ระบบจำลองการขับรถเสมือนจริง

Virtual Driving Simulator

นาย ทศพล พิสูจน์ศิลป์
นางสาว นันทวรรณ จิระวัฒนพงศ์
นาย นิทัศน์ อภิมนต์บุตร

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2547

ระบบจำลองการขับรถเสมือนจริง

Virtual Driving Simulator

โดย

นาย ทศพล พิสูจน์ศิลป์

นางสาว นันทวรรณ จิระวัฒนพงศ์

นาย นิทัศน์ อภิมนต์บุตร

อาจารย์ที่ปรึกษา

ดร. สมศักดิ์ วลัยรัชต์

ดร. อรัญญา วลัยรัชต์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2547

ปริญญานิพนธ์ปีการศึกษา 2547

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบจำลองการขับรถเสมือนจริง

Virtual Driving Simulator

ผู้จัดทำ

1. นาย ทศพล พิสูจน์ศิลป์
2. นางสาว นันทวรรณ จิระวัฒนพงศ์
3. นาย นิทัศน์ อภิมนต์บุตร

_____ อาจารย์ที่ปรึกษา

(ดร. สมศักดิ์ วลัยรัชต์)

_____ อาจารย์ที่ปรึกษา

(ดร. อรัญญา วลัยรัชต์)

บทคัดย่อ

การฝึกหัดขับรถในเบื้องต้น ผู้ขับขี่จำเป็นจะต้องมีผู้สอนขับรถ และที่สำคัญคือ จะต้องมีรถยนต์ ซึ่งหลายๆ ครั้งผู้ที่หัดขับรถต้องการจะหัดขับรถยนต์ แต่ไม่สามารถหารถที่จะใช้ในการฝึกหัดขับได้ เนื่องจากเหตุผลหลายๆ ประการ นอกจากนี้หากผู้ที่หัดขับรถใหม่ๆ เตรียมความพร้อมไม่เพียงพอที่จะไปสอบจริง ก็อาจต้องเสียเวลาในไปสอบใหม่ในภายหลัง

โครงงานระบบจำลองการขับรถเหมือนจริง(Virtual Driving Simulator) เป็นระบบที่จำลองสภาพการขับขี่ให้คล้ายกับผู้ขับขี่ได้เข้าไปนั่งขับอยู่ในรถยนต์จริง โดยจะต้องพัฒนาโปรแกรมจำลองภาพ 3 มิติ และมีการรับอินพุทจากอุปกรณ์ในการขับรถต่างๆ ได้แก่ พวงมาลัยบังคับทิศทาง , คันเร่ง, เบรก, คลัช และ ระบบเกียร์ธรรมดา ซึ่งอุปกรณ์เหล่านี้จะทำงานสัมพันธ์กับโปรแกรม และจะสร้างภาพ 3 มิติออกทางจอแสดงผล โดยที่การแสดงผลนั้นเป็นมุมมองที่มองจากภายในห้องคนขับของรถยนต์ออกไปที่ถนน ช่วยให้ผู้ขับขี่สามารถฝึกซ้อมขับจนเกิดความชำนาญได้ก่อนที่จะไปทดสอบกับรถยนต์จริง

การพัฒนาส่วนของโปรแกรม ใช้ภาษาซีพลัสพลัส และใช้กราฟฟิคไลบรารีไดเร็กเอ็กซ์ในการพัฒนามัลติมีเดีย 3 มิติ และแสดงผลเสียงโดยใช้ไดเร็กซาวน์ ในส่วนของการรับค่าจากอุปกรณ์ต่างๆ ใช้ ไดเร็กอินพุท ในการนำค่าจากอุปกรณ์ไปประมวลผลร่วมกับโปรแกรม 3 มิติ การเชื่อมต่ออุปกรณ์กับคอมพิวเตอร์จะทำโดยผ่านเกมพอร์ต

ABSTRACT

In order to practice driving, the driver needs a trainer and also a car. Many times the driver cannot find a car to practice because of some reasons, moreover if the driver has not prepare enough driving skill for performing the real driving license test, he may fail the test and waste the time for retest.

The Driving Simulator Project is the system that simulates the real driving environment to the driver, by developing 3D simulator program which gets input from devices. Devices compose of steering wheel, accel, brake, clutch and manual gear. These devices operates with the program for simulating 3D graphic to the display monitor which shows driver's view looks into the road. This enables the new driver can practice before doing the real test.

Program parts is developed by C++ , 3D multimedia is developed by DirectX and uses DirectSound to play the sound. Input is get by DirectInput through Game Port, brings input value to calculate in program and simulate 3D graphic to the driver.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ ต้องขอขอบคุณความร่วมมือจาก ภาควิชาวิศวกรรมคอมพิวเตอร์ ที่ได้จัดอำนวยห้องแลปในการวิจัยและพัฒนาให้ อีกทั้งยังมีฝ่ายสโตร์ที่ช่วยจัดสรรทรัพยากร ให้ยืมอุปกรณ์ เครื่องมือเครื่องใช้ที่จำเป็นสำหรับการทำงาน นอกจากนี้ขอขอบคุณเพื่อนๆ ที่คอยให้คำปรึกษาเป็นกำลังใจให้ ในการแก้ปัญหาในการทำงาน

โครงงานนี้คงไม่อาจสำเร็จได้หากไม่มีอาจารย์ที่ปรึกษา ทั้งสองท่าน อาจารย์ สมศักดิ์ วลัยรัชต์ และ อาจารย์ อรัญญา วลัยรัชต์ ที่ได้ดูแลข้าพเจ้าเป็นอย่างดีตลอดมา ตั้งแต่ให้คำปรึกษา ตรวจข้อผิดพลาดและแนะแนวทางงานที่ทำ รวมถึงกรุณาจัดหาอุปกรณ์ที่จำเป็นให้มาโดยตลอด ต้องขอกราบขอบพระคุณอย่างสูงเอาไว้ ณ ที่นี้ด้วย

สารบัญ

สารบัญตาราง

สารบัญรูป

บทที่ 2

บทที่ 3

บทที่ 4

บทที่ 5

บทที่ 1
บทนำ

1.1 หลักการและเหตุผล

เนื่องจากปัจจุบันการฝึกหัดขับรถยนต์เพื่อจะไปสอบใบอนุญาตขับขี่รถยนต์นั้น สำหรับประชาชนที่อยู่ในเมืองใหญ่ เช่น กรุงเทพฯ นั้นเป็นไปได้ไม่สะดวก เนื่องจากขาดสถานที่ฝึกซ้อมหรือไม่มีเวลา บางครั้งก็ต้องอาศัยหน่วยงานเอกชนที่เปิดสอนขับขี่ซึ่งก็ทำให้ต้องเสียค่าใช้จ่ายสูงและต้องเสียเวลาที่จะไปเข้าอบรม จึงเห็นความสำคัญที่จะต้องมีระบบจำลองการขับรถเสมือนจริงสำหรับผู้ที่ต้องการสอบใบอนุญาตขับขี่รถยนต์ได้ทดลองฝึกขับรถให้เกิดความชำนาญ สามารถนำมาใช้ฝึกฝนเวลาใดก็ได้ตามแต่สะดวก นอกจากนี้เพื่อความป้องกันความเสียหายที่เกิดขึ้นกับทั้งผู้ขับขี่และรถยนต์ด้วย ในกรณีที่ขับรถชน จึงได้มีหลักสูตรการสอบใบขับขี่และอื่นๆ อีกมาก สามารถอัพเดทสถานการณ์การขับขี่เพิ่มเติมแบบใหม่ๆได้อีกด้วย ทำให้การฝึกหัดขับรถหรือจะถือว่าเป็นการเล่นเกมที่ได้ทั้งความบันเทิง และได้ทั้งความชำนาญในการฝึกหัดขับรถไปด้วยพร้อมๆกัน สำหรับประเทศไทยการพัฒนาโปรแกรมทางด้านนี้ยังมีอยู่ไม่มากนักจึงเป็นแรงบันดาลใจในการพัฒนาโปรแกรมจำลองการขับรถเสมือนจริงนี้

1.2 วัตถุประสงค์

1. เพื่อจำลองลักษณะการขับรถยนต์ในลักษณะที่ผู้ใช้นั่งขับรถกับอุปกรณ์ต่อพ่วงกับคอมพิวเตอร์ที่มีลักษณะเดียวกับการขับรถยนต์จริง
2. เพื่อเป็นแนวทางการศึกษา และฝึกฝนเพื่อเตรียมความพร้อมสำหรับการสอบใบอนุญาตขับขี่รถยนต์ประเภทต่างๆ
3. เพื่อฝึกฝนพัฒนาโปรแกรมแสดงภาพคอมพิวเตอร์กราฟฟิคแบบ 3 มิติ ในลักษณะมัลติมีเดียรูปแบบต่างๆ เช่น ภาพ เสียง และการปฏิสัมพันธ์ระหว่างมนุษย์กับคอมพิวเตอร์
4. เพื่อศึกษาการใช้งานกราฟฟิคไลบรารีไดเร็กเอ็กซ์ในการพัฒนาโปรแกรม

1.3 ปัญหาหรือประโยชน์ที่เป็นเหตุผลให้ควรพัฒนาโครงงาน

1. เนื่องจากประชาชนส่วนใหญ่ที่อยู่ในเมืองไม่ค่อยมีเวลา และไม่สะดวกในการหาสถานที่เพื่อฝึกหัดขับรถยนต์ จึงเป็นการเหมาะสมอย่างยิ่งที่จะมีระบบจำลองการขับรถซึ่งสามารถต่อเข้ากับคอมพิวเตอร์ส่วนบุคคลเพื่อใช้สำหรับฝึกหัดในที่พักอาศัย
2. เนื่องจากการจำลองการขับรถปัจจุบันจะมุ่งเน้นไปในเรื่องของความบันเทิงในรูปแบบของเกมซึ่งได้ออกแบบให้การขับรถในแบบง่ายๆ เท่านั้น คือ ไม่มีระบบเกียร์ทั้งเกียร์ธรรมดาและเกียร์ออโตเมติก ทำให้ไม่ได้ฝึกฝนการขับรถยนต์ที่ถูกต้อง
3. การพัฒนาระบบเสมือนจริงในประเทศยังไม่ค่อยเป็นที่แพร่หลาย

1.4 เป้าหมายของโครงงาน

จำลองระบบการขับรถยนต์ออกมาในรูปแบบของโปรแกรม 3 มิติด้วยไดเร็กเอ็กซ์ มีการใช้วิดีโอคลิปสำหรับสอนการขับขี่และรับอินพุทจากอุปกรณ์ที่เตรียมขึ้น โดยจะสร้างโลกเสมือนจริงขึ้นมาเป็นสภาพของการสอบใบขับขี่ทีละระดับตามความยากง่าย ก่อนทดสอบจริงจะมีการสาธิตวิธีการขับที่ถูกต้องให้จากนั้นจึงเริ่มทดสอบโดยจะมีการตัดคะแนนถ้าขับชนหรือทำผิดกฎจราจร เมื่อผ่านก็จะได้ทดสอบในสถานีถัดไปและประมวลผลรวมว่าผ่านการทดสอบโดยรวมว่าสอบผ่านหรือไม่

1.5 ขอบเขตและข้อจำกัดของโครงงาน

โปรแกรมจะลองการขับรถเสมือนจริงนี้พัฒนาขึ้นเพื่อใช้ในการฝึกฝนการขับรถยนต์สำหรับการสอบใบอนุญาตขับขี่ ในส่วนของการแสดงผลมีการแสดงภาพ 3 มิติในลักษณะเดียวกับการขับรถในความเป็นจริง แต่ด้วยข้อจำกัดของการประมวลผลและการสร้างภาพ 3 มิติ ภาพที่แสดงยังอาจมีรายละเอียดไม่เพียงพอ นอกจากนั้นในส่วนของฮาร์ดแวร์ส่วนของการเก็บค่าข้อมูลยังคงจำกัดอยู่เฉพาะค่าข้อมูลที่สำคัญเท่านั้น เช่น การเลี้ยว การเร่ง การเบรก และ คลัช ในความเป็นจริงยังมีข้อมูลอีกหลายอย่างที่จะต้องนำมาประมวลผลร่วมด้วย

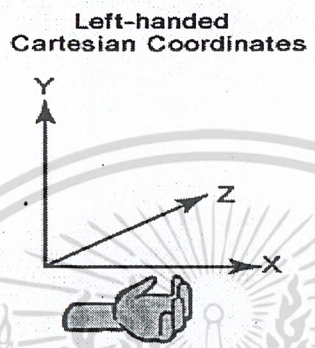
ในส่วนของฮาร์ดแวร์เป็นการพัฒนาเพิ่มเติมเฉพาะส่วนสำคัญที่หาไม่ได้ในท้องตลาด เนื่องจากอุปกรณ์ขับรถมาตรฐานมักมีให้เพียงพวงมาลัย คันเร่ง เบรก และเกียร์แบบชิฟ แต่การขับรถจริงตามวัตถุประสงค์ของโครงงานนี้ต้องการให้ขับเกียร์แบบธรรมดา ซึ่งจะต้องอาศัยอุปกรณ์เพิ่มอีกชิ้นหนึ่ง คือ คลัช ซึ่งจะต้องทำฮาร์ดแวร์เพิ่มในส่วนของคลัช และเกียร์

สุดท้ายโปรแกรมนี้ ยังขาดในส่วนของการแสดงผลการตอบสนองกับผู้ใช้ อาทิ การตอบสองทางแรง เช่น การสั่นสะเทือน ลม หรืออุณหภูมิ ข้อจำกัดต่างๆ ที่กล่าวมาข้างต้นจะได้นำมาพิจารณาและพัฒนาให้ครบถ้วนต่อไปในอนาคต

บทที่ 2
ทฤษฎี

2.1 การทำ Transformation ใน DirectX

World coordinate ใช้หลักLeft-handed Cartesian Coordinates จะมีลักษณะดังรูปด้านล่างคือ ประกอบด้วยแกน Xในแนวนอน , แกน Y ในแนวตั้ง และ แกน Z ทิศพุ่งเข้า



รูปที่ 2.1 แสดง World coordinate ของ DirectX ตามกฎมือซ้าย

จะใช้โครงสร้าง ของ D3DXMATRIX ซึ่งเป็น matrix แบบ 4*4 ดังรูป โดยแบ่งเป็น

- การทำ translation จากจุด P ใดๆ ไปยังตำแหน่ง p_x , p_y , p_z ใดๆ จะใช้ matrix

$$\mathbf{T}(\mathbf{p})=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ p_x & p_y & p_z & 1 \end{bmatrix}$$

- การทำ Rotation รอบแกน X , Y , Z เป็นมุม Ø ใดๆ จะใช้ matrix

$$\mathbf{X}(\vartheta)=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

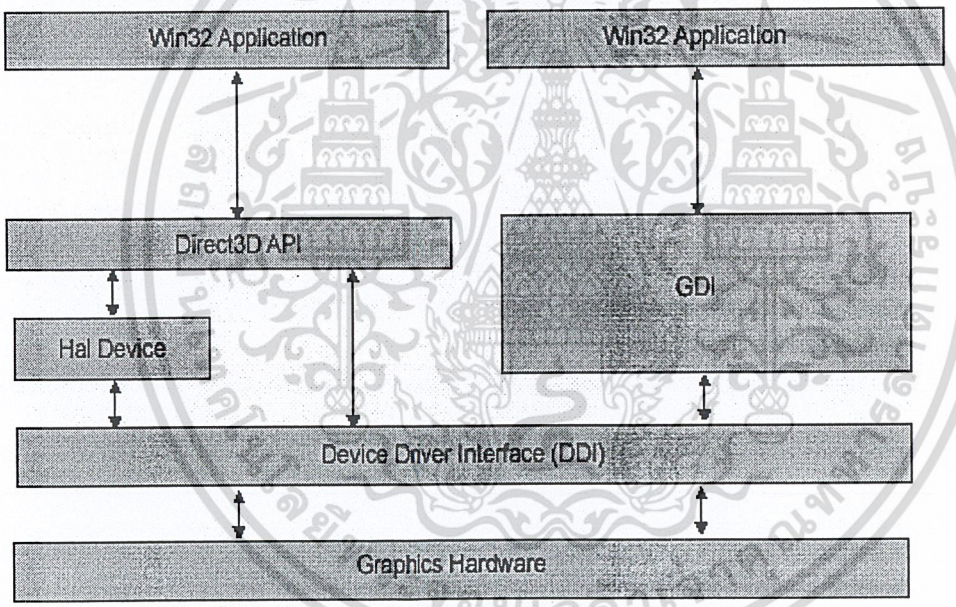
$$\mathbf{Y}(\vartheta)=\begin{bmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Z(\vartheta)=\begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- การทำ Scaling ด้วยอัตราส่วน ในแกน X , Y , Z ด้วย ค่า q_x , q_y , q_z ตามลำดับด้วย matrix

$$S(q)=\begin{bmatrix} q_x & 0 & 0 & 0 \\ 0 & q_y & 0 & 0 \\ 0 & 0 & q_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2.2 หลักการของ Direct3D



รูปที่ 2.2 Diagram ความสัมพันธ์ ของ Direct3D

จากรูปแสดงเป็น Diagram แสดงความสัมพันธ์ระหว่าง Direct3D , Window GDI , Hardware Abstraction Layer(HAL) และ Hardware แสดงผล

Direct3D API จะสามรถติดต่อเข้ากับ Device Driver ของการ์ด 3 มิติ ได้โดยตรงโดยจะทำงานร่วมกับ HAL เพื่อตรวจสอบการสนับสนุนงานด้านต่างๆ ของการ์ด 3 มิติ

2.2.1 การประมวลผล Vertex

ในการประมวลผลแต่ละ vertex ของรูป 3D แบ่งเป็น

1) Software Vertex Processing มีข้อดี จะสามารถใช้ได้กับทุก platform แต่ช้า

2) Hardware Vertex Processing จะเร็วกว่า Software Vertex Processing แต่ใช้ได้กับ Graphic Card ที่ Support เท่านั้น โดยที่ CPU ไม่ต้องทำการคำนวณการประมวลผลของ vertex

2.2.2 การใช้งาน Direct3D

1) ร้องขอการใช้งาน IDirect3D interface

2) ระบุชนิดของ Vertex Processing โดยการตรวจสอบว่าเครื่องที่ใช้นั้น Graphic Card สามารถทำ Hardware Vertex Processing ได้หรือไม่ ถ้าไม่ได้ก็ให้ใช้ Software ประมวลผลแทน

3) กำหนดลักษณะทั่วไปของ IDirect3DDevice9 object ผ่านทาง D3DPRESENT_PARAMETERS Structure

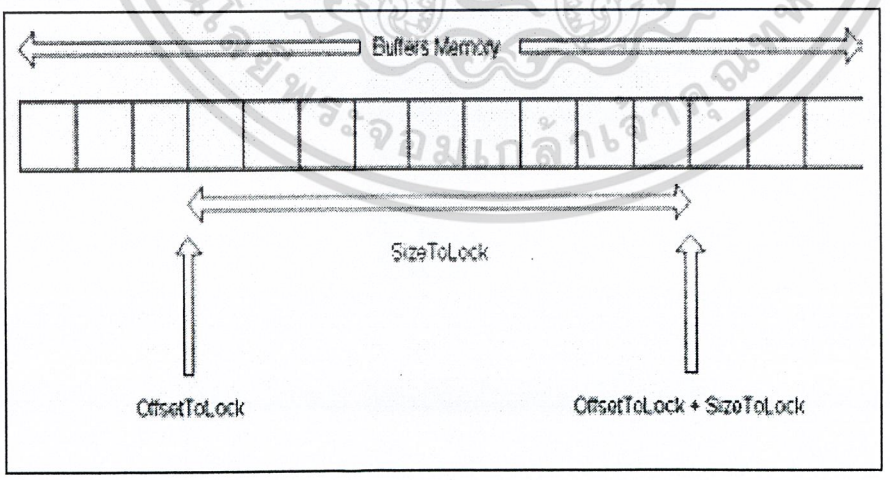
4) สร้าง IDirect3DDevice9 interface

2.2.3 Vertex / Index Buffer

จะสามารถเก็บลงใน Video Ram ได้ซึ่งจะทำให้การประมวลผลนั้นเร็วกว่าข้อมูลที่อยู่ใน System Ram โดย

- Vertex Buffer เป็นส่วนของ memory ที่ใช้เก็บ Vertex ต่างๆที่จะรวมกันเป็นรูปทรง
- Index Buffer เป็นส่วนของ memory ที่ใช้เก็บ Index เพื่อบอกว่า vertex ที่ถูกเก็บใน Vertex Buffer จะวางรวมกันอย่างไรให้เป็นแต่ละรูปสามเหลี่ยม

การใช้งาน Vertex / Index Buffer จะต้องทำการ Lock ไว้โดยกำหนด OffsetToLock เป็นจุดเริ่มต้น และ SizeToLock เป็นขนาดที่จะ Lock ดังรูป เมื่อใช้เสร็จแล้วจึงปลดล็อค



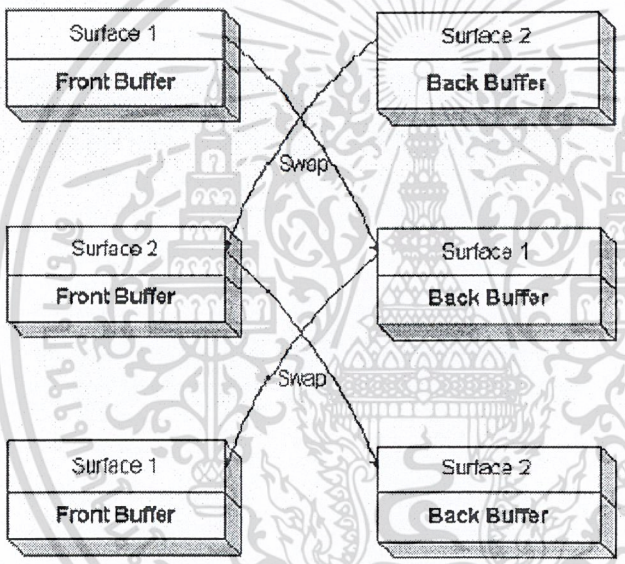
รูปที่ 2.3 แสดงการใช้งาน Vertex / Index buffer ด้วยการ Lock

2.2.4 Swap Chain

การทำงานของโปรแกรมจะมี front buffer ที่จะแสดงผลและ back buffer ที่ใช้จัดการกับรูปทรงต่าง ๆ ก่อนที่จะสลับไปแสดงผลด้วย function

HRESULT Present(CONST RECT **pSourceRect*,
CONST RECT **pDestRect*,
HWND *hDestWindowOverride*,
CONST RGNDATA **pDirtyRegion*);

แสดงการ Swap Front Buffer และ Back Buffer ดังในรูป เมื่อ Back Buffer swap ไปแสดงผลก็จะทำหน้าที่เป็น Front Buffer ต่อไป และ Front Buffer เดิมที่ถูก swap กลับมาก็ทำหน้าที่เป็น Back Buffer แทน จะทำให้ Performance ของการแสดงผลดีขึ้น หรืออาจใช้ Buffer มากกว่านี้ก็ได้

