแก้ปัญหาเมื่อเกิดปัญหาขึ้นมาที่ดีกว่าคนที่ไม่เคยเผชิญปัญหาดังกล่าวมาก่อน

3) การออกแบบงาน และแบ่งงานกันทำเป็นทีม ซึ่งจะต้องรู้จักวางแผนและออกแบบงานให้ดี รวมไปถึงการมอบหมายงานให้ทำตามความถนัดและความชอบของแต่ละคนด้วย งานที่ได้ออกมาจึงจะสำเร็จลุล่วงไปได้ด้วยดี

4) ความสนุกสนานเพลิดเพลิน เนื่องจากโครงงานนี้เป็นโครงงานทางด้านมัลติมีเดียที่เน้นทางด้านความเพลิดเพลินเป็นหลัก ดังนั้นสิ่งที่ได้รับอีกอย่างจากโครงงานนี้ก็คือ ความสนุกสนานเพลิดเพลินจากตัวโครงงาน

10.3 ปัญหาที่เจอและแนวทางในการแก้ไข

1. การคำนวณความเร็ว การคำนวณองศาการเลี้ยว การคำนวณแคลอรีและการส่งคำสั่งในการหมุนมอเตอร์ยังมีความคลาดเคลื่อนอยู่ เพราะอุปกรณ์ที่ใช้ในการวัดยังมีความผิดพลาดอยู่, การผ่านส่งข้อมูลจากต้นทางไปยังปลายทางก็อาจมีดีเลย์ รวมถึงการใช้การจับเวลาด้วยคอมพิวเตอร์ก็อาจได้ค่าที่ไม่แม่นยำ ซึ่งแนวทางในการแก้ไขปัญหานี้ คือ การคิด ทดสอบและปรับปรุงอัลกอริทึมที่ใช้ให้ดีขึ้นเรื่อย

2. การสร้างภาพ 3 มิติของวัตถุต่าง ๆ ที่จะนำมาแสดงเป็นสิ่งแวดล้อมของโปรแกรมเป็นงานที่ต้องใช้เวลาในการทำอย่างมาก จึงไม่สามารถออกแบบวัตถุให้มีรายละเอียดมาก ๆ ได้ อีกทั้งถ้านำวัตถุที่มีรายละเอียดมาก ๆ มาใช้ จะทำให้การแสดงภาพ 3 มิติช้าลงไปมาก ซึ่งแนวทางในการแก้ไขปัญหานี้ คือ ออกแบบวัตถุให้มีรายละเอียดน้อยที่สุด หรือถ้าเป็นวัตถุที่มีรายละเอียดมาก ก็จะใช้วิธีการหาจากอินเตอร์เนตแล้วนำมาใช้ในโปรแกรม

3. วงจรที่ติดตั้งที่ตัวจักรยานอุปกรณ์ต่าง ๆ มีความผิดพลาดได้ ทำให้เวลาเริ่มใช้งานทุกครั้งต้องมีการปรับค่าและทดลองก่อนใช้งาน การแก้ไขทำได้โดยพยายามให้ผู้ใช้งานมีความเข้าใจการทำงานต่าง ๆ เพื่อจะได้ปฏิบัติได้อย่างถูกต้อง ทำให้ลดเวลาในการปรับค่าเริ่มต้น

4. การที่มีอุปกรณ์ต่อพ่วงหลายส่วนทำให้มีการใช้กระแสมาก และมีการใช้ระดับความต่างศักย์หลายค่าทำให้ ต้องมีวงจรที่ทำหน้าที่จ่ายไฟให้ตามความต้องการ และอีกทั้งอุปกรณ์บางตัวใช้กำลังมาก

เมื่อใช้แหล่งจ่ายไปร่วมกันทำให้เกิดการแย่งกระแสกันจนทำให้การใช้งานผิดปกติ การแก้ไขโดยออกแบบการจ่ายไฟให้เหมาะสมกับความต้องการ และมีการแยกส่วนจ่ายไฟสำหรับส่วนที่ใช้กระแสมาก ๆ เพื่อให้การทำงานเป็นไปอย่างมีเสถียรภาพ

5. การใช้งานจริงนั้นมีปัญหากับน้ำหนักของผู้ใช้งานเนื่องจากตัวมอเตอร์มีกำลังที่ต่ำ จึงต้องมีการปรับแก้เรื่องการตอบสนองที่เร็ว ๆ และให้กำลังได้แรงกว่านี้ การแก้ไขโดยการหามอเตอร์ที่มีกำลังมากกว่าปัจจุบันมาใช้งาน หรืออาจเปลี่ยนแนวคิดใหม่โดยอาจเปลี่ยนตัวอุปกรณ์หรือโครงสร้างที่มีการรับน้ำหนักและทดแรงได้ดีกว่านี้

10.4 แนวทางการพัฒนาต่อ

1) พัฒนาในส่วนของกราฟฟิกให้มีความสวยงามมากขึ้น และเพิ่มขนาดของเวิร์ลที่ใช้ในแอพพลิเคชันให้ใหญ่ขึ้น ทั้งนี้จะต้องคำนึงถึงความสามารถของเครื่องด้วยว่า มีความสามารถในการแสดงภาพ 3 มิติได้ดีเพียงใด

2) พัฒนาในส่วนของอุปกรณ์ฮาร์ดแวร์ที่ติดกับตัวจักรยานให้ดีขึ้น โดยจะต้องใช้อุปกรณ์ตรวจวัดความเร็ว อุปกรณ์ตรวจวัดองศาการเลี้ยว และอุปกรณ์ที่ใช้ทำ feedback ที่มีความละเอียดในการวัดและมีคุณภาพมากขึ้น

3) พัฒนาในส่วนของตัวแอพพลิเคชันให้เป็นแอพพลิเคชันแบบออนไลน์ ซึ่งมีแนวโน้มที่เป็นไปได้ในอนาคต เพราะแอพพลิเคชันออนไลน์กำลังได้รับความนิยมอย่างมากในปัจจุบัน

บรรณานุกรม

[1] นิรุธ อำนวยศิลป์ : คู่มือเขียนโปรแกรม Visual C++ Version 6.0 ฉบับเพื่อใช้งานจริง , บริษัท ซัคเส มีเดีย จำกัด,2544

[2] นิรุธ อำนวยศิลป์ : เขียนเกมอย่างมืออาชีพด้วย Visual C++ และ DirectX ,อินโฟเพรส,2545

[3] Saeef B. Niku : Introfuction to Robotics Analysis,Systems,Applications,Prentice Hall,2001

[4] วรพจน์ กรแก้ว และ ชัยวัฒน์ ลิ้มพรจิตรวิไล : เรียนรู้และปฏิบัติการ ไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช , Innovative Experiment Co.,Ltd.

[5] Jim Adams : Programming Role Playing Games with DirectX, Premier Press, Inc.,2002

[6] Frank D. Luna : Introduction to 3D Game Programming with DirectX® 9.0, Wordware Publishing, Inc.,2003

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

[8] ยุทธนา ลีลาศวัฒนกุล : คู่มือเขียนโปรแกรมและใช้งาน Visual C++ .NET ฉบับสมบูรณ์,Infopress Developer Book,2546