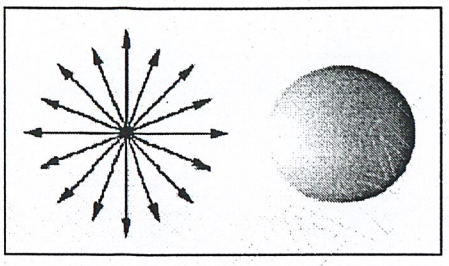


รูปที่ 2.4 แสดงการ swap buffer ในการประมวลผล

2.2.5 รูปแบบของแสง

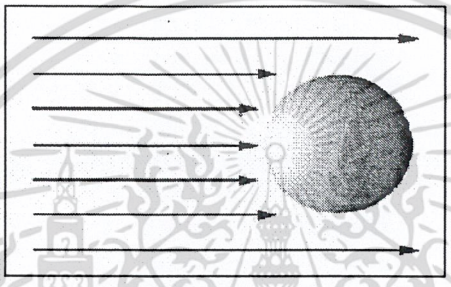
Direct3D จะสนับสนุน 3 รูปแบบของแหล่งกำเนิดแสง คือ

1) Point Light เป็นแหล่งกำเนิดแสงที่เกิดจากจุดใดจุดหนึ่งใน world space และจะกระจายแสงออกมาทุกทิศทุกทาง



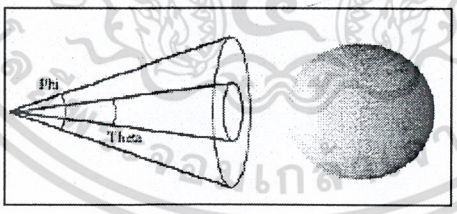
รูปที่ 2.5 รูป Point Light

2) Directional Light เป็นแหล่งกำเนิดแสงที่ไม่มีตำแหน่งของต้นกำเนิดแต่จะมีทิศทางของแสงที่ขนานกันไปตลอดในทิศทางเดียวกัน



รูปที่ 2.6 รูป Directional light

3) Spot Light แสงชนิดนี้คล้ายกับแสงของไฟฉาย คือจะมีตำแหน่งของต้นกำเนิดแสงชัดเจนและส่องแสงออกมาในลักษณะเป็นรูปกรวย ซึ่งจะมีคุณสมบัติประกอบด้วยมุม 2 มุม คือ มุมของรูปกรวยภายใน และ ภายนอก



รูปที่ 2.7 รูป Spot light

แสงที่ออกมาจากแหล่งกำเนิดแสงจะประกอบด้วย 3 ส่วนประกอบ คือ

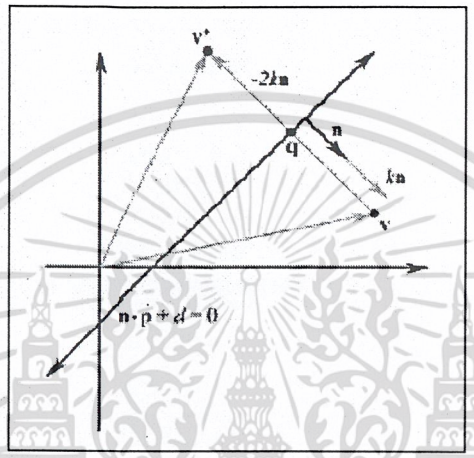
1) Ambient Light แสงชนิดนี้จะสะท้อนออกจากพื้นผิวและช่วยให้ฉากมีความสว่างมากขึ้นด้วย

2) Diffuse Light แสงชนิดนี้เมื่อเดินทางมากระทบกับพื้นผิวแล้วจะสะท้อนออกในทุกทิศทุกทาง

3) Specular Light แสงชนิดนี้เมื่อเดินทางมากระทบกับพื้นผิวแล้วจะสะท้อนออกในทิศทางเดียวกัน

2.2.6 หลักการของ Stencil Buffer

Stencil Buffer เป็น Off-screen buffer ที่ใช้ทำ special effect โดยจะมี resolution และ พิกเซล ที่ i*j ของ stencil buffer จะตรงกันกับพิกเซลล์ที่ i*j ของ back buffer และ depth buffer ใช้ mask บางส่วนของ back buffer ในแต่ละ pixel โดยในโครงงานนี้อาจใช้ทำกระจกที่สะท้อนภาพออกมาเพียงบางส่วน โดยทำให้ส่วน surface ที่เป็นกระจกทำหน้าที่สะท้อนภาพออกมา การสะท้อนทำได้โดยการคูณเมททริกที่ใช้ render ใน world space ด้วย เมททริกที่ได้จากการสะท้อนของจุดเทียบกับระนาบ ดังรูป เป็นการสะท้อนจุด V



รูปที่ 2.8 Stencil Buffer

จาก **q = v - kñ** โดยที่ k คือ ระยะทางที่สั้นที่สุดจาก V และ ระนาบ จะหาจุดที่สะท้อนของ v (v') บนระนาบ n.p + d = 0 ได้จาก

$$V' = v - 2k\tilde{n}$$
$$= v - 2(\tilde{n}.v + d)\tilde{n}$$
$$= v - 2[(\tilde{n}.v)\tilde{n} + d\tilde{n}]$$

และสามารถแทนค่าได้จากเมททริก

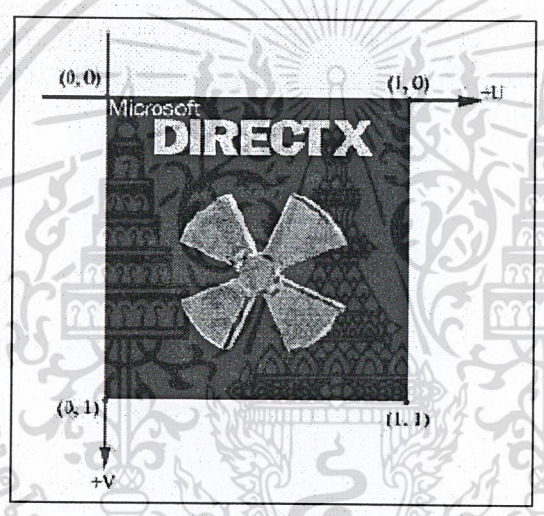
$$\mathbf{R} = \begin{bmatrix} -2n_x n_x + 1 & -2n_y n_x & -2n_z n_x & 0 \\ -2n_x n_y & -2n_y n_y + 1 & -2n_z n_y & 0 \\ -2n_x n_z & -2n_y n_z & -2n_z n_z + 1 & 0 \\ -2n_x d & -2n_y d & -2n_z d & 1 \end{bmatrix}$$

ตัวอย่างกรณีเมททริกพื้นฐานที่สะท้อนจาก 3 ระนาบบนโคออร์ดิเนต ได้แก่

$$\mathbf{R}_{yz} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{R}_{xz} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{R}_{xy} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

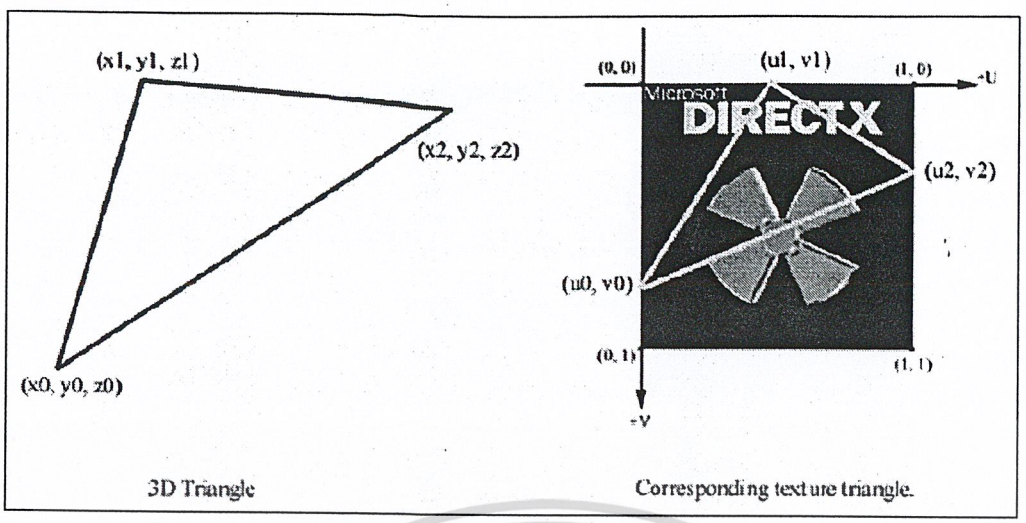
2.2.7 หลักการของ Texture mapping

เป็นเทคนิคการ map image data ไปบน polygon สามเหลี่ยม Direct3D จะใช้ texture coordinate ซึ่งประกอบด้วยแกน u ตามแนวนอน และแกน v ตามแนวตั้ง ตามรูป คู่ลำดับของ (u,v) จะเป็นการระบุตำแหน่งบนรูปภาพ โปรดสังเกตว่า แกน +v มีทิศทางชี้ลง



รูปที่ 2.9 รูปแสดง texture coordinate

Texture coordinate นั้นถูกทำให้อยู่ภายในช่วง [0,1] ซึ่งการแม็ปรูปภาพลงไปบนรูปสามเหลี่ยม ก็จำเป็นจะต้องกำหนด coordinate จอง texture ให้กับแต่ละจุดเวอร์เท็กบนรูปสามเหลี่ยม แสดงได้ดังรูป

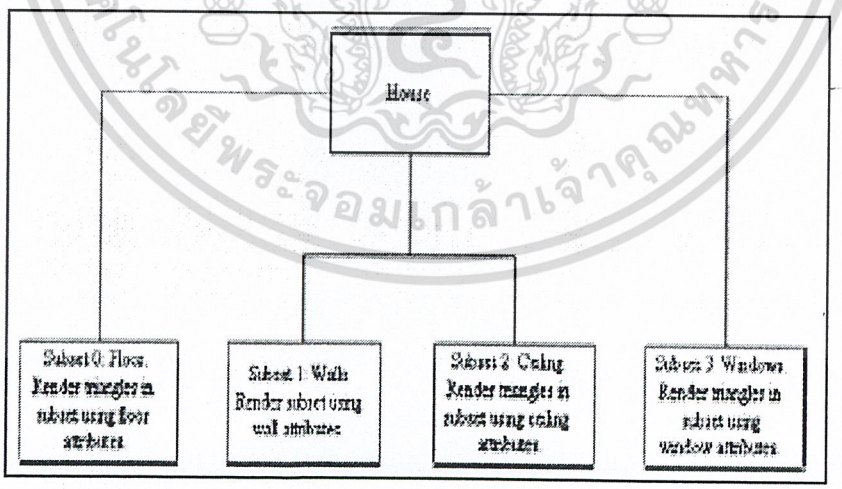


รูปที่ 2.10 การทำ Texture Mapping

รูปแสดงการทำ Texture Mapping ไปบนโพลีกอนสามเหลี่ยม ด้านซ้าย เป็นรูปสามเหลี่ยมในพิกัด 3 มิติ ด้านขวาเป็นการระบุพิกัดของรูปภาพให้กับรูปสามเหลี่ยมนั้นๆ

2.2.8 หลักการ Mesh

Mesh จะประกอบด้วย 1 subset ขึ้นไป โดย subset คือกลุ่มของโพลีกอนรูปสามเหลี่ยมที่ประกอบกันเป็น mesh ซึ่งสามารถ render ได้ด้วยคุณสมบัติเดียวกัน คุณสมบัติเหล่านี้ได้แก่ material , texture และ สเตจของการ render ตัวอย่างการแบ่ง mesh ออกเป็น subset แสดงได้ดังรูป เป็น mesh ของบ้าน ซึ่งแบ่งเป็น subset ของพื้น ผนัง เพดาน และ หน้าต่าง เป็นต้น

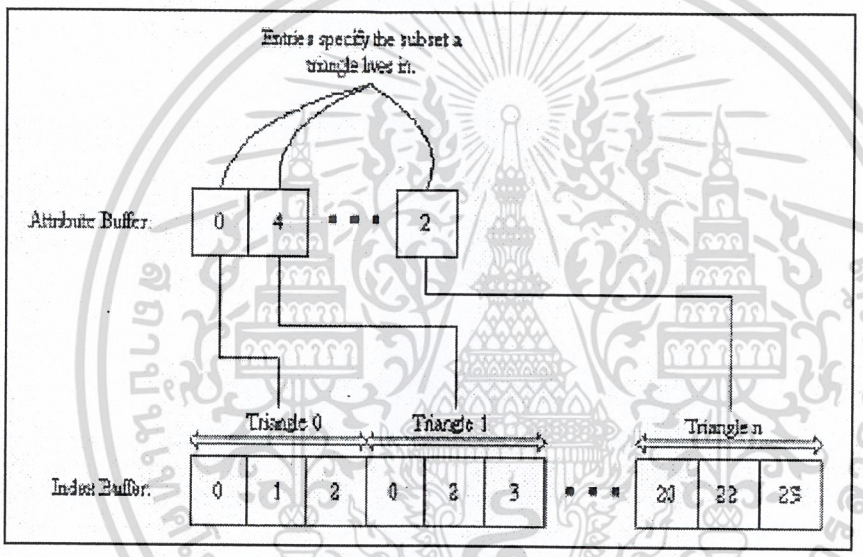


รูปที่ 2.11 แสดงตัวอย่างการแบ่ง mesh ของบ้านออกเป็น subset

เราจะระบุหมายเลขของแต่ละ subset ด้วยการกำหนดด้วยเลขจำนวนเต็มบวกที่แตกต่างกัน ซึ่งตัวเลขเหล่านี้จะมีขนาดที่พอจะเก็บในข้อมูลแบบ DWORD ดังเช่นในรูป แบ่งเป็นเลข 0,1,2,3

โพลีกอนสามเหลี่ยมแต่ละรูปใน mesh จะถูกระบุโดย attribute ID ซึ่งจะกำหนดว่าโพลีกอนนั้นเป็นสมาชิกของ subset ใดด้วย จากตัวอย่าง mesh ของบ้าน โพลีกอนสามเหลี่ยมที่ประกอบกันเป็นพื้นบ้านจะถูกระบุ attribute ID ด้วย 0 เพราะอยู่ใน subset 0 โพลีกอนสามเหลี่ยมที่ประกอบกันเป็นผนังจะมี attribute ID เป็น 1 เพราะอยู่ใน subset 1

Attribute ID ของแต่ละโพลีกอนรูปสามเหลี่ยมจะถูกเก็บอยู่ใน attribute buffer ซึ่งเป็นอาร์เรย์ของข้อมูลแบบ DWORD ซึ่งจะมีขนาดเท่ากับจำนวนโพลีกอนรูปสามเหลี่ยมทั้งหมดในแต่ละ mesh ซึ่งแต่ละสมาชิกจะสัมพันธ์กันโดยตรงกับข้อมูลของโพลีกอนรูปสามเหลี่ยมใน Index buffer ตามรูป



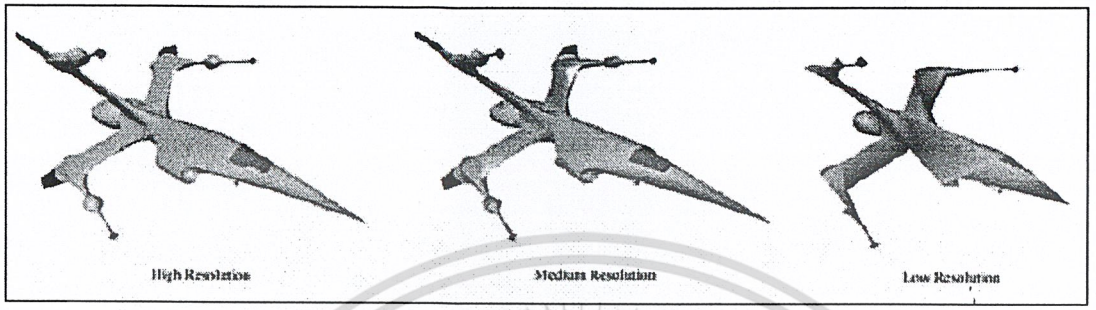
รูปที่ 2.12 แสดงความสัมพันธ์ระหว่างข้อมูลของโพลีกอนรูปสามเหลี่ยมใน Index buffer และแต่ละสมาชิกใน attribute buffer

2.2.9 หลักการของ X ไฟล์

การนำวัตถุ 3 มิติ ที่สร้างจากโปรแกรมกราฟฟิก 3 มิติ เช่น 3D Studio Max , Maya มาใช้กับ Direct X ต้องเปลี่ยนชนิดไฟล์เป็น X ไฟล์ จำเป็นต้องใช้โปรแกรมแปลงชนิดไฟล์หรือโหลด plugin เพิ่มเติมเพื่อเปลี่ยนชนิดไฟล์ของวัตถุ เป็น X ไฟล์ โดยมีหลายๆส่วนประกอบ เช่น geometry , material , animations others ซึ่ง DirectX สามารถโหลดมาใช้งานได้ โดยสร้าง ID3DXMesh interface ซึ่งเป็น child ของ ID3DXBaseMesh Interface เป็นตัวติดต่อ

2.2.10 หลักการของ Progressive Mesh

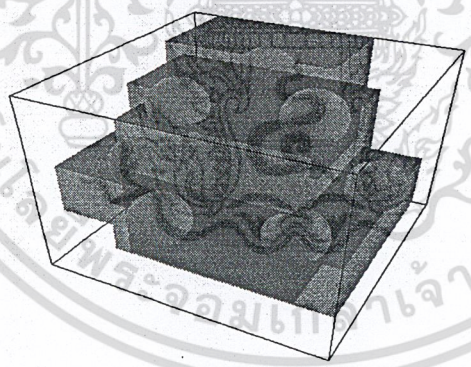
ใช้แทน ID3DXMesh interface ช่วยปรับเปลี่ยนรายละเอียดของ mesh ขึ้นอยู่กับระยะทางจากกล้อง LOD(Levels Of Detail) เพื่อเพิ่มประสิทธิภาพในการแสดงผล โดยนำเอา ID3DXMesh มา generate เป็น ID3DXPMesh Interface ตัวอย่างการใช้ Progressive Mesh



รูปที่ 2.13 ตัวอย่างการใช้ Progressive Mesh แบบ High , Medium , Low Resolution

2.2.11 หลักการของ Bounding Volume

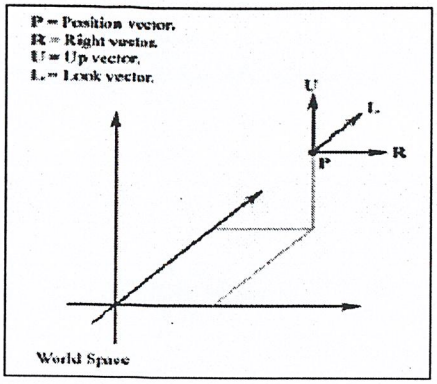
ใช้เพื่อเพิ่มความเร็วในการทดสอบการชน หรือเพื่อใช้ในการ clipping โดยการ check collision ทำโดยทดสอบจาก Ray / Plane intersection โดยจะคิดเฉพาะที่ bounding volume เท่านั้น ตัวอย่าง



รูปที่ 2.14 แสดงการใช้ bounding box กับ object

2.2.12 หลักการของ Camera

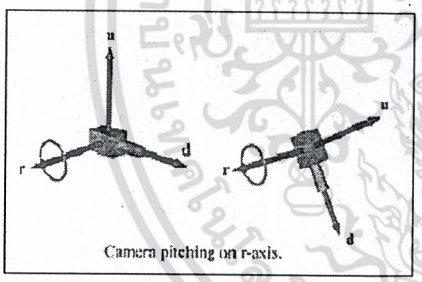
Component ของกล้องจะประกอบด้วยเวกเตอร์ ดังรูป



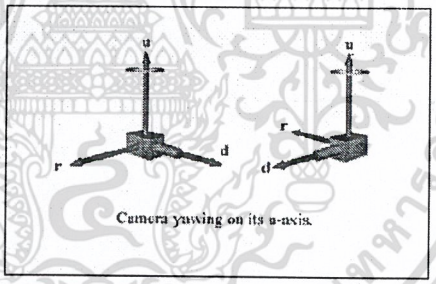
รูปที่ 2.15 แสดง vector ส่วนประกอบของกล้อง

ด้วย 4 เวกเตอร์จะสามารถทำ operation กับกล้องได้ 6 อย่าง(six degree of freedom)

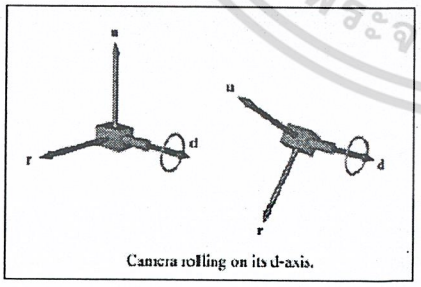
- หมุนรอบ right vector (pitch)
- หมุนรอบ up vector (yaw)
- หมุนรอบ look vector (roll)
- เคลื่อนที่ไปบน right vector
- เคลื่อนที่ไปบน up vector
- เคลื่อนที่ไปบน look vector



รูปที่ 2.16 แสดงการหมุนกล้องแบบ pitch



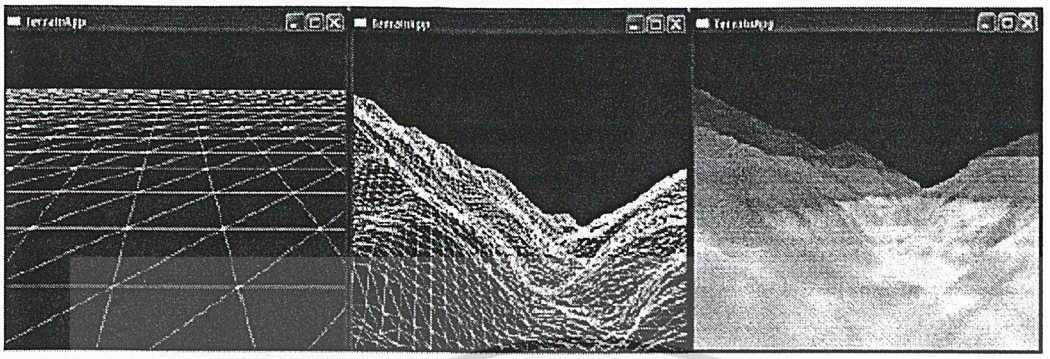
รูปที่ 2.17 แบบ yaw



รูปที่ 2.18 แบบ roll

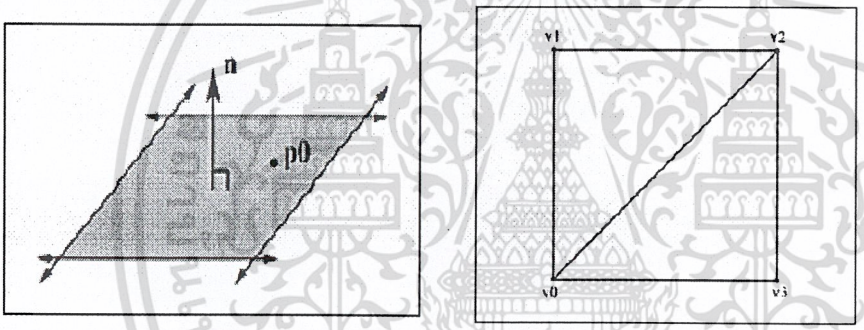
2.2.13 หลักการของ Terrain

เป็น triangle grid ที่เก็บ vertex และ index data ของแต่ละ vertex และต้องมี array เก็บค่าความสูงของแต่ละ triangle เพื่อให้มีรูปร่างที่เรียบ



รูปที่ 2.19 แสดงการสร้าง terrain ด้วย triangle grid

ในเบื้องต้นจะทำพื้นโดย การสร้าง plane ในลักษณะ ดังรูป คือจะประกอบด้วยรูปสามเหลี่ยม 2 รูปต่อกันแล้วนำ Texture ไปติด



รูปที่ 2.20 แสดงการสร้าง terrain ด้วย primitive triangle 2 รูป

2.3 หลักการของ DirectInput

Microsoft DirectInput เป็น API(Application Programming Interface) แบบหนึ่งที่ใช้สำหรับอุปกรณ์ เช่น เมาส์ คีย์บอร์ด จอยสติก และอุปกรณ์ควบคุมเกมส์แบบอื่นๆ รวมถึงอุปกรณ์ที่สามารถสร้าง force-feedback ได้ด้วย

DirectInput ให้การเข้าถึงข้อมูลจากอุปกรณ์ที่รวดเร็วด้วยการติดต่อกับhardware device driverโดยตรง โดยไม่ต้องผ่านการตรวจจับ message ของ window แอพพลิเคชั่นสามารถดึงข้อมูลจากอุปกรณ์ได้แม้ว่าจะอยู่ไม่ได้ถูกโฟกัส ด้วยการทำ action mapping แอพพลิเคชั่นสามารถรับข้อมูลได้โดยไม่จำเป็นต้องรู้ว่าอุปกรณ์ที่ใช้นั้นเป็นชนิดใด แต่ DirectInput ไม่ได้มีความสามารถในการรับข้อมูลตัวอักษรจากคีย์บอร์ดและการรับตำแหน่งของเมาส์

ขั้นตอนอย่างง่ายๆ สำหรับแอพพลิเคชั่นในการตอบสนองกับข้อมูลที่ถูกสร้างขึ้นมาในแต่ละชนิดอุปกรณ์

1. สร้าง DirectInput object เพื่อที่จะใช้งาน method ภายใน

2. ระบุอุปกรณ์ที่ใช้ส่งข้อมูล ขั้นตอนนี้อาจไม่จำเป็นถ้าใช้เพียง เมาส์ หรือ คีย์บอร์ด ในการตรวจสอบว่ามีอุปกรณ์ใดสามารถใช้งานได้บ้างในขณะนั้นก็สามารถให้ DirectInput หาให้ได้และเมื่อใดที่ DirectInput เจออุปกรณ์ที่ตรงกับข้อมูลที่ตั้งไว้ก็จะสามารถใช้ได้โดยจะได้ ID กำกับแต่ละอินพุตที่ไม่ซ้ำกัน

3. สร้าง DirectInputDevice object สำหรับแต่ละอุปกรณ์ที่ต้องการจะใช้โดยการใช้ ID ที่ได้รับตอนค้นหา สำหรับ เมาส์หรือคีบอร์ด สามารถใช้ Standard globally Unique Identifier (GUID)

4. ตั้งค่าอุปกรณ์ โดยเริ่มต้นด้วยการตั้งค่าการแบ่งการใช้ข้อมูลจากอุปกรณ์สำหรับแต่ละแอพพลิเคชั่น กำหนดรูปแบบข้อมูลเช่นเป็นแบบปุ่มกด หรือ แบบแกน และถ้าต้องการใช้ buffered data ก็จะต้องระบุขนาดของ buffer ไว้ด้วย ใขขั้นตอนนี้สามารถที่จะดึงข้อมูลของตัวอุปกรณ์และปรับปรุงให้เข้ากับแอพพลิเคชั่นได้ เช่นการกำหนดช่วงการรับข้อมูลของแต่ละแกนของจอยสติก

5. เข้าควบคุมอุปกรณ์ ในขั้นตอนนี้เป็นการบอก DirectInput ว่าพร้อมที่จะรับข้อมูลแล้ว

6. ขั้นตอนการรับข้อมูล ผ่านลูปการตรวจจับ message โดยรับสเตจปัจจุบันของอุปกรณ์หรือเหตุการณ์การใช้อุปกรณ์ล่าสุด และสามารถให้ DirectInput แจ้งเตือนเมื่อมีเหตุการณ์ใหม่เข้ามาได้

7. ใช้งานข้อมูล แอพพลิเคชั่นสามารถใช้งานได้ทั้ง สเตจของปุ่มกดและแกน หรือ เหตุการณ์ที่เกิด เช่น กด หรือ ปล่อย

8. ปิดการใช้งาน DirectInput แต่ก่อนจะออกต้องให้แอพพลิเคชั่นยกเลิกการควบคุมอุปกรณ์ทุกตัวเสียก่อน

2.4 หลักการของ DirectMusic

DirectMusic ทำอะไรได้มากกว่าการเล่นเสียงแบบธรรมดามากมาย เพราะมันมีระบบที่สมบูรณ์แบบสำหรับนำ Dynamic soundtrack มาใช้งาน ซึ่งจะมีข้อได้เปรียบตรงที่ใช้ Hardware เป็นตัวเร่งความเร็ว (Hardware acceleration), สามารถ Download เสียงได้ (Downloadable Sounds), ใช้ DirectX Media Objects (DMOs) และการวาง Effect 3 มิติขั้นสูง

การนำ DirectMusic มาใช้งานกับ Application จะทำให้สามารถทำได้ดังนี้โหลด และ เล่น เสียงจากไฟล์นามสกุล MIDI, WAV หรือ ไฟล์จาก DirecMusic Producer ได้

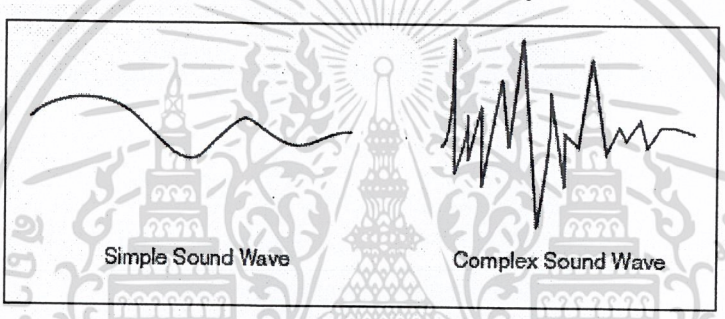
- สามารถเล่นเสียงหลายๆ เสียงได้ในเวลาเดียวกัน
- จัดการกับเวลาที่ต้องการเล่นเสียงได้อย่างแม่นยำ
- สามารถตั้งรูปแบบการเปลี่ยนแปลงทำนองเสียง, เพิ่มเติม และแก้ไขรูปแบบ MIDI แบบอื่นๆ ได้
- ใช้เสียงที่สามารถดาวน์โหลดได้ ซึ่งการนำ DLS มาใช้นั้นจะทำให้ Application มั่นใจได้ว่า เสียงที่เล่นจะเป็นเสียงเดียวกันบนเครื่องคอมพิวเตอร์ทุกเครื่อง นอกจากนี้ Application สามารถเล่นเสียงเครื่องดนตรีได้ไม่จำกัดชนิด และสร้างเสียงที่มีลักษณะจำเพาะสำหรับบางคนได้
- เล่นเสียงในสภาวะแวดล้อมแบบ 3 มิติ

- ง่ายในการปรับตั้งค่าระดับเสียง, เสียงสะท้อน และ Effect เสียงอื่นๆ
- การที่ใช้งานมากกว่า 16 ช่อง MIDI ดังนั้นจึงเป็นไปได้สำหรับ DirectMusic ที่จะเล่นเสียงพร้อมๆ กันได้มากไม่จำกัด ซึ่งแล้วแต่ขีดจำกัดของตัวสังเคราะห์เสียง
- เล่นเสียงเป็นบางส่วน บน audiopath อื่นๆ ได้ ดังนั้น Effect หรือคุณสมบัติพิเศษของเสียงจะนำไปใช้ได้กับเสียงแต่ละเสียง
- ดักจับ (Capture) MIDI จากข้อมูล หรือ stream มาจากพอร์ตหนึ่งไปยังอีกพอร์ตหนึ่งก็ได้

2.4.1 การเล่นเสียงด้วย DirectX Audio

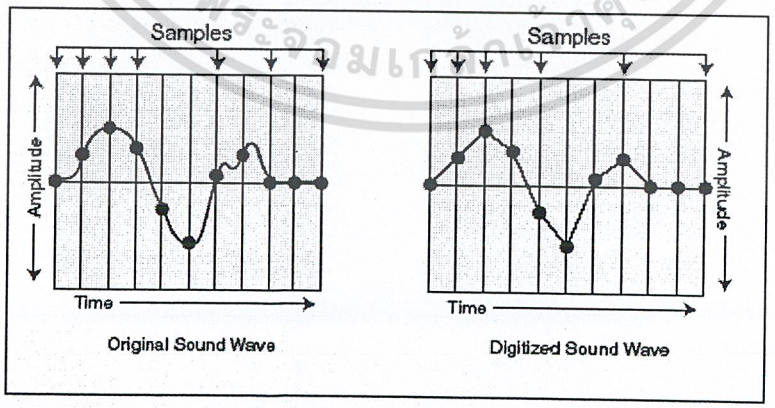
2.4.1.1 พื้นฐาน

เสียงแบ่งออกเป็นสองประเภทหลักๆ ได้แก่ เสียงธรรมดา และเสียงที่ซับซ้อน ซึ่งเสียงธรรมดานี้มักจะเป็นเสียงจากธรรมชาติทั่วๆ ไปซึ่งจะมีรูปคลื่นราบรื่น คล้ายกับ Sine Wave และเสียงที่ซับซ้อน ซึ่งไม่มีรูปแบบของคลื่นที่แน่นอน รูปแบบคลื่นแสดงดังรูป



รูปที่ 2.21 รูปเปรียบเทียบคลื่นเสียงธรรมชาติ และเสียงที่ซับซ้อน

การจะบันทึกเสียงที่เกิดขึ้น เข้าไปใช้ในคอมพิวเตอร์ต้องมีการบันทึกเสียงลงไป การบันทึกเสียงด้วยระบบ Digital นั้นจะต้องเริ่มจากการ Sampling เสียงก่อน โดยจะทำการ Sampling ได้ตั้งแต่ 8000 ถึง 44,100 ครั้ง / วินาที เพื่อให้ได้รูปแบบของสัญญาณเสียง



รูปที่ 2.22 แสดงการเปรียบเทียบคลื่นเสียงต้นฉบับ กับเสียงแบบ Digital

หลังจากทำการ Sampling แล้วก็จะนำค่าที่ได้ไปจัดเก็บ โดยจะต้องทำการ Quantization ก่อน โดยจะแบ่งระดับของสัญญาณออกเป็นหลายๆ ระดับ ที่นิยมใช้คือ 8-bit (แบ่งได้ 256 ระดับ) และ 16-bit (แบ่งได้ 65,535 ระดับ)

2.4.1.2 MIDI

ไฟล์ชนิด MIDI (นามสกุล .Mid) เป็นมาตรฐานไฟล์ที่ใช้สำหรับจัดเก็บดนตรี ซึ่งไฟล์ Midi ประกอบไปด้วย message หรือ ทิศทาง ที่มีความจำเป็นสำหรับเล่นเพลง ค่าของทิศทางเหล่านี้จะบอก Hardware เสียงให้ทราบว่าจะทำอะไร ,เมื่อไร , ทำอย่างไร และจะเปลี่ยนแปลงทำนอง หรือเปลี่ยนเสียงเครื่องดนตรีได้อย่างไร

Midi สามารถเล่นเสียงเครื่องดนตรีมาตรฐานได้ 128 รูปแบบ ทำให้สามารถจะสร้างทำนองของเพลงด้วยเครื่องดนตรีใดๆ ก็ได้โดยใช้ข้อมูลเสียงชุดเดิม เพียงแค่เปลี่ยนค่าตัวแปรหมายเลขเครื่องดนตรีเท่านั้น นอกจากนั้นเพลงจะถูกแบ่งออกเป็น Track ซึ่งแต่ละ Track จะบรรจุโน้ตเพลงที่จะเล่นลงไปในแต่ละชนิดเครื่องดนตรี ซึ่งโดยปกติจะสามารถมี 128 Track ในไฟล์เดียวได้ และความยาวไม่จำกัดอีกด้วย

2.4.1.3 DirectX Audio

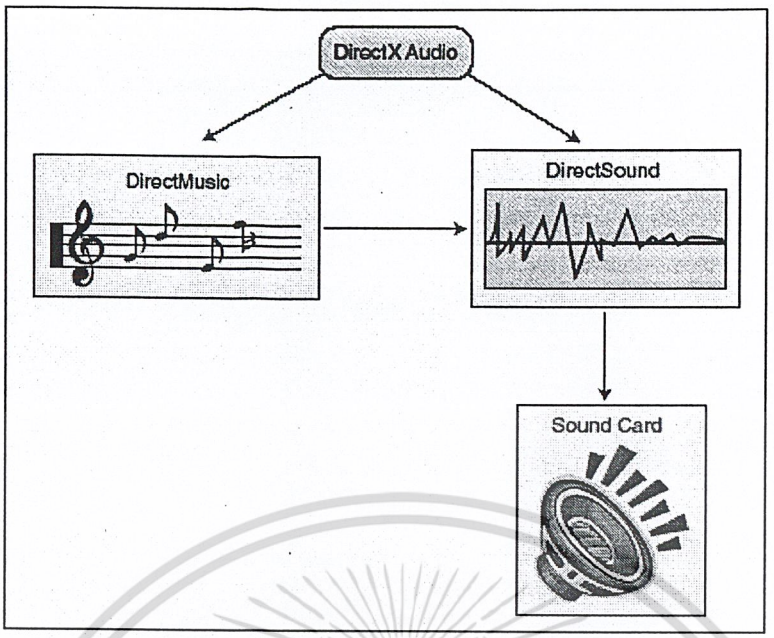
DirectX Audio ประกอบด้วยส่วนประกอบหลักๆ สองส่วน ซึ่งก็คือ DirectSound และ DirectMusic ซึ่ง DirectMusic เป็นส่วนที่ได้รับการพัฒนาเพิ่มขึ้นมากมาย ถ้าเทียบกับ DirectSound ในเวอร์ชันก่อน

DirectSound เป็นส่วนประกอบหลักที่นิยมใช้ในการเล่นเสียง และ DirectMusic จะใช้ในการเล่นเสียงทุกรูปแบบ (format) ไม่ว่าจะเป็น WAVE, MIDI หรืออื่นๆ และยังเรียกใช้ DirectSound ในการสร้างสัญญาณเสียงใหม่ได้ด้วย เช่น กรณี MIDI ไฟล์ จะสามารถบันทึกการเล่นดนตรีแบบแยกเครื่องได้ ขณะที่ทำการเล่นแบบปกติ

2.4.1.4 การนำ DirectSound มาใช้งาน

เริ่มแรกต้องสร้าง COM Object ขึ้นมาติดต่อกับ Sound Hardware ก่อน ซึ่ง Object นี้จะทำให้สามารถสร้าง Sound Buffer ขึ้นมาโดยเฉพาะได้ เรียกว่า “Secondary Sound Buffer” ใช้สำหรับเก็บข้อมูลเสียง ข้อมูลที่จัดเก็บเอาไว้ใน Sound Buffer จะถูกผสมเข้าด้วยกันใน Mixing Buffer หรือเรียกว่า “Primary Sound Buffer” และเล่นเสียงออกมาได้ในทุกๆ รูปแบบที่ต้องการ รูปแบบที่กล่าวนี้สามารถเปลี่ยนแปลงได้ทั้งจำนวน Channel , ความถี่ และจำนวน Bit ที่ใช้ในการ Sampling

สามารถแก้ไข Channel เสียงได้ เพื่อเล่นเสียงที่ความถี่แตกต่างกัน หรือระดับเสียงที่แตกต่างกัน (โดยการเปลี่ยนแปลง Pitch) ปรับระดับเสียงระหว่างการเล่น หรือแม้กระทั่งเล่นเสียงวนไปเรื่อยๆ ยิ่งไปกว่านั้นสามารถจำลองเสียงในรูปแบบ 3 มิติ ซึ่งเหมือนเสียงที่เกิดขึ้นรอบๆ ตัว

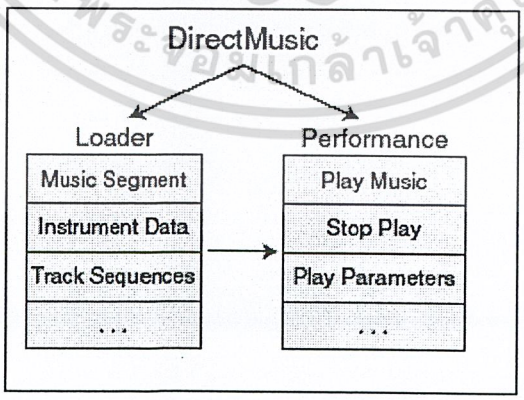


รูปที่ 2.23 รูปแสดงส่วนประกอบหลักของ DirectX Audio

ส่วนประกอบหลักของ Direct X Audio มีสองส่วนคือ DirectSound และ DirectMusic ซึ่งทำงานแยกส่วนกัน แต่ DirectMusic สามารถเรียกใช้ DirectSound เพื่อให้สร้างเสียงดนตรี และเล่นเสียงผ่าน Sound Card ให้

2.4.1.5 DirectMusic

ขั้นตอนการนำ DirectMusic มาใช้งานเริ่มจากสร้าง Main Object ขึ้นมา (เรียกว่า Performance Object) แทนระบบเสียงทั้งหมด ถัดไปสร้าง Object ที่เรียกว่า Loader Object ซึ่งมีหน้าที่โหลดไฟล์สำคัญที่เกี่ยวกับเสียงทั้งหมดขึ้นมา ซึ่งทั้งสอง Object นี้จะสัมพันธ์กันตามรูป



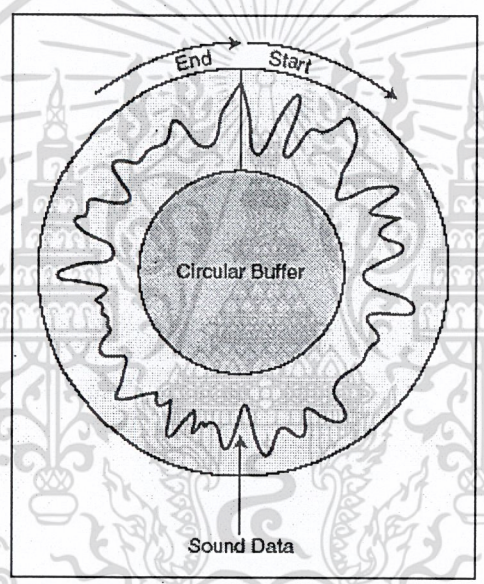
รูปที่ 2.24 แสดงการทำงานของ DirectMusic

สุดท้ายทำการโหลดส่วนของเพลงขึ้นมาบน Segment Object ซึ่งสามารถโหลดและเล่นได้ทีละหลายส่วน เพื่อที่จะสร้างเสียงที่มีความซับซ้อน และมีความยาวมากขึ้น

2.4.1.6 Primary Sound Buffer

สิ่งที่ต้องทำก็คือ สั่งให้ Primary Buffer เริ่มเล่นเสียง ถึงแม้ว่าจะไม่มีเสียงใดที่จะเล่นในตอนนั้นก็ตาม ทางที่ดีควรเริ่มให้ Buffer ทำงานตั้งแต่แรก และทำงานไปจนสุดั้งระบบการเล่นเสียงทั้งหมด การให้ Buffer เริ่มทำงานพร้อมๆ กับ Application จะช่วยประหยัดเวลาในการประมวลผลในการทำการเปิด-ปิดการใช้งาน

เนื่องจากเนื้อที่ของ Memory มีขนาดที่จำกัดได้ โดยเฉพาะอย่างยิ่ง Hardware ซึ่ง Data Buffer ที่นำมาใช้งานอาจเป็นขนาดเท่าใดก็ได้ (อาจเล็กเพียงไม่กี่พัน Byte) ดังนั้นจึงเป็นเหตุผลที่ทำไม Primary และ Secondary Sound Buffer จึงใช้ Buffer ที่มีลักษณะเป็นวงกลม



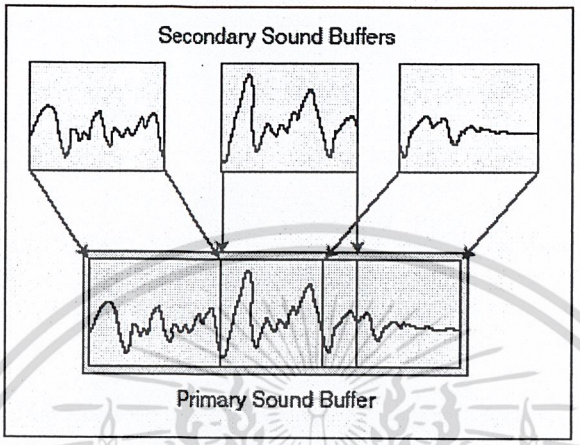
รูปที่ 2.25 แสดง Primary Sound Buffer ที่มีลักษณะเป็น Circular

แม้ว่า Data Buffer จะมีลักษณะเป็น Array 1 มิติ ซึ่งจะตามลบส่วนแรกไปเรื่อยๆ ซึ่งช่วยทำให้ประหยัด Memory ได้มาก เมื่อมีการเล่นเสียง จะมีการผสมกันใน Primary Sound Buffer ในวงกลมดังรูปด้านบน เมื่อสิ้นสุด Buffer เสียงจะเริ่มต้นวนกลับมาที่ตำแหน่งเริ่มต้นของ Buffer อีกครั้งหนึ่ง และขับเสียงออกไปอย่างต่อเนื่อง

หมายเหตุ : Primary Sound Buffer จะไม่หยุดเล่นถ้า Secondary Buffer ทุกตัวยังไม่หยุดทำงาน เมื่อไรก็ตามที่ Secondary Buffer ทำงาน ส่วน Primary จะถูกเรียกขึ้นมาทำงานโดยอัตโนมัติด้วย

2.4.1.7 Secondary Sound Buffer

การสร้าง Secondary Sound Buffer ขึ้นมาเพื่อเก็บเสียงที่แท้จริงที่ต้องการเล่น ซึ่งไม่มีการจำกัดจำนวนเสียงที่ Secondary สามารถเก็บเอาไว้ได้ในเวลาขณะหนึ่ง และด้วยความสามารถของ DirectSound จะช่วยให้เล่นเสียงได้ทุกอย่างที่ต้องการ



รูปที่ 2.26 แสดงการทำงานร่วมกันของ Primary และ Secondary Sound Buffer

การทำงานของ Secondary Sound Buffer คล้ายกับที่แสดงไว้ตามรูปข้างบน ข้อมูลเหล่านี้จะถูกผสมกันเข้าไปใน Primary Sound Buffer ตราบเท่าที่เสียงยังคงเล่นต่อไป ดังนั้นการสร้างเสียงแรกและเสียงถัดไปลงไปในตำแหน่งเดียวกันของ Primary Sound Buffer จะทำให้เกิดเสียงสองเสียงขึ้นในเวลาเดียวกัน

2.5หลักการของ DirectShow

Microsoft DirectShow เป็นสถาปัตยกรรมสำหรับข้อมูลที่เป็น stream ใน window platform สามารถใช้บันทึกและเล่นข้อมูลมัลติมีเดียด้วยคุณภาพที่สูง DirectShow เป็นการรวมกันของเทคโนโลยี DirectX หลายๆอย่าง จะสามารถตรวจหาอุปกรณ์เร่งความเร็วออดิโอและวิดีโอและนำมาใช้งานได้อย่างอัตโนมัติ

DirectShow มามารถใช้งาน เล่นมีเดียร์ เปลี่ยนฟอร์แมท และงานบันทึกได้อย่างง่ายๆ และผู้ใช้ยังสามารถสร้างคอมโพเนนท์ของ DirectShow เองเพื่อสนับสนุนรูปแบบใหม่ๆได้ด้วย

รูปแบบไฟล์ที่ DirectShow สามารถสนับสนุน ชนิดของไฟล์

- Windows Media™ Audio (WMA)*
- Windows Media™ Video (WMV)*
- Advanced Systems Format (ASF)*
- Motion Picture Experts Group (MPEG)
- Audio-Video Interleaved (AVI)

- QuickTime (version 2 and lower)
- WAV
- AIFF
- AU
- SND
- MIDI

2.6 ทฤษฎีความเร็ว-ความเร่ง

ความเร่ง คือ อัตราส่วนของความแตกต่างของความเร็วใน 1 หน่วยเวลา ดังแสดงตามสมการ

$$a = \frac{\Delta v}{\Delta t} = \frac{v - v_0}{\Delta t} \quad \text{..... (2.1)}$$

จัดรูปแบบสมการใหม่

$$v = v_0 + a\Delta t \quad \text{.....(2.2)}$$

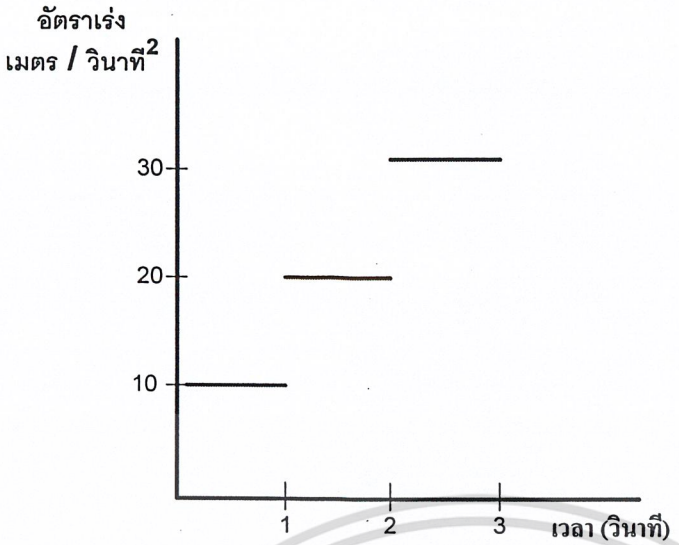
a = acceleration (อัตราเร่ง)

v = velocity(อัตราเร็ว)

t = time(เวลา)

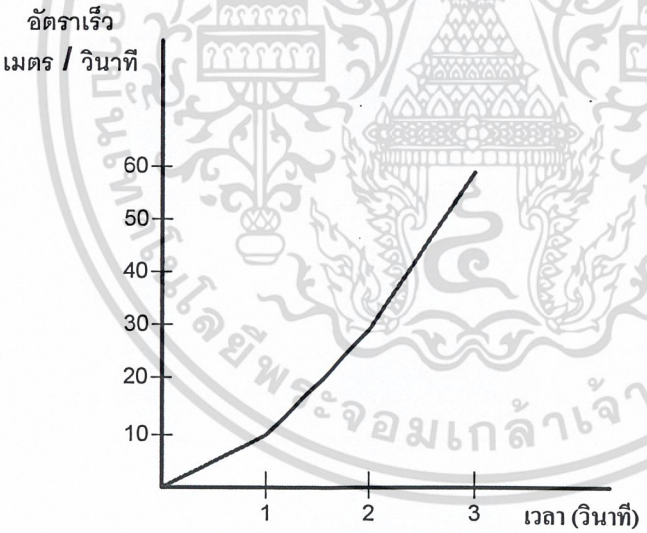
v_0 = อัตราเร็วต้น

ตัวอย่างกราฟแสดงอัตราเร่งเป็นดังรูป



รูปที่ 2.27 ตัวอย่างกราฟอัตราเร่ง เทียบกับเวลา

กราฟนี้แสดงอัตราเร่งคงที่ กล่าวคือ อัตราเร่งจะไม่มีความชันจะ แต่จะมีค่าคงที่เป็นช่วงเวลาหนึ่งๆ ตั้งแต่วินาทีที่ 0 ถึงวินาทีที่ 1 จะมีอัตราเร่งคงที่ที่ 10 เมตร / วินาที^{2} และ วินาทีที่ 1 ถึง 2 จะมีอัตราเร่งคงที่ที่ 20 เมตร / วินาที^{2} และวินาทีที่ 2 ถึง 3 จะมีอัตราเร่งคงที่ๆ 30 เมตร / วินาที^{2} ตามลำดับ อัตราเร่งคงที่นี้ เมื่อนำไปคำนวณจากสมการที่ 2.2 แล้วจะทำให้ได้กราฟอัตราเร็วเป็นดังนี้



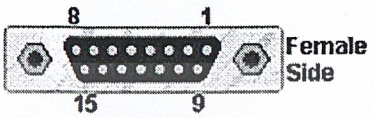
รูปที่ 2.28 ตัวอย่างกราฟอัตราเร็ว เทียบกับเวลา

อัตราเร็วที่เกิดขึ้นในกราฟ คำนวณได้จากค่าอัตราเร่ง โดยนำไปเข้าสมการ (2.2) ซึ่งเมื่อ เวลาเริ่มต้นที่ 0 วินาที รถยนต์มีอัตราเร่งที่ 10 เมตร / วินาที^{2} เวลาผ่านไป 1 วินาที จากความเร็วเริ่มต้นที่ 0 จึงกลายเป็น 10 เมตร / วินาที และหลังจากนั้นเมื่อเร่งที่อัตราเร่ง 20 เมตร / วินาที^{2} ต่อไปอีกเป็นเวลา 1 วินาที ความเร็วจึงเพิ่มขึ้นเป็น 30 เมตร / วินาที ช่วงสุดท้ายมีอัตราเร่งที่ 30 เมตร / วินาที^{2} เร่งไปอีกเป็นเวลา 1 วินาที ก็จะได้ความเร็ว

เพิ่มขึ้นเป็น 60 เมตร / วินาที ซึ่งทั้งหมดนี้แสดงให้เห็นว่า สามารถคำนวณอัตราเร็วได้จากการหาอัตราเร่งเป็นช่วงๆ และนำมาเข้าสมการที่ (2.2)

2.7 การ เชื่อมต่อผ่าน Game Port

Game Port มีขาทั้งหมด 15 ขา เป็นขาสำหรับรับสัญญาณจาก Switch 4 ขา, สำหรับรับสัญญาณจาก Potentiometer อีก 4 ขา นอกจากนี้เป็น +5V และ Ground ดังแสดง



รูปที่ 2.29 พอร์ตมาตรฐาน DB-15

Pin NO.	แบบธรรมดา	แบบ MIDI
1	+5 V	+5 V
2	Button 0	Button 0
3	J1X Axis	J1X Axis
4	Ground	Ground
5	Ground	Ground
6	J1Y Axis	J1Y Axis
7	Button 1	Button 1
8	+5 V	+5 V
9	+5 V	+5 V
10	Button 2	Button 2
11	J2X Axis	J2X Axis
12	** Ground **	Midi RxD
13	J2Y Axis	J2Y Axis
14	Button 3	Button 3
15	** +5 V **	Midi TxD

ตารางที่ 2.1 แสดงขาสัญญาณของ Game Port

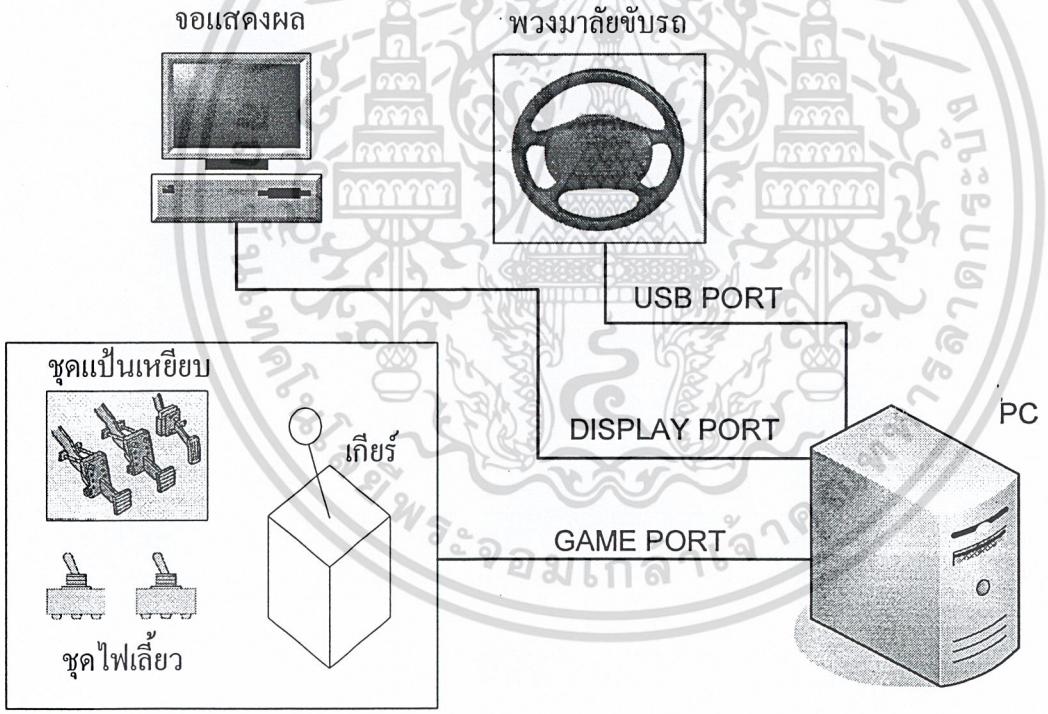
โดยปกติจะนิยมใช้การเชื่อมต่อผ่าน Game Port ด้วยแบบธรรมดาเป็นส่วนมาก และจะนำมาใช้ในโครงงานนี้ด้วยเช่นกัน โดยทั่วไปจะปล่อยขา 12 และ 15 ว่างเอาไว้ (ไม่นำไปต่อพ่วงกับอุปกรณ์ใด ๆ) นอกเหนือจากนี้การเชื่อมต่ออีกแบบหนึ่งคือ แบบที่ใช้ MIDI ได้ด้วย โดยให้อุปกรณ์ทั้งสองนี้ รับส่งข้อมูลเสียงชนิด Midi กันไปยัง Sound Card เพื่อสร้างเสียงประกอบ

บทที่ 3
การออกแบบ

3.1 การเชื่อมต่อ Hardware เข้ากับคอมพิวเตอร์

อุปกรณ์ทั้งหมดที่ใช้ในโครงงาน "ระบบการจำลองการขับรถเสมือนจริง" ที่ได้มีการเชื่อมต่อกับคอมพิวเตอร์ประกอบไปด้วย

- จอแสดงผล ใช้สำหรับแสดงผลมุมมองของผู้ขับขี่
- พวงมาลัยขับรถ ใช้บังคับทิศทางรถ
- ชุดแป้นเหยียบ ซึ่งมีแป้นเหยียบจำนวน 3 ชิ้น ได้แก่ คันเร่ง, เบรก และคลัช ทั้งหมดใช้ควบคุมความเร็วรถ
- ชุดไฟเลี้ยว ประกอบไปด้วยสวิตช์ไฟเลี้ยว และ สตาร์ท-ดับเครื่องยนต์
- ชุดเกียร์ ใช้สำหรับกำหนดทิศทางการวิ่ง และอัตราทดของเครื่องยนต์



รูปที่ 3.1 แสดงรายละเอียดของอุปกรณ์ทั้งหมดที่เชื่อมต่อกับคอมพิวเตอร์

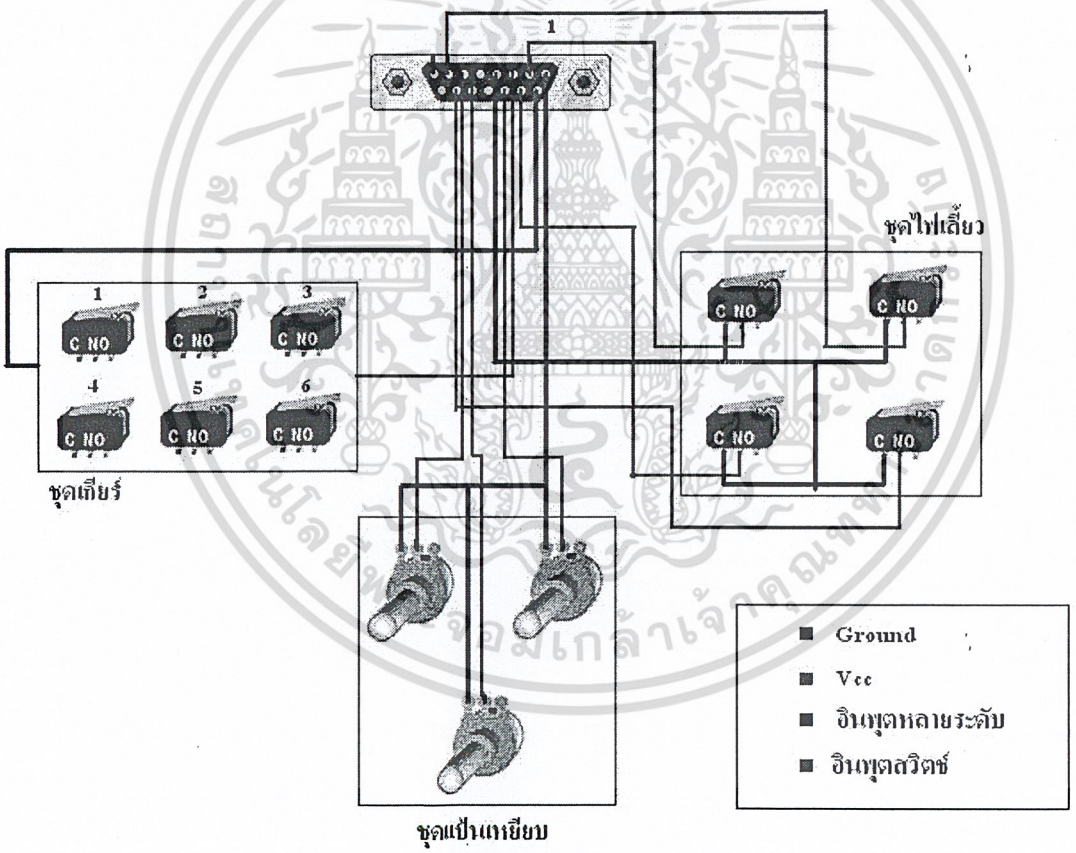
จากรูปแสดงให้เห็นว่าการเชื่อมต่ออุปกรณ์ส่วนใหญ่จะทำผ่าน Game Port เนื่องจากมีรูปแบบการเชื่อมต่อที่ง่ายถ้าเทียบกับ Port อื่นๆ และสามารถรองรับอุปกรณ์พื้นฐานที่จะใช้สำหรับโครงงานนี้ได้เกือบ

ครบถ้วน ส่วนที่เหลือ ได้แก่ พวงมาลัยจึงนำไปต่อผ่าน USB Port และจอภาพซึ่งต่อผ่าน Display Port ปกติสำหรับทุกเครื่องคอมพิวเตอร์อยู่แล้ว

3.1.1 การเชื่อมต่อเซ็นเซอร์เข้ากับ Game Port

การต่ออุปกรณ์กับ Game Port เป็นส่วนที่ใช้สำหรับโครงงานโดยตรง ซึ่งวงจรโดยรวมของระบบประกอบด้วยอุปกรณ์ต่างๆ ดังนี้

- ไมโครสวิตช์จำนวนทั้งหมด 6 ตัวสำหรับทำเกียร์ตั้งแต่เกียร์ 1 จนถึง 6 โดยการแปลงอินพุตผ่านขา J1X เพียงขาเดียว
- ความต้านทานปรับค่าได้ขนาด 100 กิโลโอห์ม จำนวน 3 ตัวเพื่อเป็นเซ็นเซอร์ในการรับอินพุตจากการเหยียบคันเร่ง, เบรก และคลัช
- ไฟเลี้ยว และสวิตช์สตาร์ท-ดับเครื่องยนต์ด้วย ซึ่งเป็นสวิตช์ชนิด ON-OFF-ON แต่ละตัวมี 3 ขา จำนวน 2 ตัว

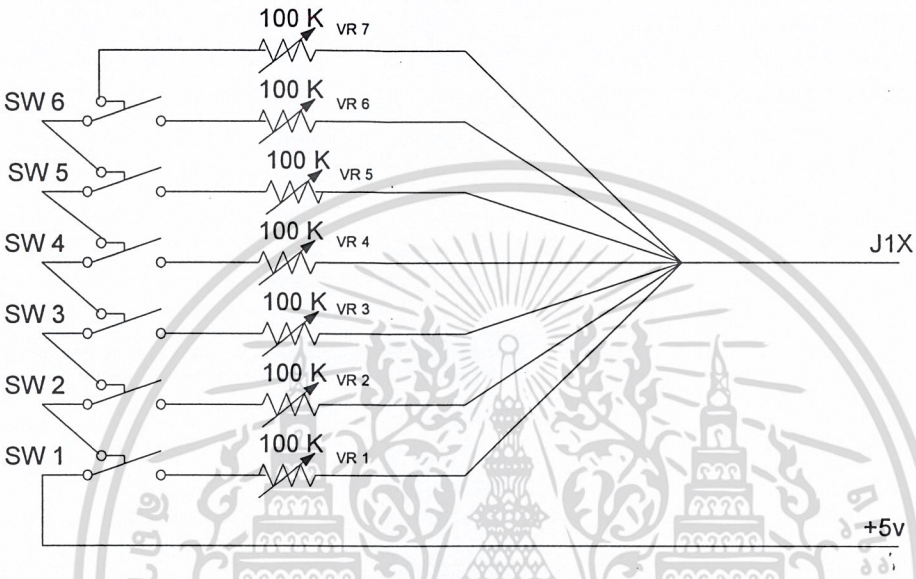


รูปที่ 3.2 การเชื่อมเซ็นเซอร์รับค่าผ่าน Game Port

จากรูปจะเห็นว่ามีการแบ่งอุปกรณ์ออกเป็น 3 ชิ้นหลักๆ ได้แก่ ชุดเกียร์, ชุดแป้นเหยียบ และชุดคลัช ซึ่งจะสามารถรับอินพุตจากแต่ละชุดอุปกรณ์ได้ดังนี้

3.1.1.1 การรับอินพุตจากชุดเกียร์

ชุดเกียร์จะใช้สายสัญญาณจาก Game Port เพียงสองเส้นได้แก่ J1X และ +5v ซึ่งจะต้องแปลงค่าของ V ที่ขา J1X ให้อยู่ในช่วงระหว่าง 0 ถึง 5 Volts จึงต้องใช้ R เข้ามาช่วยในการแปลงค่าความต่างศักย์ดังนี้



รูปที่ 3.3 แสดงวิธีการแปลงสัญญาณของชุดเกียร์

การรับสัญญาณอินพุตของขา J1X, J2Y, J2X และ J2Y ของ Game Port เป็นขาสัญญาณที่สามารถรับอินพุตได้หลากหลายระดับ และทำหน้าที่แปลงสัญญาณ Analog เป็น Digital ที่สามารถตรวจสอบระดับความต่างของสัญญาณได้ถึง 255 ระดับ (ตั้งแต่ 0 จนถึง +5 Volts) ดังนั้นจึงสามารถนำมาประยุกต์ใช้สำหรับรับสัญญาณอินพุตจากเกียร์ได้ดังนี้

ใช้ไมโครสวิตช์จำนวน 6 ตัวในการแปลงสัญญาณ กรณีของ SW 1 ซึ่งสวิตช์นี้จะมีหน้าสัมผัสอยู่ 2 ตำแหน่ง คือ เมื่อเวลาปล่อยสวิตช์ สวิตช์จะเชื่อมต่อกับสวิตช์ SW 2 แต่เมื่อกดสวิตช์ ส่วนที่เชื่อมต่อกับสวิตช์ SW 2 จะขาด และมาเชื่อมต่อกับ VR 1 แทน ซึ่ง VR 1 นี้จะเป็นตัวต้านทานปรับค่าได้ขนาด 100 กิโลโอห์ม ทำให้ค่าความต่างศักย์ของขา J1X ขึ้นอยู่กับ VR 1 เพียงตัวเดียว เช่นเดียวกัน กับ SW 2 จนถึง SW 5 ซึ่งความต่างศักย์จะขึ้นอยู่กับ VR ที่เชื่อมต่อตัวนั้นๆ แต่ SW 6 นั้นจะเชื่อมต่อกับ VR 7 ที่ใช้แสดงสถานะของเกียร์ว่าง ค่าความต้านทานของแต่ละเกียร์เป็นดังนี้

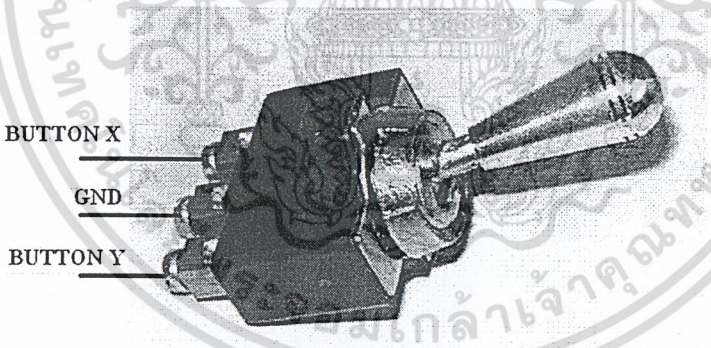
สวิตช์ที่ถูกกด	VR ที่มีผล	เกียร์
ไม่ได้กด	VR 7	ว่าง
SW 1	VR 1	1
SW 2	VR 2	2
SW 3	VR 3	3
SW 4	VR 4	4
SW 5	VR 5	5
SW 6	VR 6	ถอยหลัง

ตารางที่ 3.1 ค่าความต้านทาน สำหรับแต่ละเกียร์

ดังนั้น VR ทั้ง 7 ตัวจะต้องตั้งค่าโดยการประมาณค่าความต่างศักย์ออกเป็น 7 ช่วง โดยช่วงจะอยู่ที่ 0 ถึง +5V ให้ทำการปรับค่าจนได้ค่าที่ต่างกัน 7 ค่าและความห่างเท่าๆ กัน หลังจากนั้นจะตรวจสอบสถานะของเกียร์ด้วย Software ต่อไป

3.1.1.2 การรับอินพุตจากชุดไฟเลี้ยว

ชุดไฟเลี้ยวใช้สวิตช์ชนิด ON-OFF-ON หรือเรียกว่าสวิตช์แบบ 3 เสตท สวิตช์ชนิดนี้จะมีตำแหน่งในการโยก 3 ตำแหน่ง ซึ่งจะทำงานได้เทียบเท่ากับสวิตช์ ON-OFF จำนวนสองตัว โดยจะทำการต่อสายสัญญาณดังนี้



รูปที่ 3.4 แสดงวิธีการเชื่อมต่อสวิตช์เข้ากับสายสัญญาณ

- สำหรับไฟเลี้ยวก็ต่อสัญญาณ GND เข้าที่ขาตรงกลาง และต่อ BUTTON X และ BUTTON Y เข้าที่ขา 2 และ 7 ของ Game Port ซึ่งเป็นขาที่ใช้สำหรับรับอินพุตแบบสวิตช์ BUTTON 0 และ BUTTON 1 ตามลำดับ
- สำหรับสวิตช์ที่ใช้สำหรับสตาร์ท-ดับเครื่องยนต์ ก็เช่นเดียวกัน ต่อสัญญาณ GND เข้าที่ขาตรงกลาง และต่อ BUTTON X และ Y เข้าที่ขา 10 และ 14 ซึ่งเป็นขา BUTTON 2 และ BUTTON 3 ตามลำดับ

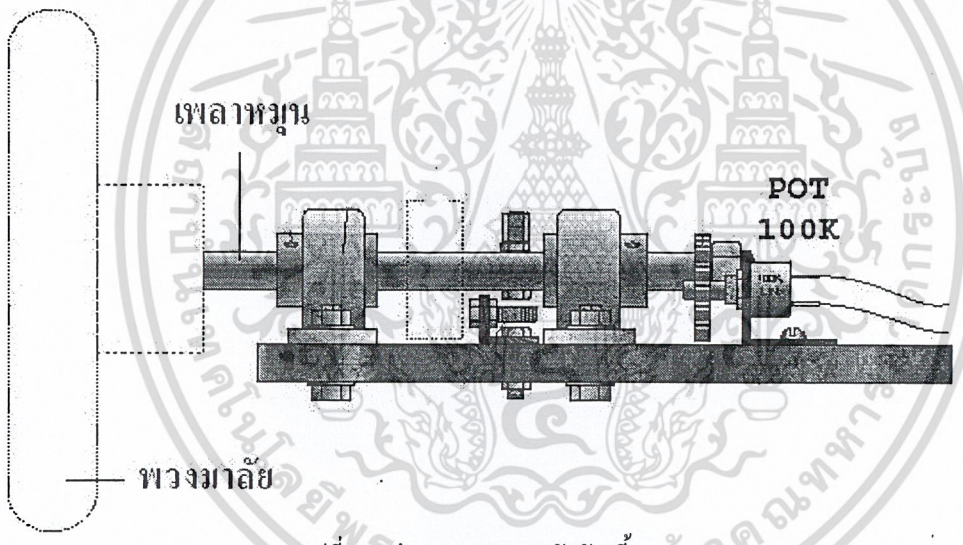
3.1.1.3 การรับอินพุตจากชุดแป้นเหยียบ

ชุดแป้นเหยียบประกอบด้วยแป้นเหยียบทั้งหมด 3 ชุด ได้แก่ คันเร่ง, เบรก และคลัช ซึ่งแป้นเหยียบแต่ละชุดจะมีการนำความต้านทานแบบปรับค่าได้ 100 กิโลโอห์ม มาติดที่แกนหมุน เมื่อมีการเหยียบแป้นเหยียบลงไป ก็จะทำให้ตัวต้านทานหมุนตามที่แป้นเหยียบถูกเหยียบลงไป ทำให้เกิดค่าความต่างศักย์ที่ตกคร่อมความต้านทาน จึงสามารถรับค่าความต่างศักย์นั้นเอาไปใช้เป็นอินพุตสำหรับการเหยียบแป้นเหยียบได้แตกต่างกันหลายระดับ

ตัวต้านทานแต่ละตัวจะต่อกับไฟเลี้ยง +5V และอีกขาหนึ่งจะใช้ต่อกับขาอินพุตหลายระดับ ซึ่งได้แก่ J1Y , J2X และ J2Y (J1X ได้ถูกใช้ไปแล้วสำหรับเกียร์)

3.1.2 การทำงานของพวงมาลัยขับรถ

พวงมาลัยขับรถที่ใช้ในโครงงานนี้ ใช้ชุดสำเร็จรูปพวงมาลัยซึ่งมีอัตราการเลี้ยวที่แม่นยำ และน้ำหนักเหมาะสม หลักการของพวงมาลัยนี้ อาศัยการหมุนพวงมาลัยจะเป็นการหมุนเพลา ซึ่งเพลาจะไปหมุนตัวต้านทานปรับค่าได้ ที่มีขนาด 100K อธิบายดังรูป



รูปที่ 3.5 ส่วนควบคุมการบังคับเลี้ยว

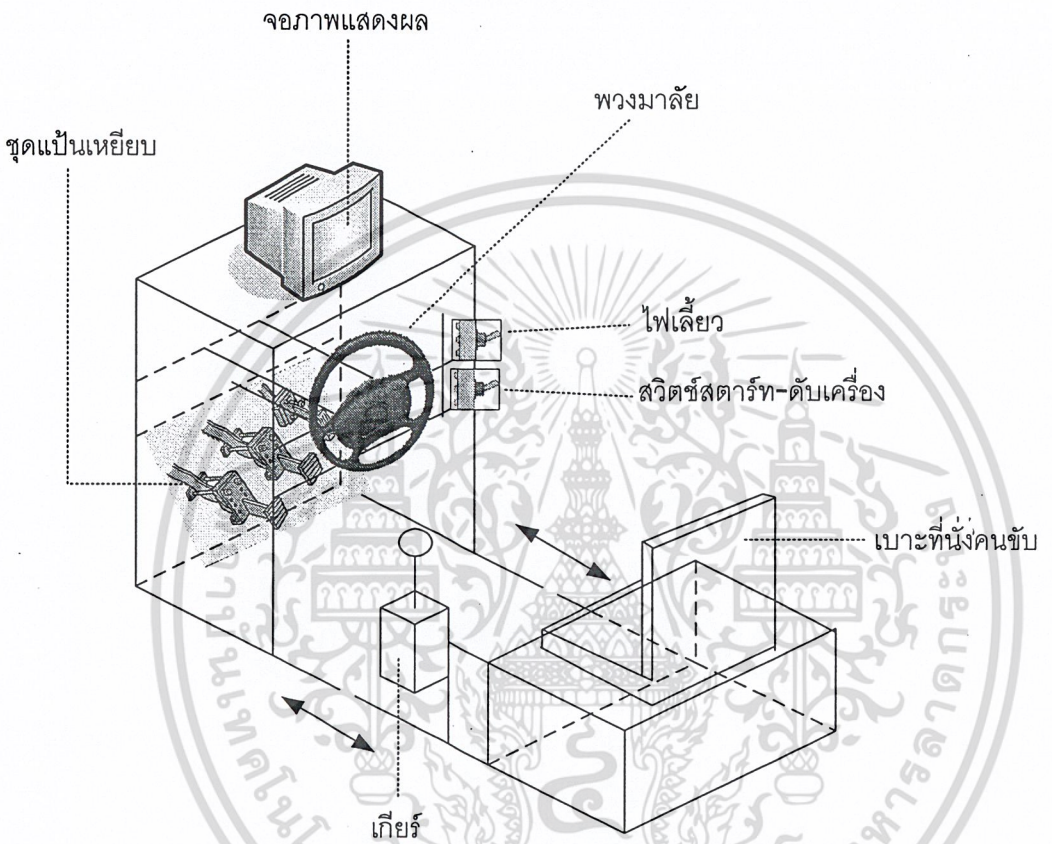
อธิบายจากรูปที่ 3.4 การหมุนพวงมาลัยนั้นจะทำให้ค่า R นั้นเปลี่ยนแปลงไป เริ่มต้นเมื่อตำแหน่งพวงมาลัยอยู่ตรงกลาง จะมีค่าความต้านทานที่ 50k และมีค่าความต่างศักย์ที่ 2.5V เมื่อมีการหมุนพวงมาลัยตามเข็มนาฬิกา ค่าของ R จะเพิ่มขึ้นจาก 50K ขึ้นไปอย่างช้าๆ จนมีค่าสูงสุดเป็น 100k (พวงมาลัยหักขวาสุด) และมีค่าความต่างศักย์ที่ +5V หากมีการหมุนทวนเข็มนาฬิกาค่า R ต่ำสุดก็คือ 0 Ohm (พวงมาลัยหักซ้ายสุด) มีความต่างศักย์ที่ 0V นำค่าความต่างศักย์ที่เปลี่ยนแปลงไปเป็นองศา และนำไปคำนวณหาทิศทางการเคลื่อนที่ของรถ

หมายเหตุ : การเชื่อมต่อของพวงมาลัยนั้นติดต่อผ่าน USB Port

3.2 โครงสร้างและส่วนประกอบชิ้นงาน

3.2.1 โครงเหล็กและการวางตำแหน่ง

โครงสร้างหลักประกอบด้วยสองส่วน คือโครงส่วนหน้าและโครงส่วนหลัง ซึ่งโครงส่วนหน้าจะใช้สำหรับติดตั้งจอภาพแสดงผล, พวงมาลัย, ไฟเลี้ยว, สวิตช์สตาร์ท-ดับเครื่อง และชุดแป้นเหยียบลงไป และโครงส่วนหลังจะใช้สำหรับติดตั้งเบาะที่นั่งคนขับ และเกียร์ แสดงดังรูป



รูปที่ 3.6 แสดงโครงสร้างหลัก และการวางอุปกรณ์ทั้งหมด

โครงส่วนหน้ามีขนาด (มากที่สุด) กว้าง x ยาว x สูง เท่ากับ 60 x 60 x 90 เซนติเมตร ตามลำดับ ทำจากเหล็กฉากรู และยึดแน่นด้วยน๊อต

- ชั้นบนสุดจะใช้แผ่นอครีลิคใส หนา 5 มิลลิเมตร ขนาด กว้าง x ยาว เท่ากับ 60 x 40 เซนติเมตร ใช้สำหรับรองจอภาพแสดงผล ให้อยู่ในระดับสายตาของผู้ขับขี่
- ชั้นรองลงมา ความสูงระดับ 60 เซนติเมตร (จากพื้น) จะใช้สำหรับยึดฐานพวงมาลัย หรับให้มีความสูงอยู่ระดับบริเวณหน้าอกของคนขับ
- ชั้นที่ 3 ระดับความสูง 45 เซนติเมตร ใช้สำหรับยึดชุดแป้นเหยียบซึ่งมีน้ำหนักมาก
- ที่ด้านขวาของพวงมาลัยจะใช้สำหรับติดตั้ง ไฟเลี้ยว และ สวิตช์สตาร์ท-ดับเครื่อง

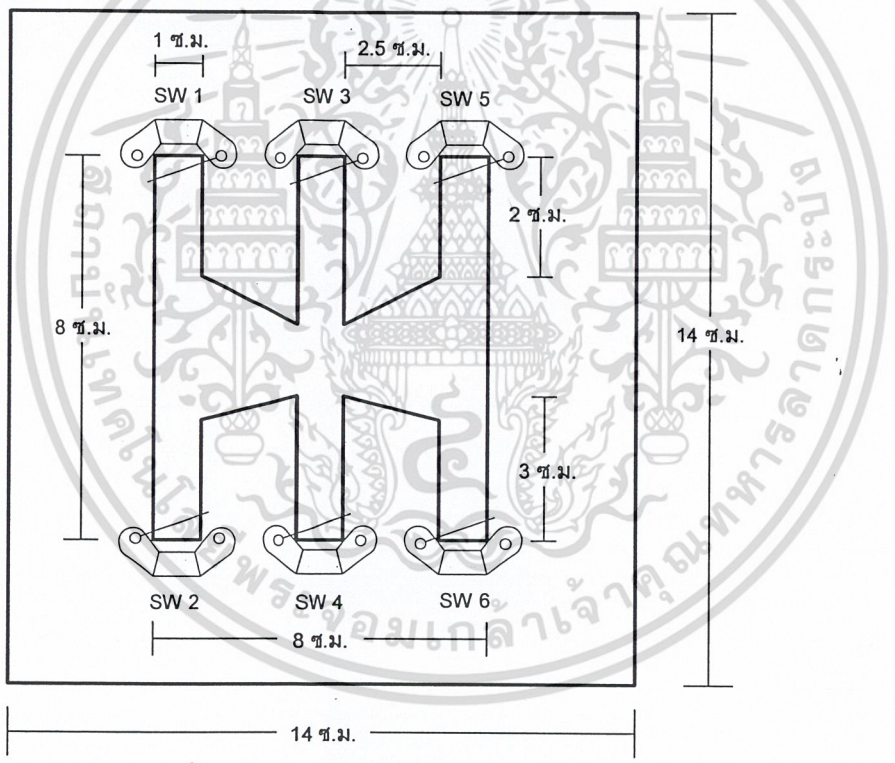
โครงส่วนหลังมีขนาด (มากที่สุด) กว้าง x ยาว x สูง เท่ากับ 60 x 70 x 30 เซนติเมตร ตามลำดับ ทำจากเหล็กฉากรู และขันด้วยน็อตเช่นเดียวกัน

- ด้านบนจะใช้สำหรับยึดเบาะที่นั่งคนขับ
- ด้านซ้ายเยื้องไปทางด้านหน้าของคนขับจะใช้สำหรับวางกระปุกเกียร์

โครงส่วนหน้าและหลังจากรูป จะเห็นว่าสามารถ ปรับระยะของโครงส่วนหน้า และหลังได้ ช่วยให้ระยะขาของผู้ขับขี่เหมาะสมกับความยาวส่วนขา ตามลูกศรที่ได้เขียนเอาไว้ในรูป

3.2.2 ชุดเกียร์

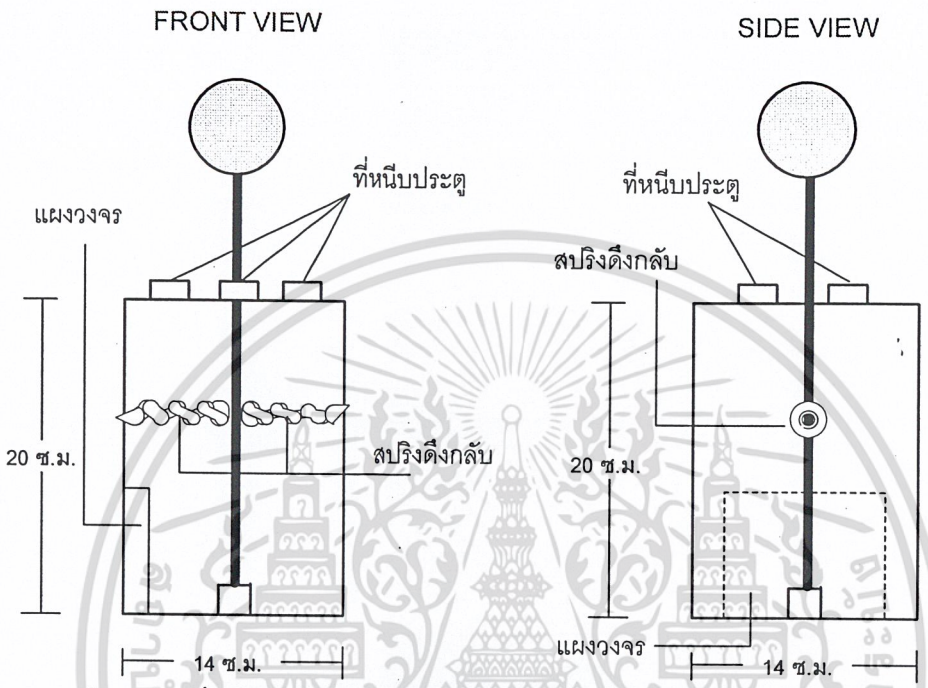
ชุดเกียร์ใช้สำหรับควบคุมอัตราทด และทิศทางการเคลื่อนที่ของรถ ประกอบด้วยเกียร์เดินหน้า 5 เกียร์ และเกียร์ถอยหลัง 1 เกียร์ นอกจากนี้มีสถานะอีกเกียร์หนึ่งที่เรียกว่าเกียร์ว่าง คือ ไม่มีการเข้าเกียร์ใดๆ รูปมุมมองทางด้านบนของชุดเกียร์เป็นดังนี้



รูปที่ 3.7 แสดงมุมมองทางด้านบนของชุดเกียร์

รูปมุมมองทางด้านบน แสดงให้เห็นเฉพาะชิ้นส่วนฝาบนของชุดเกียร์ซึ่ง แผ่นฝาจะทำจากแผ่นอครีลิคหนา 5 มิลลิเมตร ขนาดกว้างxยาว เท่ากับ 14x14 เซนติเมตร โดยที่พื้นที่ตรงกลางที่ใช้ทำเป็นร่องเกียร์จะถูกเจาะเป็นลักษณะ 6 แฉก ให้มีความกว้างของช่องเท่ากับ 1 เซนติเมตรเป็นอย่างน้อย เพื่อให้แท่งโลหะที่ใช้ทำเกียร์สามารถเลื่อนไปตามร่องได้ทุกเกียร์

สุดช่องแต่ละเกียร์ทั้ง 6 จะมีไมโครสวิตช์จำนวนทั้งสิ้น 6 ตัว ซึ่งไมโครสวิตช์แต่ละตัวจะนำไปเชื่อมต่อกับวงจรชุดเกียร์ตามที่ได้กล่าวมาแล้ว (ดูได้จากชื่อไมโครสวิตช์ที่กำกับเอาไว้) และนอกจากนี้สุดช่องเกียร์แต่ละเกียร์จะมี ก้ามปูที่หนีบประตู ซึ่งนำมาประยุกต์ใช้เพื่อให้เกียร์สามารถล๊อคอยู่ที่ตำแหน่งนั้นๆ ได้ มิฉะนั้นถ้าไม่ได้ล๊อคเกียร์เอาไว้ เกียร์จะถูกสปริงดึงให้กลับไปอยู่ที่ตำแหน่งเกียร์ว่างเสมอ



รูปที่ 3.8 แสดงมุมมองด้านหน้า และด้านข้างของชุดเกียร์

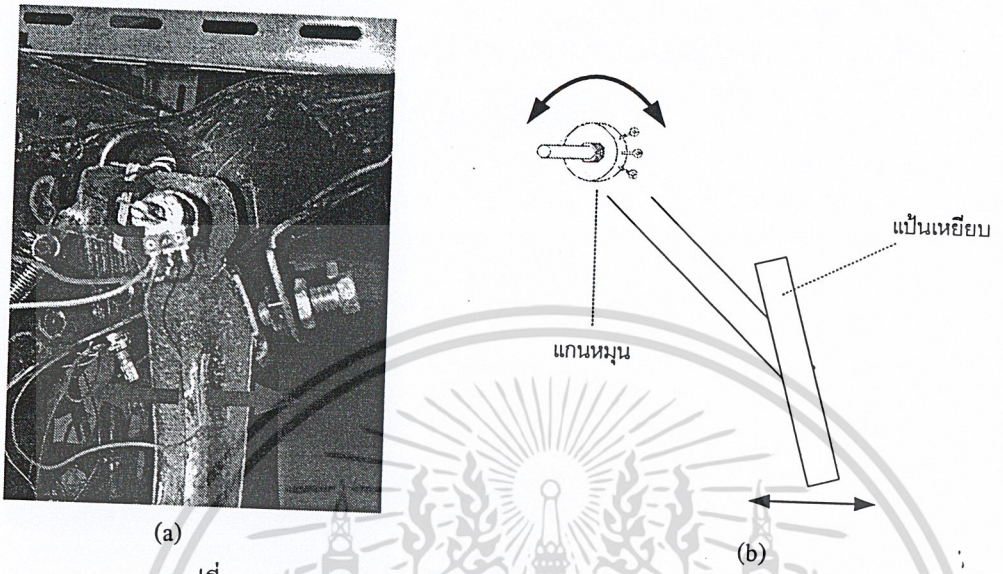
ภายในกล่องเกียร์ซึ่งมองจากมุมมองด้านหน้าจะเห็นว่ากล่องเกียร์จะมีขนาด กว้างxยาวxสูง เท่ากับ 14x14x20 เซนติเมตร ตามลำดับ ใช้แผ่นอครีลิคหนา 5 มิลลิเมตร ประกอบกันเป็นชุดเกียร์ด้วยกาวชนิดพิเศษ และยึดกับโครงเหล็กด้วยนอตเพื่อความแข็งแรงซึ่งภายในกล่องเกียร์จะมองเห็นว่าประกอบด้วยแท่งเกียร์ซึ่งเป็นแท่งโลหะกลม ขนาดเส้นผ่านศูนย์กลางประมาณ 8 มิลลิเมตร ระหว่างแท่งโลหะ กับฐานโลหะที่ติดกับแผ่นอครีลิคนั้น สามารถโยกได้อย่างอิสระ และใช้สปริงหดจำนวน 2 ตัวเพื่อใช้ดึงเกียร์ให้อยู่ตำแหน่งตรงกลางเสมอ เมื่อไม่ได้เข้าเกียร์อื่นเอาไว้

นอกจากนี้ภายในชุดเกียร์จะประกอบไปด้วยแผงวงจรไข่ปลาที่เป็นแผงวงจรชุดเกียร์ มีการติดตัวต้านทาน 10 กิโลโอห์ม จำนวนหลายๆ ตัว ร่วมกับไมโครงสวิตช์ (ต่อสายมาจากแผ่นฝาด้านบน) ซึ่งแสดงให้เห็นจากมุมมองด้านข้าง ตำแหน่งที่เป็นเส้นประ จะเป็นแผงวงจรที่มีการใช้ตัวต้านทาน ร่วมกับไมโครสวิตช์ในการแปลงสัญญาณจากชุดเกียร์ให้เป็นค่าความต่างศักย์ ผ่านขา J1X ตามที่ได้กล่าวมาแล้ว

ชุดเกียร์ทำจากแผ่นอครีลิคใสมีความหนาของแผ่น 5 มิลลิเมตร ประกอบขึ้นเป็นโครง มีขนาด กว้างx ยาวxสูง เท่ากับ 14x14x20 เซนติเมตร ตามลำดับ

3.2.3 ชุดแป้นเหยียบ

แป้นเหยียบที่นำมาใช้เป็นชุดแป้นเหยียบที่ถอดออกมาจากรถยนต์ โดยนำค่าตัวต้านทานปรับค่าได้ 100 กิโลโอห์มไปติด ดังแสดงในรูป



รูปที่ 3.9 แสดงการนำตัวต้านทานปรับค่าได้ ไปติดที่แกนหมุนคลัช

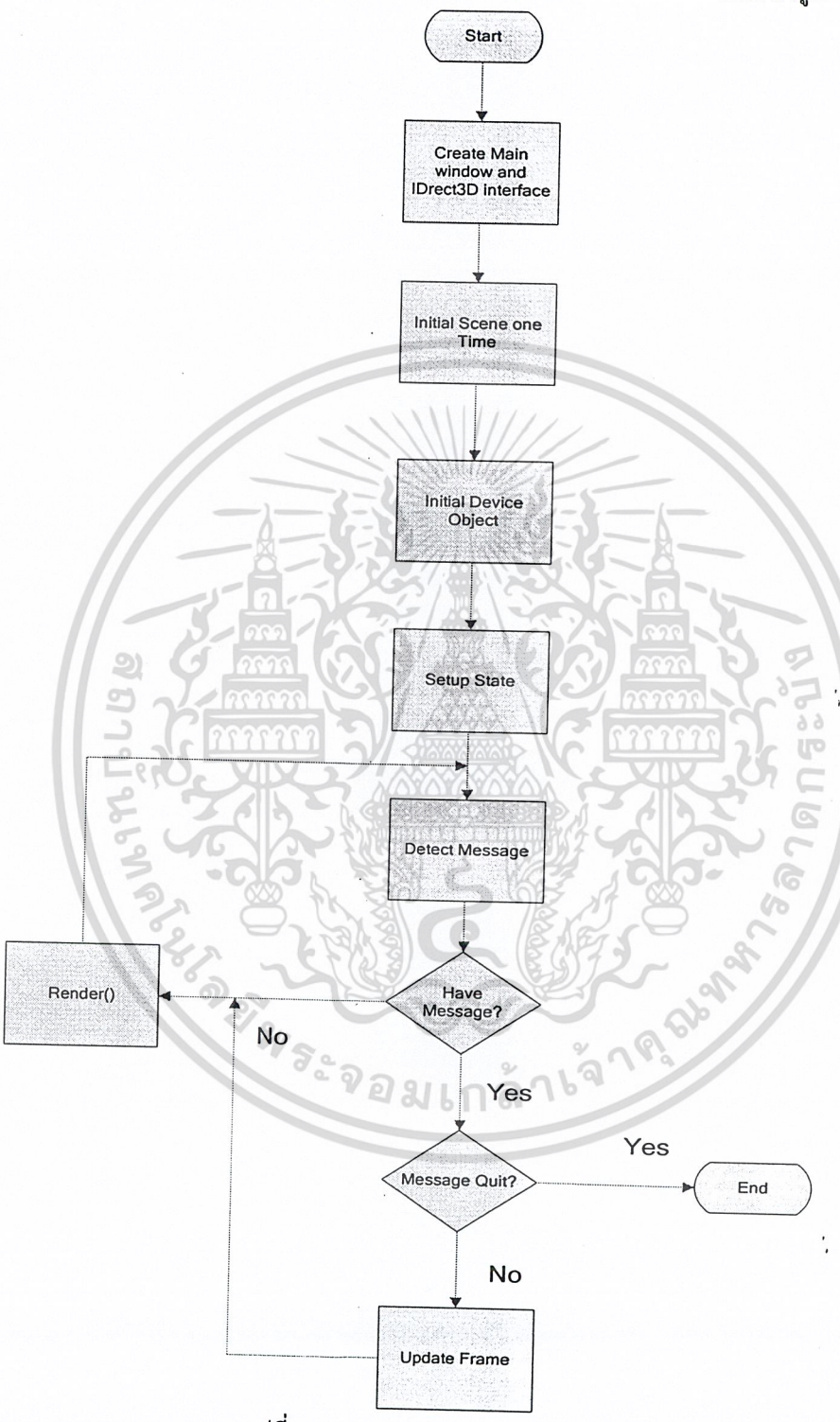
ตัวอย่างเป็นรูปการนำตัวต้านทานปรับค่าได้ ไปติดกับแกนหมุนของคลัช ซึ่งรูป (a) เป็นรูปที่ถ่ายจากอุปกรณ์จริง ส่วนรูป (b) เป็นการวาดจำลองขึ้นเพื่อให้รูปดูเข้าใจง่าย อธิบายได้ดังนี้ เมื่อมีการเหยียบแป้นเหยียบลงไป แกนหมุนจะหมุน และทำให้ R ปรับค่าได้นั้นหมุนตาม จึงทำให้เกิด ค่าความต้านทานที่เปลี่ยนแปลงไป และนำไปสู่ค่าความต่างศักย์ที่เปลี่ยนแปลงไป คอมพิวเตอร์จึงสามารถรับข้อมูลได้

แป้นเหยียบเบรก และคันเร่งก็อาศัยหลักการคล้ายๆ กัน โดยการนำค่าตัวต้านทานไปติด แต่เบรกมีการใช้ลวดในการเชื่อมแกนหมุนเข้ากับแกนแป้นเหยียบแทน

3.3 Software

3.3.1 Flow Chart

การออกแบบโปรแกรมเพื่อควบคุมการทำงานของระบบ เป็นไปตาม Flow Chart ดังรูป



รูปที่ 3.10 Flow Chart ของระบบ

แต่ละขั้นตอนการทำงาน อธิบายได้ดังต่อไปนี้

Create Main window and IDirect3D9 interface

เป็นจุดเริ่มต้น application มีหน้าที่

- Register Window class
- สร้าง window หลักเพื่อรองรับการแสดงผล
- ตั้งค่าคุณสมบัติให้กับ Window
- สร้าง interface ต่างๆที่ใช้สำหรับโปรแกรม เช่น การแสดงผล เสียง อินพุต

Initial Scene One Time

เป็นขั้นตอนหลังจากที่มีการ สร้าง IDirect3D9 interface แล้ว มีหน้าที่

- Set working directory เพื่อระบุโฟลเดอร์ที่ media files ทั้งหมดถูกรวบรวมไว้
- Create Thread
- ตั้งค่าเริ่มต้นสำหรับแต่ละ object
- สร้าง menu เพื่อให้ผู้ใช้เลือก โดยภายในประกอบด้วย main menu และ sub menu ย่อยต่างๆ

Initial Device Object

จะสร้าง object ต่างๆ ที่จำเป็นต้องใช้ มีหน้าที่

- สร้าง Vertex buffer ได้แก่ terrain vertex buffer , skybox vertex buffer , mirror vertex buffer และ console vertex buffer
- Initial font สำหรับแสดงผลตัวอักษร
- โหลด texture จากไฟล์
- โหลดไฟล์ .X

Setup State

ใช้ setup Render state , setup matrix ต่างๆ , setup texture etc. มีหน้าที่

- กำหนดค่า material เบื้องต้นที่จะใช้ในการแสดงผล
- กำหนดคุณสมบัติของ texture
- กำหนดสเตจที่จะใช้ในการ render เช่น การ enable z buffer
- กำหนด world matrix โดยเริ่มต้นให้เป็นเมททริก Identity
- กำหนด projection matrix เป็นการกำหนด viewing volume
- กำหนด view matrix เป็นการกำหนดลักษณะของกล้อง เช่น ทิศทาง
- กำหนดตำแหน่งของแหล่งกำเนิดแสงและคุณสมบัติของแสง

Detect Message

จะใช้ดักจับ Message ที่เข้ามาจากอุปกรณ์อินพุต มีหน้าที่

- รับ window message เมื่อมี message เข้ามา ก็ TranslateMessage และ DispatchMessage
- รับ input จาก DirectInput interface
- ถ้าไม่มี message เข้ามา เป็น idle time ก็สั่งให้ render ต่อไป
- ถ้า message ที่เข้ามาเป็น quit message ก็ให้ออกจากโปรแกรม

Render

ใช้แสดงผล ซึ่งเกิดจากการ swap back buffer มายัง font buffer มีหน้าที่

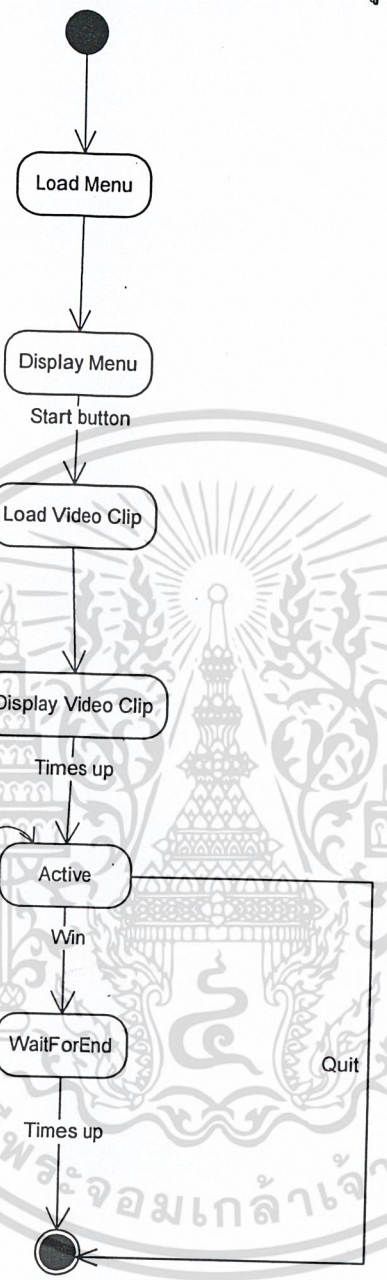
- Clear viewport
- Render scene สำหรับ ซึ่งจะแตกต่างกันไปตามแต่ละสเตจของโปรแกรม
- Swap back buffer to front buffer
- อาจจะมีการ setup render state ไปแทนที่ state เดิมเมื่อต้องการ render ต่างไปจากที่เคย setup ไว้ตอนต้น

Update Frame

สำหรับทำ animation กับโปรแกรมโดยรับ input แล้วปรับค่าต่างๆ ตาม input ที่รับเข้ามา ค่า input ต่างๆ ได้แก่ ค่าสถานะของเกียร์ ค่าเบรก คันเร่ง คลัช

3.3.2 State Diagram

State Diagram ซึ่งสามารถอธิบายการทำงานของระบบแสดงได้ดังรูป



รูปที่ 3.11 State Diagram ของโปรแกรมควบคุมระบบ

คำอธิบาย

Load Menu

เมื่อเริ่มต้นโปรแกรมจะเข้าสู่ state load menu เพื่อทำการโหลดค่าต่างๆที่จำเป็น ในการแสดงผลเมนู

Display Menu

เมื่อโหลดเมนูสำเร็จจึงให้ render ออกมาให้ผู้ใช้เลือก

Load Video Clip

เมื่อผู้ใช้เลือกเมนู play โดยการกดปุ่มสตาร์ทจึงเริ่มต้นโหลดวิดีโอคลิปสอนการขับขี่

Display Video Clip

แสดงการขับขี่จนจบตามเวลาที่กำหนด

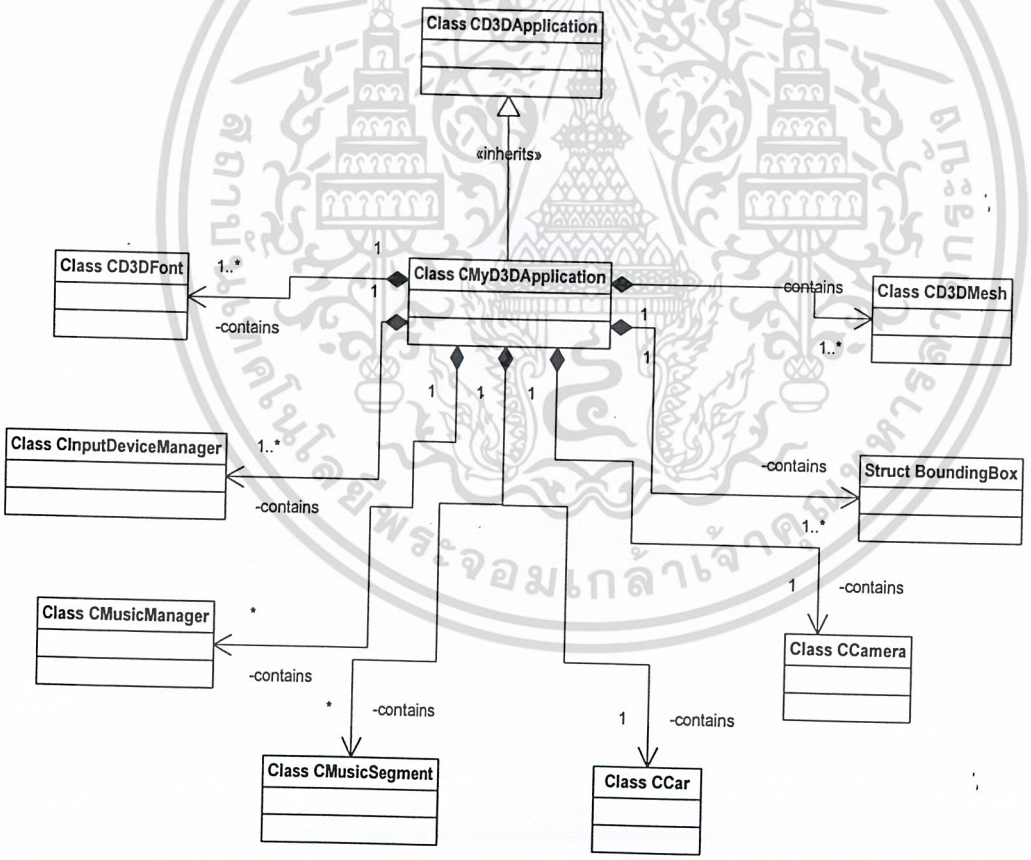
Active

Render ฉาก และรับอินพุตจากผู้ใช้ เป็นสเตจการสอบขับขี่จะสามารถสอบไปได้เรื่อยๆ จนกว่าจะผ่านการทดสอบหรือกดปุ่มออกจากโปรแกรม

WaitforEnd

เมื่อผู้ขับขี่สามารถผ่านการทดสอบ ก็ให้เล่นเพลงสักครู่และแสดงผลการทดสอบเป็นการจบการทดสอบ

3.3.3 Class Diagram



รูปที่ 3.12 Class Diagram แสดงความสัมพันธ์ระหว่าง class (รายละเอียดดูในภาคผนวก)

คำอธิบาย

Class CD3DApplication

- ใช้เริ่มต้นโปรแกรมในการสร้างหน้าต่างวินโดว์เพื่อใช้รองรองการแสดงผลโปรแกรมทั้งหมด
- ใช้ในการสร้างตัวอินเตอร์เฟสติดต่อกับ DirectX
- ใช้รันโปรแกรมพร้อมกับรับข้อมูล เพื่อเรียกใช้งาน class CMyD3DApplication

Class CMyD3DApplication

เป็นคลาสที่ใช้ทำงานหลักๆทั้งหมดของโปรแกรม

- ใช้ในการโหลดข้อมูลจากไฟล์ .x , ไฟล์เสียง และไฟล์รูปภาพ
- ใช้ในการแสดงผลโมเดลทั้งหมด
- ใช้ในการคำนวณการเคลื่อนที่ของรถที่สัมพันธ์กับการเคลื่อนที่ของกล้อง
- ใช้ในการสร้าง bounding box และ ตรวจสอบการชน
- ใช้ในการคำประมวลผลและแสดงผลภาพในกระจก

Class CD3DFont

เป็นคลาสที่ใช้จัดการกับตัวอักษรทั้งแบบ 2 มิติและ 3 มิติ

Class CInputDeviceManager

- ใช้ direct input เพื่อติดต่อกับอุปกรณ์อินเตอเฟสภายนอก เช่น คีย์บอร์ด , เมาส์ หรืออุปกรณ์ต่อพ่วงอื่นๆ

Class CMusicManager

- ใช้โหลดข้อมูลเสียงจากไฟล์ wav , MIDI มาเก็บลงในเซกเมนต์
- ใช้จัดการเรื่องของเสียงที่จะสัมพันธ์กับผู้ฟัง เช่น การเล่นเสียงแบบ 3 มิติ การควบคุมคุณภาพของเสียง

Class CSegmentManager

- ใช้จัดการกับข้อมูลเสียงที่อยู่ในแต่ละเซกเมนต์ เช่น การเล่น(Play) การหยุด(Stop) การเล่นซ้ำ (Repeat)

Class CCar

- ใช้เก็บข้อมูลซึ่งเป็นพารามิเตอร์ที่จำเป็นของรถยนต์ ได้แก่
- ตำแหน่งปัจจุบันใน world space
- ตำแหน่งของด้านหน้ารถทางซ้าย

- ตำแหน่งของหน้ารถทางขวา
- ตำแหน่งของด้านหลังรถทางซ้าย
- ตำแหน่งของหลังรถทางขวา
- มุมของล้อหน้า
- ความเร็ว
- ความเร่ง
- เกียร์
- ค่าความเร็วรอบของเครื่อง
- สถานะของเครื่อง

Class CCamera

- ใช้สำหรับการควบคุมการทำงานของกล้อง

Class CD3DMesh

- ใช้ในการโหลดไฟล์ .x
- ใช้ในการจัดเรียงข้อมูลเวอร์เท็กในไฟล์ .x ใหม่เพื่อความเร็ว
- ใช้ในการแสดงผล

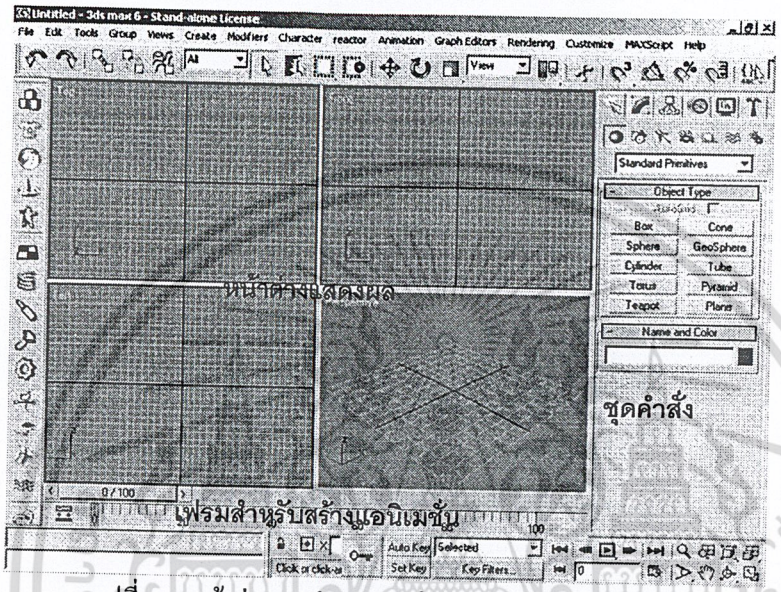
Struct BoundingBox

- ใช้ในการเก็บข้อมูล bounding box
- ใช้ในการคำนวณการชนในแต่ละ bounding box
- ใช้ในการอัปเดทข้อมูลในกรณีที่มีการเคลื่อนย้าย bounding box

บทที่ 4
การสร้างกราฟฟิกด้วย 3D Studio Max

4.1 การสร้างภาพกราฟฟิก

ใช้โปรแกรม 3D Studio Max สร้างวัตถุ 3 มิติ และโปรแกรม Quick 3D แปลงชนิดไฟล์เป็น x ไฟล์



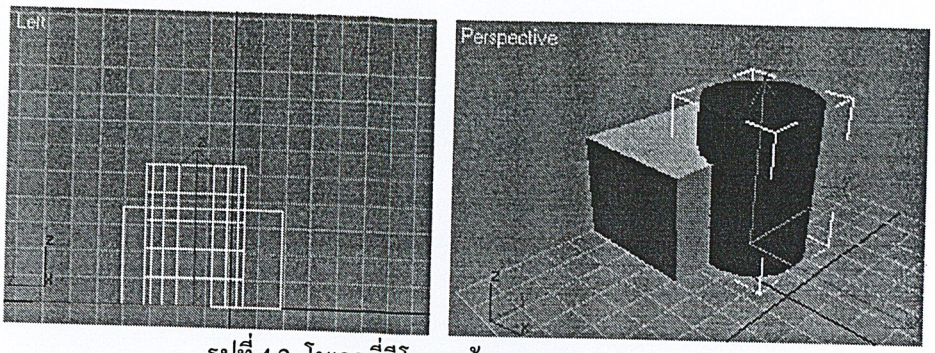
รูปที่ 4.1 หน้าต่างการทำงานของโปรแกรม 3d Studio Max

จากรูป มีหน้าต่างย่อยๆแสดงมุมมองด้านบน, ด้านหน้า, ด้านข้าง และแบบ perspective เพื่อความแม่นยำในการสร้างโมเดลและการทำแอนิเมชั่น

4.1.1 สร้างวัตถุให้เป็นโมเดล 3 มิติ มี 3 วิธีคือ

4.1.1.1 โครงสร้างแบบ Mesh และ Poly

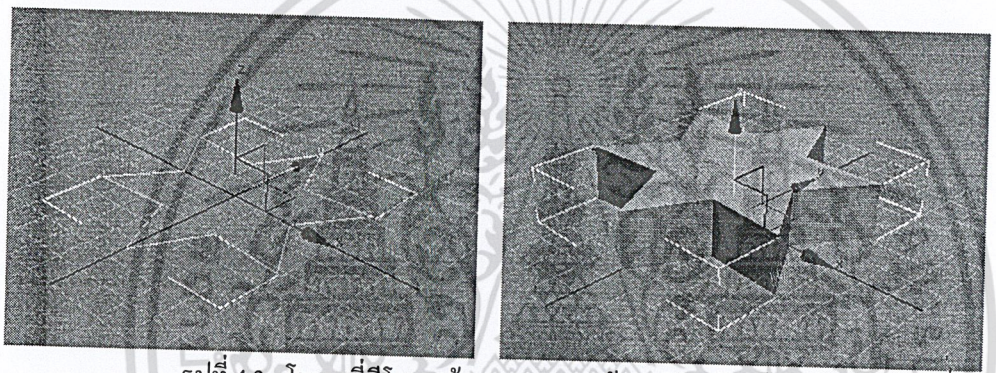
เป็นโครงสร้างที่ประกอบขึ้นมาจากจุดและเส้นตรงที่เชื่อมต่อกันเป็นระนาบและนำระนาบมาต่อกันเป็นรูปทรง 3 มิติ เป็นวิธีที่นิยมนำมาใช้งานมากที่สุดในปัจจุบัน เพราะความง่ายในการขึ้นรูปทรงของโมเดล



รูปที่ 4.2 โมเดลที่มีโครงสร้างแบบ Mesh และ Poly

4.1.1.2 โครงสร้างแบบ Spline

เป็นโครงสร้างที่ประกอบขึ้นมาจากจุดและเส้นเช่นเดียวกับ Mesh และ Poly แต่สามารถดัดให้เป็นเส้นโค้งได้อย่างอิสระก่อนนำไปสร้างพื้นผิว นิยมใช้สร้างรูปทรงอิสระ



รูปที่ 4.3 โมเดลที่มีโครงสร้างจากจุดและเส้น 2 มิติ สร้างเป็นวัตถุ 3 มิติ

4.1.1.3 โครงสร้างแบบ NURBS

เหมาะสำหรับสร้างงานโมเดลแบบรูปทรงอิสระ ลักษณะส่วนประกอบต่างๆคล้ายกับแบบ Spline แต่สามารถแก้ไขรูปทรงได้ยืดหยุ่นกว่า

4.2 ตกแต่งแก้ไขรูปทรงของโมเดล

คำสั่งต่างๆ ในชุด modify จะแบ่งตามลักษณะของ Space หรือแกนอ้างอิงในการทำงาน แบ่งเป็น 2 ชนิด ใหญ่ๆ คือ

1. World Space Modifier คือ คำสั่งที่ทำงานกับ World Space หรือ ทำงานกับฉากทั้งหมด หรือวัตถุทั้งชิ้น

2. Object Space Modifier คือ คำสั่งในชุด Object Space ทำงานกับวัตถุทั้งชิ้น หรือ ทำงานเฉพาะส่วนประกอบย่อยที่ถูกเลือก เป็นชุดคำสั่งที่ใช้งานเป็นหลัก

คำสั่ง modify ที่สำคัญและใช้งานบ่อย

- Explicit Modifier เป็นคำสั่งที่ใช้เลือกโครงสร้างของวัตถุมาแก้ไข สามารถใช้เครื่องมือ Transformation แก้ไขส่วนประกอบย่อยได้โดยตรง

- Selection Modifiers เป็นคำสั่งที่ใช้เรียกส่วนประกอบย่อยของ Model

- Patch/Spline Editing เป็นคำสั่งสำหรับแก้ไข Model ที่เป็นเส้นสองมิติ

- Mesh Editing เป็นคำสั่งสำหรับแก้ไข Model ที่มีโครงสร้างแบบ Mesh หรือ Polygon

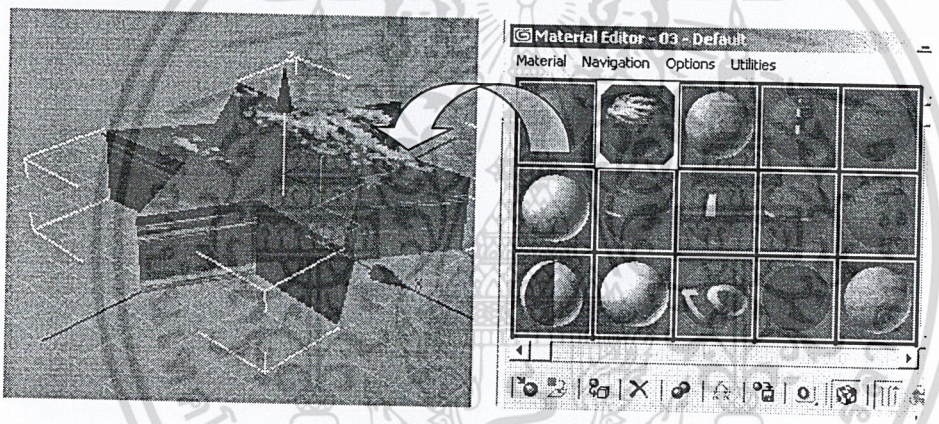
- Animation Modifiers เป็นคำสั่งที่ใช้กับงาน Animation

- UV Coordinate เป็นชุดคำสั่งสำหรับทำงานกับเรื่องของการกำหนดพื้นผิว คำสั่งส่วนใหญ่ใช้สำหรับกำหนดตำแหน่งในการใส่ลวดลวายลงบน Model

- Subdivision Surface เป็นชุดคำสั่งที่ใช้เพิ่มความละเอียดให้พื้นผิว เช่น การเปลี่ยนพื้นผิวที่มีลักษณะหยาบๆ ให้มีความเรียบโค้งมากขึ้น

4.3 ใส่ลักษณะพื้นผิวและลวดลายให้กับโมเดลของวัตถุ

เปิด Material Editor เลือกสีและลวดลาย คลิกแล้วลากมาวางที่โมเดล



รูปที่ 4.4 การใส่ลวดลายให้กับโมเดล

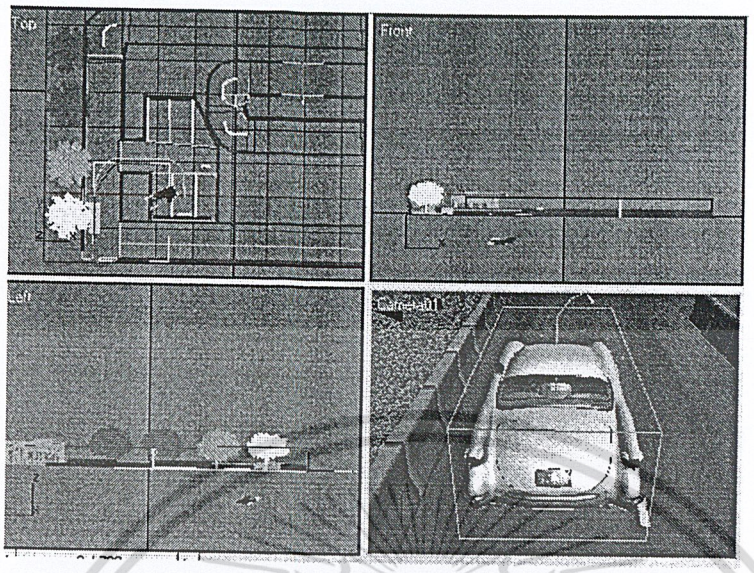
4.4 บันทึกไฟล์และแปลงชนิดไฟล์เพื่อนำไปใช้งาน

บันทึกโมเดลวัตถุเป็นไฟล์ชนิด 3ds แล้วนำไปแปลงเป็น x ไฟล์ ด้วยโปรแกรม Quick 3D

4.5 การสร้างแอนิเมชั่น

สร้างแอนิเมชั่นจากโปรแกรม 3D Studio Max แล้วบันทึกเป็นวิดิโอคลิป (ไฟล์ avi หรือ ไฟล์ mpeg) เพื่อสอนหลักเกณฑ์การสอบขับรถยนต์ภาคปฏิบัติ การสร้างแอนิเมชั่นมีขั้นตอนดังนี้

4.5.1 สร้างโมเดลวัตถุและจัดวางฉาก



รูปที่ 4.5 ตัวอย่างฉากแสดง 4 มุมมอง

จากรูป เป็นฉากที่จำลองสนามสอบขับรถยนต์ แสดง 4 มุมมอง คือ มุมบน มุมด้านหน้า มุมด้านข้าง และมุมกล้องตัวที่1 ซึ่งกำหนดให้แสดงมุมกล้องที่สร้างไว้ แทนที่มุมของ perspective

4.5.2 จัดแสง เงา กำหนดรายละเอียดให้แหล่งกำเนิดแสง และเอฟเฟ็กต์ของแสง

ในฉากมีการจัดแสงโดยใช้แหล่งกำเนิดแสงแบบ Skylight ซึ่งให้แสงเงาที่นุ่มนวล แต่ใช้เวลาเรนเดอร์นานขึ้น นอกจากแสงแบบ Skylight แล้วยังมีแสงอีกหลายประเภท คือ

4.5.2.1 Target Spot

เป็นแหล่งกำเนิดแสงเหมือน Spotlight แสงจะส่องจากขนาดเล็กออกไปหาขนาดใหญ่มีจุดเป้าหมายในตัว

4.5.2.2 Free Spot

เหมือนกันกับ Target Spot แต่ไม่มีจุดให้ปรับตำแหน่งเป้าหมาย ต้องหมุนตัว Spotlight เท่านั้น

4.5.2.3 Target Direct

แหล่งกำเนิดแสงที่ปล่อยแสงออกไปด้วยขนาดเท่ากันตลอด และมีจุดสำหรับปรับตำแหน่งเป้าหมายในตัว

4.5.2.4 Free Direct

เหมือนกันกับ Target Direct แต่ไม่มีจุดให้ปรับตำแหน่งเป้าหมาย

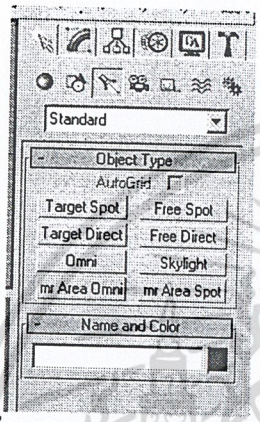
4.5.2.5 Omni

แหล่งกำเนิดแสงที่ปล่อยแสงออกไปรอบๆตัวเองทุกทิศทุกทางเหมือนดวงอาทิตย์ดวงเล็กๆ

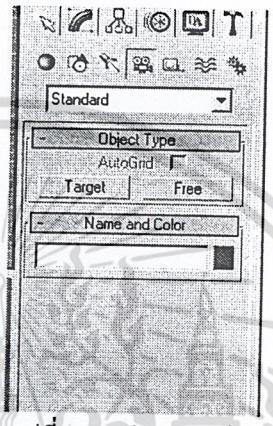
4.5.2.6 Skylight

แหล่งกำเนิดแสงแบบพิเศษที่ต้องใช้งานร่วมกับ Light Tracer ทำงานโดยจำลองตัวเองเป็นเหมือนโดมที่ปล่อยแสงเข้าหาฉากทุกทิศทาง ทำให้แสงเงานุ่มนวลกว่าปกติ แต่ใช้เวลาเรนเดอร์นาน

แหล่งกำเนิดแสงทั้งหมดมีการกำหนดค่าที่คล้ายกัน เช่น ค่าความสว่างหรือค่าการกระจายตัวของแสง



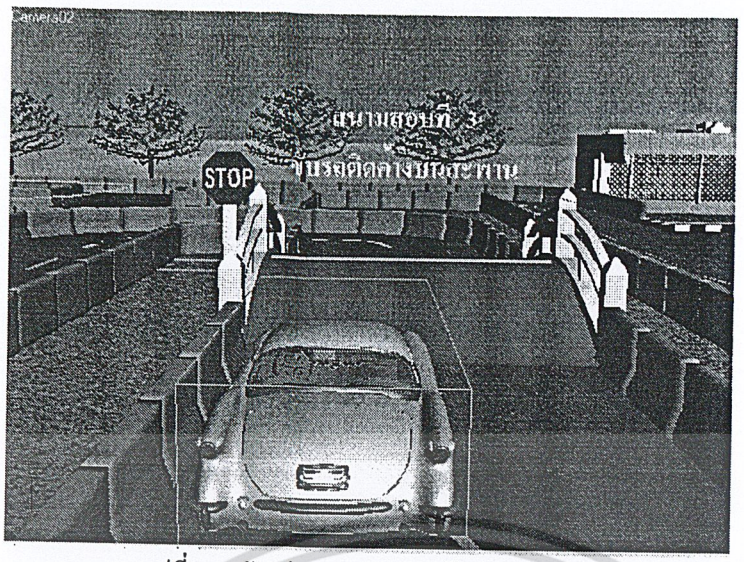
รูปที่ 4.6 ชนิดของแหล่งกำเนิดแสง



รูปที่ 4.7 ชนิดของกล้อง

4.5.3 สร้างกล้องและกำหนดการควบคุมมุมมองของกล้อง

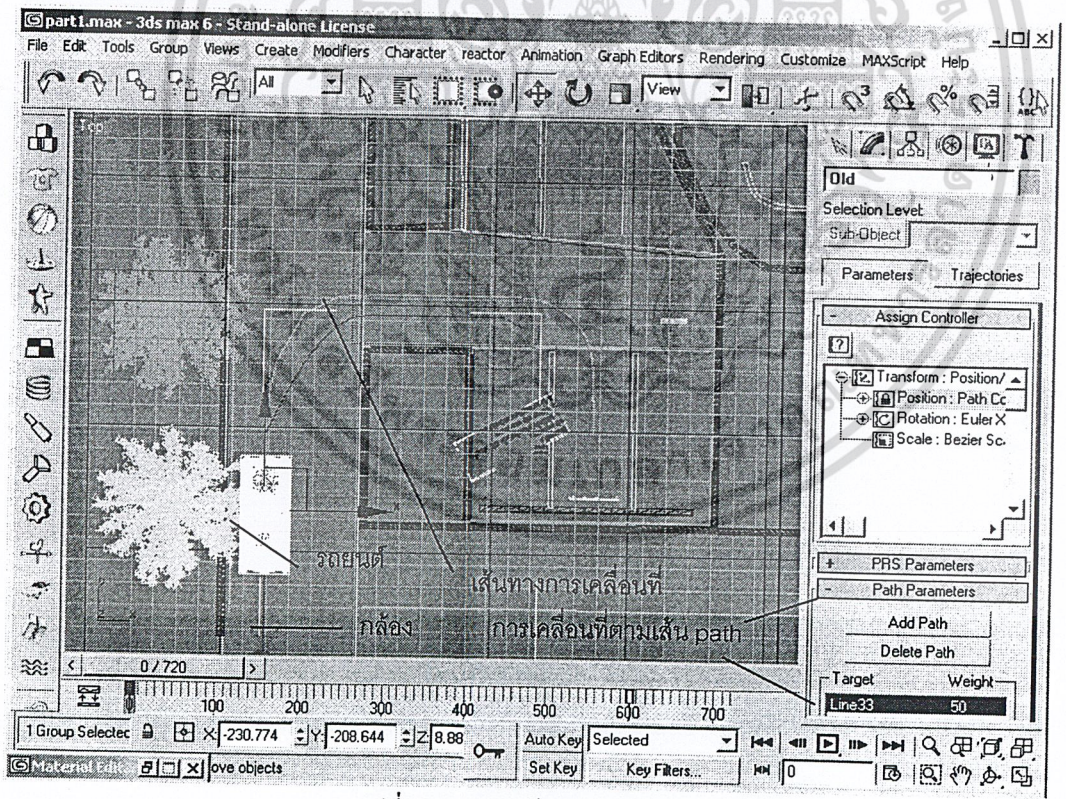
กล้องมี 2 แบบ คือ แบบที่มีจุดกำหนดตำแหน่งเป้าหมายและแบบที่ไม่มีจุดกำหนด กล้องสามารถปรับตำแหน่งมุมมองได้โดยใช้เครื่องมือ move, rotate และ scale เหมือนที่ใช้งานกับการสร้างโมเดล การดูภาพจากมุมกล้องให้คลิกขวาที่ชื่อของหน้าต่างแสดงผลอันใดอันหนึ่ง เลือกview > Camera(เลือกชื่อของกล้องที่ต้องการ)



รูปที่ 4.8 ตัวอย่างฉากจากมุมกล้อง camera02

4.5.4 กำหนดการเคลื่อนที่ให้โมเดลและกล้อง

แอนิเมชั่นที่สร้างขึ้นเป็นการจำลองการขับรถในสนามสอบที่ถูกต้อง จึงกำหนดการเคลื่อนที่ให้กับรถยนต์และกล้องเพื่อติดตามการเคลื่อนที่ของรถยนต์

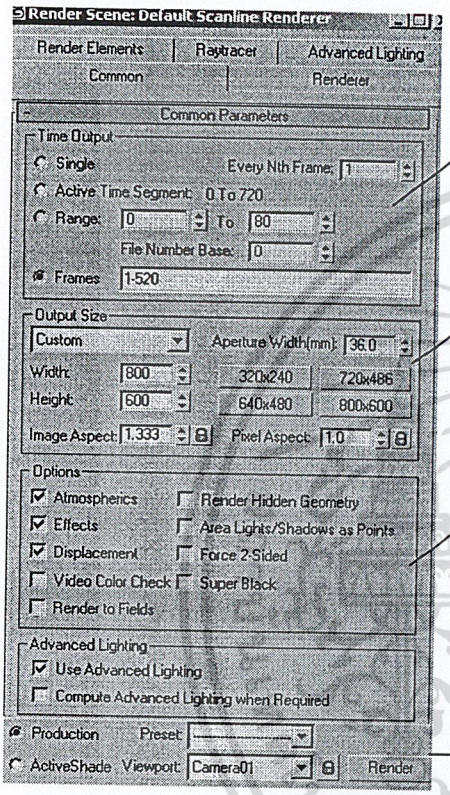


รูปที่ 4.9 มุมมองด้านบนของฉาก

จากรูป เป็นมุมมองด้านบนของฉาก หน้าต่างด้านขวาเป็นรายละเอียดของการเคลื่อนที่ของวัตถุ รถยนต์ซึ่งเชื่อมเข้ากับเส้นทางการเคลื่อนที่(path) รถยนต์จะเคลื่อนที่ไปตามเส้นทางที่กำหนดไว้ ในส่วนของกล้อง กำหนดในลักษณะเดียวกันแต่เคลื่อนที่ตามหลังรถยนต์

4.5.5 กำหนดการเรนเดอร์และกำหนดค่าสำหรับนำไปใช้งาน

แอนิเมชั่นที่เสร็จสมบูรณ์แล้ว นำมาบันทึกเป็นวิดิโอคลิป(ไฟล์ชนิด mpeg หรือ avi) ซึ่งนำไปใช้กับ direct X ได้



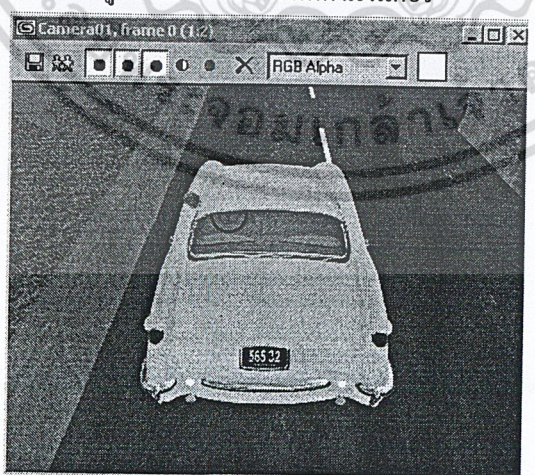
กำหนดค่าเวลาของแอนิเมชั่น สามารถกำหนดได้เป็นช่วงเฟรมหรือเฟรมเดียว

กำหนดขนาดหน้าต่างแสดงผล

กำหนดลักษณะการเรนเดอร์

เรนเดอร์งานแอนิเมชั่น บันทึกตามชนิดไฟล์ที่กำหนด

รูปที่ 4.10 การกำหนดค่าเรนเดอร์



รูปที่ 4.11 ตัวอย่างผลลัพธ์ของการเรนเดอร์แบบเฟรมเดียว

บทที่ 5
การทำงานของระบบ

5.1 การประมวลผลข้อมูลของระบบ

5.1.1 การคำนวณความเร็วรถ

ค่าอินพุทที่รับมาจากขาอินพุตหลายระดับของ Game Port ได้แก่ J1X, J2Y, J2X และ J2Y จะอยู่ในลักษณะของค่าทศนิยมที่มีค่าตั้งแต่ -1.00 ถึง 1.00 แต่โดยทั่วไปแล้วถ้าต่อตัวต้านทานปรับค่าได้เข้ากับขาเหล่านี้แล้ว จะสามารถแสดงค่าได้เพียง 0.00 ถึง 1.00 เพราะว่าค่าลบนั้นจะเกิดขึ้นได้เมื่อมีการจ่ายกระแสไฟลบเท่านั้น ซึ่งจะไม่ใช้กับตัวต้านทานปรับค่าได้

ค่าอินพุตจากคันเร่งก็เช่นเดียวกัน โดยหลักการแล้วจะมีค่าตั้งแต่ 0.00 ถึง 1.00 ดังนั้นเมื่อมีการเหยียบคันเร่งก็จะทำให้เกิดความเร่งของรถยนต์ขึ้น แล้วแต่ค่าอินพุตของคันเร่งว่าเหยียบลงไปมากเพียงใด ซึ่งค่าที่เหยียบคันเร่งนี้จะมีผลให้เกิดอัตราเร่งจริงของรถยนต์ก็ต่อเมื่อไม่มีการเหยียบคลัช และตำแหน่งเกียร์ปัจจุบันเป็นไม่ใช่เกียร์ว่างเท่านั้น มิฉะนั้นการเหยียบคันเร่งจะไม่ได้ทำให้รถเคลื่อนที่ เพียงแต่ทำให้รอบเครื่องยนเพิ่มขึ้นเท่านั้น

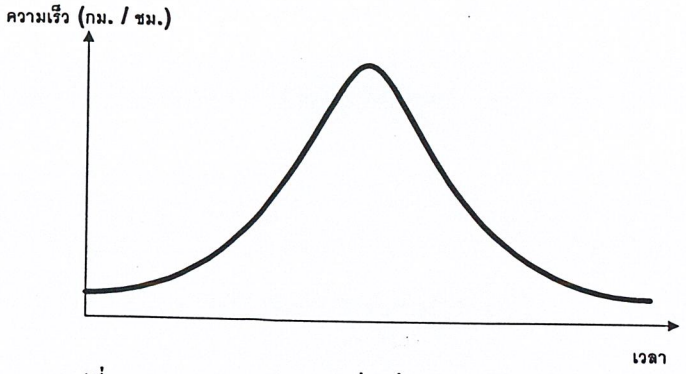
อัตราเร็วของรถยนต์ที่เคลื่อนที่ไปคำนวณจากสมการที่เคยได้กล่าวไปแล้ว (สมการ 2.2)

$$v = v_0 + a\Delta t$$

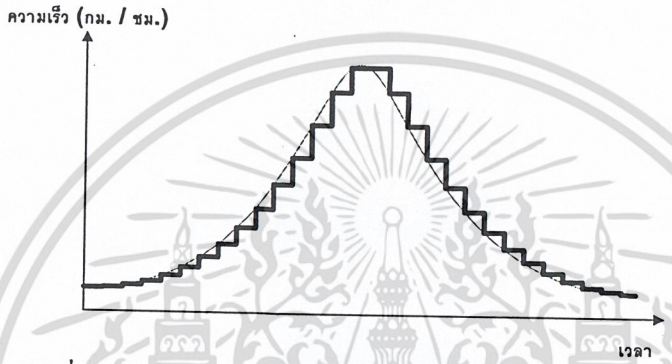
โดยจะแทนค่าของ v_0 ด้วยอัตราเร่งจากค่าที่ส่งมาจากเครื่องยนต์ โดยจะแบ่งระดับอัตราเร่งออกเป็นทั้งหมด 7 ระดับ ซึ่งอัตราเร่งจะมีค่าเป็น 0 เมื่อไม่มีการเหยียบคันเร่ง และจะมีค่าเพิ่มขึ้นๆ จนเป็น 7 เมื่อคันเร่งถูกเหยียบจนสุด ซึ่งเมื่อนำมาใช้คำนวณจริงแล้ว ค่า a (อัตราเร่ง) ซึ่งกล่าวไปแล้วข้างต้น จะสามารถแบ่งออกเป็นสองส่วน ได้แก่ Accel 1 และ Accel 2 โดย Accel 1 จะเป็นค่าที่รับมาจากอินพุตคันเร่งซึ่งจะมีค่า 0 ถึง 7 ระดับ และ Accel 2 นี่จำลองขึ้นมาจากอัตราทดเสมือน ที่ได้จากเฟืองทดของเกียร์ ซึ่งจะได้กล่าวต่อไปในเรื่องของอัตราทด ความสัมพันธ์ของ a กับ Accel 1 และ Accel 2 เป็นดังนี้

$$a = \text{Accel 1} \times \text{Accel 2} \qquad \ldots\ldots\ldots (5.1)$$

สุดท้ายก็นำมาคูณกันตามสามการ ได้ค่าของความเร็วที่เพิ่มมากขึ้นเมื่อมีการเหยียบคันเร่งเพิ่มขึ้นเรื่อยๆ ซึ่งทุกๆ เวลาเสี้ยววินาที จะมีการวนลูปกลับมาเข้าสมการนี้อีกเพื่อเพิ่มความเร็วขณะที่มีการเหยียบคันเร่งค้างเอาไว้ ความเร็วรถยนต์จริงที่เปลี่ยนแปลงไป เทียบกับคามเร็วรถในโปรแกรมจำลองจะแตกต่างกันตามลักษณะดังรูป



รูปที่ 5.1 กราฟแสดงความเร็วที่เปลี่ยนไปของรถยนต์จริง

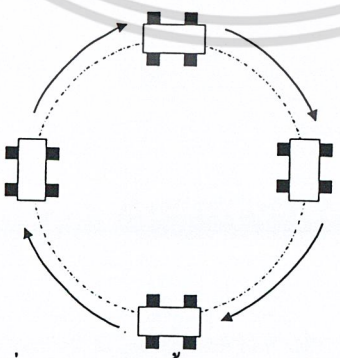


รูปที่ 5.2 กราฟแสดงความเร็วที่เปลี่ยนไปของรถในโปรแกรม

จากรูปทั้งสองด้านบนแสดงให้เห็นว่าโปรแกรมจะทำหน้าที่จำลองความเร็วจริงที่เปลี่ยนแปลงไป โดยมีการวนลูปมาตรวจสอบค่าความเร็วทุกเสี้ยววินาที แล้วแทนตลอดช่วงนั้นด้วยค่านั้นๆ ทั้งช่วงจนกว่าจะวนลูปมาคำนวณใหม่อีกครั้ง

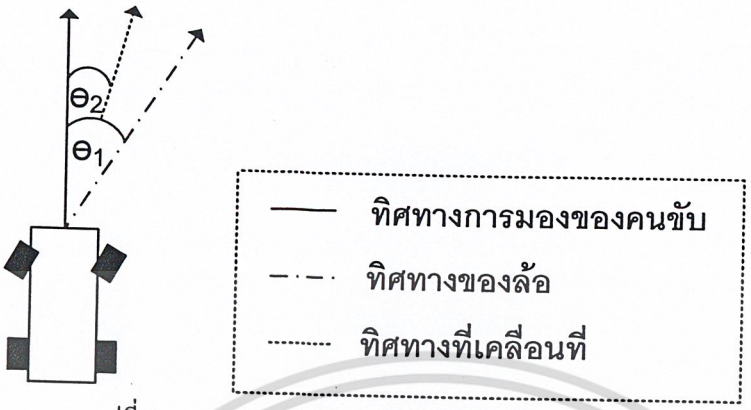
5.1.2 การเลี้ยว

การเลี้ยวของรถยนต์ก็เป็นสิ่งหนึ่งที่ต้องคำนึงถึงในการทำให้เกิดความเหมือนจริงในการขับขี่ โดยหลักการในการเลี้ยวนั้น ถ้าเป็นกรณีของรถยนต์จริง รถยนต์จะเคลื่อนที่ไปในลักษณะวงกลมโค้งมนซึ่งมีลักษณะดังรูป



รูปที่ 5.3 แสดงการเลี้ยวของรถยนต์จริง

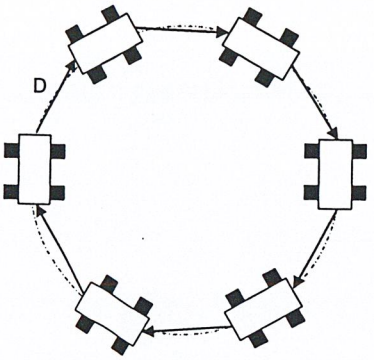
เมื่อมีการหักพวงมาลัยทำมุมคงที่รถยนต์จะวิ่งเป็นวงกลมมน ขณะที่รถยนต์ในโปรแกรมจะไม่สามารถทำได้เหมือนกับรถยนต์จริงอย่างสมบูรณ์เสียทีเดียว ต้องมีการใช้หลักการนี้มาช่วย



รูปที่ 5.4 แสดงมุมและทิศทางของรถจำลองในโปรแกรม

ต้องมีการพิจารณาทิศทาง 3 ทิศทาง ได้แก่ ทิศทางการมองของคนขับ (เป็นทิศทางที่คนขับมองตรงไปด้านหน้า) , ทิศทางของล้อ (เป็นทิศทางที่ล้อชี้ไปด้านหน้า) และทิศทางที่เคลื่อนที่ (เป็นทิศทางจำลองที่รถเคลื่อนที่ไป เพื่อเลียนแบบรถยนต์จริง) นอกจากทิศทางทั้ง 3 แล้ว ก็ต้องพิจารณามุมสองมุม ได้แก่ มุม θ_1 (เป็นมุมที่ทำระหย่างทิศทางการมองของคนขับ กับ ทิศทางของล้อ) และ มุม θ_2 (เป็นมุมที่ทำระหว่างทิศทางการเคลื่อนที่ และทิศทางการมองของคนขับ) หลักวิธีการจำลองการเลี้ยวมีดังนี้

- ทิศทางจริงจะมีเพียงทิศทางการมองของคนขับ และทิศทางของล้อ
- ทิศทางที่เคลื่อนที่เป็นเพียงทิศทางเสมือนที่สมมุติขึ้นเท่านั้น
- ความสัมพันธ์ของมุม θ_1 และ θ_2 อยู่ในลักษณะ θ_2 เป็นสัดส่วนของค่า θ_1 ในลักษณะ ตายตัว เช่น ถ้า θ_1 เท่ากับ 30 และอัตราส่วนเป็น 0.5 ดังนั้น θ_2 จะมีค่าเท่ากับ 15 องศา
- รถยนต์จะเคลื่อนที่ไปใน “ทิศทางที่เคลื่อนที่” เป็นระยะสั้นๆ สมมุติว่าระยะนั้นมีค่าเป็นตัวแปรค่าหนึ่ง (D)
- หลังจากที่เคลื่อนที่ไปเป็นระยะทาง D ใน “ทิศทางที่เคลื่อนที่” แล้ว จะอัพเดต และคำนวณทิศทางและมุมทั้งหมดใหม่ โดยรับค่าอินพุตจากพวงมาลัยเพื่อให้ได้ “ทิศทางของล้อ” และ “ทิศทางที่เคลื่อนที่ไป” จะกลายมาเป็น “ทิศทางการมองของคนขับ” แทน และ “ทิศทางที่เคลื่อนที่ไป” อันใหม่จะได้มาจากมุม θ_2 (ซึ่งต้องหา θ_1 มาก่อน) ก็จะสามารถจำลองการเคลื่อนที่ของรถในโปรแกรมให้สมจริงได้



รูปที่ 5.5 แสดงการเลี้ยวของรถยนต์ในโปรแกรม

จากรูปจะแสดงให้เห็นว่าการเลี้ยวของรถยนต์ในโปรแกรมจะทำในลักษณะเป็นเส้นตรงวิ่งไต่ไปตามเส้นโค้ง โดยค่า D ที่เห็นเป็นค่าระยะทางที่จะเคลื่อนที่ไป ก่อนที่จะมีการเริ่มคำนวณมุมและทิศทางกันใหม่ ค่า D นี้ยิ่งลดให้ได้สั้นลงเท่าใด ก็จะทำให้รถวิ่งได้สมจริงมากยิ่งขึ้น เพราะจะวิ่งในลักษณะเกือบเป็นวงกลมมน ซึ่งรถยนต์จริงจะมีค่า D ที่ต่ำมากๆ จนเข้าใกล้ 0

5.1.3 ความเร็วและอัตราทด

ความเร็วจะถูกจำกัดเอาไว้ในแต่ละเกียร์ ซึ่งได้มีการจำกัดค่าความเร็วสูงสุดของแต่ละเกียร์จำลองมาจากรถยนต์จริง โดยนำข้อมูลอ้างอิงจากรถยนต์จากรุ่น Toyota Corolla Altis มาเป็นต้นแบบการพัฒนา อัตรารถของเกียร์แต่ละเกียร์เป็นดังตาราง

เกียร์	อัตราทดจริง	อัตราทดเสมือน
เกียร์ 1	3.166	1.000
เกียร์ 2	1.904	0.601
เกียร์ 3	1.310	0.413
เกียร์ 4	0.969	0.306
เกียร์ 5	0.815	0.257
เกียร์ถอยหลัง	3.250	1.02

ตารางที่ 5.1 แสดงอัตราทดจริง เทียบกับอัตราทดเสมือน

จากตาราง อัตราทดจริงจะได้มาจากข้อมูลอัตราทดของเฟืองเกียร์จากรถยนต์ต้นแบบ ส่วนอัตราทดเสมือนเป็นอัตราทดที่เกิดจากการนำอัตราทดเกียร์จริงของแต่ละเกียร์มาเทียบกับอัตราทดจริงของเกียร์ 1 ซึ่งจะนำอัตราทดนี้ไปคำนวณความเร็วสูงสุด และความเร็วต่ำสุดของแต่ละเกียร์ได้ดังนี้

ถ้ากำหนดให้รอบเครื่องยนต์ต่ำสุดที่ 700 รอบ/นาที (ถ้าต่ำกว่านี้เครื่องยนต์จะดับ) และรอบเครื่องยนต์สูงสุดเท่ากับ 5000 รอบ/นาที (เมื่อเหยียบคันเร่งจนได้ค่า Accel เป็น 7) และสมมุติให้ความเร็วสูงสุดของเกียร์ 1

ทำได้ที่ 40 กิโลเมตร/ชั่วโมง ที่เครื่องยนต์ 5000 รอบ/นาที จะทำให้สามารถคำนวณอัตราเร็วสูงสุดและต่ำสุดที่เป็นไปได้สำหรับแต่ละเกียร์ ดังตาราง

เกียร์	อัตราเร็วต่ำสุด (กม. / ชม.) (ที่ 700 รอบ / นาที)	อัตราเร็วสูงสุด (กม. / ชม.) (ที่ 5000 รอบ / นาที)
เกียร์ 1	5.6	40
เกียร์ 2	9.2	66
เกียร์ 3	13.4	96
เกียร์ 4	18.2	130
เกียร์ 5	21.7	155
เกียร์ถอยหลัง	5.46	39

ตารางที่ 5.2 แสดงอัตราเร็วต่ำสุดและสูงสุดของแต่ละเกียร์

ถ้าสมมุติค่ารอบเครื่องยนต์มีความสัมพันธ์กับความเร็วรถแบบความสัมพันธ์เส้นตรงแล้ว จะสามารถคำนวณความเร็วต่ำสุดของแต่ละเกียร์ได้จาก

อัตราเร็วต่ำสุด = (อัตราเร็วสูงสุด x 700) / 5000

ค่าความเร็วต่ำสุดของแต่ละเกียร์ดังตารางแสดงให้เห็นว่าถ้ารถมีความเร็วต่ำกว่าที่กำหนดไว้ต่ำสุดของแต่ละเกียร์และไม่มีการเหยียบคลัช เครื่องยนต์จะดับ

5.1.4 การเพิ่มความเสมือนจริงสำหรับการเคลื่อนที่

เพื่อเพิ่มความเหมือนจริงให้กับการขับรถยนต์ จึงต้องมีการจำลองพฤติกรรมต่างๆ ในโปรแกรมเพื่อให้เกิดการตอบสนองในโปรแกรมให้คล้ายกับการตอบสนองของรถยนต์จริง โดยมีรายละเอียดความเหมือนจริงที่ได้มีการตรวจสอบในโปรแกรมในกรณีที่รถดับ มีดังนี้

- เลื่อนสวิตช์สตาร์ท / ดับเครื่องยนต์ ไปที่ตำแหน่ง "ดับเครื่อง"
- สตาร์ทเครื่องโดย ตำแหน่งเกียร์ปัจจุบันไม่ใช่เกียร์ว่าง (ไม่สามารถสตาร์ทได้)
- ไม่ได้เหยียบคลัชระหว่างเปลี่ยนเกียร์ ทุกๆ เกียร์ หรือเหยียบคลัชไม่สุดก่อนเปลี่ยนเกียร์
- ความเร็วรอบเครื่องยนต์ต่ำกว่า 700 รอบ / นาที และไม่ได้มีการเหยียบคลัชเอาไว้
- ในกรณีที่มีการเปลี่ยนเกียร์เกิดอัตราทดที่เปลี่ยนแปลงขึ้น เมื่อทำการปล่อยคลัชจะคำนวณรอบเครื่องยนต์ ถ้าต่ำกว่า 700 รอบ / นาที เครื่องยนต์จะดับ
- จังหวะปล่อยคลัชให้รถออกตัว ปล่อยเร็วเกินไป ซึ่งจังหวะปล่อยคลัชนี้จะคำนวณโดยใช้ Counter นับค่าเวลาในการปล่อยคลัช ซึ่งแต่ละเกียร์มีไม่เท่ากัน ดังตาราง

เกียร์	อัตราเร็ว (กม. / ชม.)	ค่า Counter
เกียร์ 1	น้อยกว่า 5.6	น้อยกว่า 15
เกียร์ 2	น้อยกว่า 9.5	น้อยกว่า 10
เกียร์ 3	น้อยกว่า 13.4	น้อยกว่า 8
เกียร์ 4	น้อยกว่า 18.2	น้อยกว่า 5
เกียร์ 5	น้อยกว่า 21.7	น้อยกว่า 5
เกียร์ถอยหลัง	น้อยกว่า 5.6	น้อยกว่า 15
ทุกเกียร์	มากกว่าที่กำหนดไว้	น้อยกว่า 5

ตารางที่ 5.3 แสดงจังหวะปล่อยคลัชที่ต้องตรวจสอบสำหรับแต่ละเกียร์

ค่าจังหวะปล่อยคลัชให้รถออกตัวจากตารางข้างต้น ได้มาจากการประมาณค่าตามความรู้สึกเหมาะสม ในกรณีบรรทัดสุดท้าย หมายถึง ถ้าความเร็วปัจจุบันสูงกว่าความเร็วที่กำหนดไว้ในแต่ละเกียร์ (เกียร์ 1 เป็น 5.6, เกียร์ 2 เป็น 9.5) จะต้องปล่อยคลัชอย่างช้าที่สุดให้ได้ Counter เป็น 5 แต่ถ้าหากปัจจุบันมีความเร็วต่ำกว่าที่กำหนดไว้ในแต่ละเกียร์ จะต้องปล่อยคลัชช้าลงตามน้ำหนักที่ตั้งเอาไว้ดังตารางที่แสดงไว้ข้างต้น

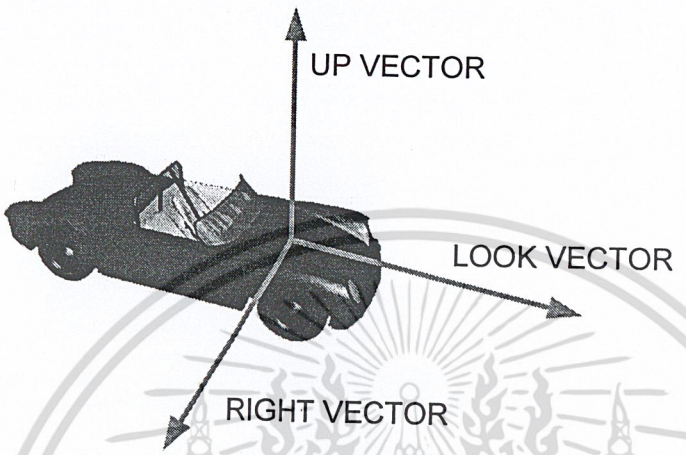
นอกจากการตรวจสอบเครื่องยนต์ดับเพื่อเพิ่มความเหมือนจริงแล้ว ยังมีส่วนอื่นๆ อีกได้แก่

- ค่าอินพุตจากคันเร่งสามารถแบ่งย่อยได้ 7 ระดับ มีระดับความเร่งที่แตกต่างกัน
- ค่าอินพุตจากเบรกสามารถแบ่งย่อยได้ 10 ระดับที่มีน้ำหนักความหน่วงไม่เท่ากัน
- ค่าอินพุตจากคลัช สามารถแบ่งย่อยได้ 30 ระดับ
- การเหยียบคันเร่งจะทำให้รถเพิ่มความเร็วเฉพาะตอนที่ปล่อยคลัช , มีการเข้าเกียร์เอาไว้ (ไม่ใช่เกียร์ว่าง) และต้องเหยียบคันเร่งอยู่ด้วย รถจึงจะสามารถเพิ่มความเร็วได้
- มีการกำหนดความเร็วสูงสุดสำหรับแต่ละน้ำหนักคันเร่งลงไปด้วย เช่น ค่าจากคันเร่งเป็น 1 จะสามารถมีความเร็วมากที่สุดได้แค่ 20 เปอร์เซ็นต์ ของความเร็วสูงสุดของเกียร์นั้นๆ ถ้ามีค่าความเร่งเป็น 2 ก็จะมีได้มากที่สุด 30 เปอร์เซ็นต์
- มีการคำนวณแรงเฉื่อยจากความเร็วรถปัจจุบัน หากไม่มีการเหยียบคันเร่งความเร็วรถจะค่อยๆ ลดลงอย่างช้าๆ
- เมื่อมีการเชนเกียร์เพิ่ม จะทำให้รอบของเครื่องยนต์ลดลง หรือ ถ้ามีการเชนเกียร์ลง จะทำให้รอบของเครื่องยนต์เพิ่มขึ้น ซึ่งจะเพิ่มขึ้นหรือลดลงมากเท่าใดขึ้นกับอัตราทดของเกียร์ ตามที่ได้กล่าวไปแล้ว
- ถ้ามีการลากเกียร์ หรือ เชนเกียร์ลงโดยที่รอบของเครื่องยนต์จะเพิ่มขึ้นจนมีรอบที่สูงมาก เครื่องยนต์จะหน่วงทำให้เกิดลดความเร็วอย่างรวดเร็วจนมาอยู่ในระดับความเร็วสูงสุดของเกียร์นั้นๆ โดยอัตโนมัติ

5.2 การจัดการกับกล้องและรถ

การจัดการกับกล้องจะจัดให้กล้องติดไปกับตัวรถ ตามรูป โดย

- ทิศด้านหน้าของรถ จะเป็นทิศของ look vector
- ทิศด้านขวาของรถ จะเป็นทิศของ right vector
- ทิศด้านบนของรถ จะเป็นทิศของ up vector



รูปที่ 5.6 แสดงการประยุกต์ใช้กล้องติดไปกับตัวรถ

และใช้หลักการTransformationของกล้องในการเคลื่อนที่ โดยในการเคลื่อนที่ของรถจะแบ่งเป็น 2 รูปแบบ คือ

1. ความสัมพันธ์ระหว่างการเคลื่อนที่รถไปข้างหน้าและการเคลื่อนที่ของกล้อง ด้วยความเร็ว V และมุมของล้อหน้าเป็น Θ เรเดียน รถจะเคลื่อนที่ใน 2 รูปแบบ คือ

1.1 เคลื่อนที่กล้องด้วยTransformation walk ในทุกๆตำแหน่งของรถ ด้วยค่า V * cos (Θ) พร้อมๆกับการเคลื่อนที่รถด้วยการ translation ตามแกน look vector

1.2 หมุนกล้องด้วยTransformation yaw ในทุกๆตำแหน่งของรถ ด้วยค่า sin (Θ) พร้อมๆกับหมุนตัวรถด้วยการrotateบน up vector ด้วยมุมต่างๆ ซึ่งเกิดจากค่า sin (Θ)

2. ความสัมพันธ์ระหว่างการเคลื่อนที่รถถอยหลังและการเคลื่อนที่ของกล้อง ด้วยความเร็ว V และมุมของล้อหน้าเป็น Θ เรเดียน รถจะเคลื่อนที่ใน 2 รูปแบบ คือ

2.1 เคลื่อนที่กล้องด้วยTransformation walk ในทุกๆตำแหน่งของรถ ด้วยค่า V * -cos (Θ) พร้อมๆกับการเคลื่อนที่รถด้วยการ translation ตามแกน look vector

2.2 หมุนกล้องด้วยTransformation yaw ในทุกๆตำแหน่งของรถ ด้วยค่า -sin (Θ) พร้อมๆกับหมุนตัวรถด้วยการrotateบน up vector ด้วยมุมต่างๆ ซึ่งเกิดจากค่า -sin (Θ)

5.3 จัดการกับ polygon ในโปรแกรม

โพลีกอนที่ใช้เป็นโมเดลในโปรแกรมจะแบ่งเป็น 2 รูปแบบ คือ

- Primitive type object
- File .x format

5.3.1 การจัดการกับ primitive object

- POINTLIST เป็นการ render เซตของจุด vertex ที่ไม่ขึ้นต่อกัน
- LINELIST เป็นการ render เซตของจุด vertex ที่ประกอบกันเป็นเส้นโดยแบ่งเซกเมนต์เป็นคู่ๆ ของ vertex
- LINESTRIP เป็นการ render เซตของจุด vertex ที่ประกอบเป็นเส้นเดียวกัน โดยจำนวน vertex จะต้องมีมากกว่า 2
- TRIANGLELIST เป็นการ render เซตของจุด vertex ที่ประกอบกันเป็นรูปสามเหลี่ยมโดยสามเหลี่ยมแต่ละรูปแยกจากกัน
- TRIANGLESTRIP เป็นการ render เซตของจุด vertex ที่ประกอบกันเป็นรูปสามเหลี่ยมต่อๆกัน
- TRIANGLEFAN เป็นการ render เซตของจุด vertex ที่ประกอบกันเป็นรูปสามเหลี่ยมคล้ายกับ trianglestrip

5.3.2 การจัดการกับรูปทรง 3 มิติ

5.3.2.1 การใช้งาน object ที่ไลบรารี direct x มีไว้ให้

เป็นรูปทรงเรขาคณิตที่กราฟฟิคไลบรารีdirect x เตรียมไว้ให้อยู่แล้ว ประกอบด้วย

- รูปทรงสี่เหลี่ยม เรียกใช้ด้วยฟังก์ชั่น D3dxCreateBox
- รูปทรงกลม เรียกใช้ด้วยฟังก์ชั่น D3dxCreateSphere
- รูปทรงกระบอก เรียกใช้ด้วยฟังก์ชั่น D3dxCreateCylinder
- รูปทรงกาน้ำ เรียกใช้ด้วยฟังก์ชั่น D3dxCreateTeapot
- รูปทรงหลายเหลี่ยม เรียกใช้ด้วยฟังก์ชั่น D3dxCreatePolygon
- รูปทรงโดนัท เรียกใช้ด้วยฟังก์ชั่น D3dxCreateTorus

5.3.2.2 การจัดการโหลดไฟล์ .x format

ขั้นตอนการโหลดไฟล์

1. create object โดยส่งพารามิเตอร์เป็นชื่อของ file.x
2. ตั้งค่าstate ต่างๆ ตามต้องการ เช่น การset material
3. ทำtransformation ให้กับแต่ละ object เพื่อนำไปวางในตำแหน่งและมุมที่ต้องการ
4. สั่ง render
5. delete object ออกจากหน่วยความจำ

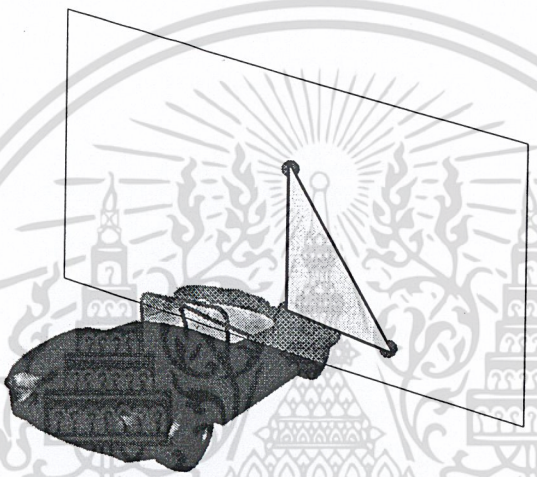
5.4 การทำกระจกโดยใช้ stencil buffer

กระจกที่จะใช้งานจะแบ่งเป็น กระจกหลัง กระจกด้านซ้าย และกระจกด้านขวา

กระจกแต่ละด้านจะสร้างจากโพลีกอนรูปสามเหลี่ยม 2 รูปติดกัน โดยมีขนาด ดังนี้

1. กระจกหลัง 12 * 5 หน่วย
2. กระจกด้านซ้าย 5 * 3 หน่วย
3. กระจกด้านขวา 5 * 3 หน่วย

จากนั้นจึงrender ส่วนที่จะใช้เป็นกระจกลงบน back buffer และ stencil buffer แล้วrender โมเดล 3 มิติ ที่จะสะท้อนลงเฉพาะในพิกเซลที่ให้แสดงเป็นกระจก โดยการหาเมทริกที่ใช้สะท้อนจากระนาบซึ่งหาได้จากจุด 3 จุด ดังรูป



รูปที่ 5.7 แสดงการใช้จุด 3 จุดเพื่อหาสมการระนาบของกระจก ประกอบด้วย จุดตำแหน่งปัจจุบัน จุดที่ได้จาก right vector และจุดที่ได้จาก left vector

ขั้นตอนการใช้งาน stencil buffer

1. เริ่มต้นด้วยการร้องขอให้มีการใช้ stencil buffer โดยจะต้องแชร์พื้นที่การทำงานร่วมกับ depth buffer จึงอาจทำให้ประสิทธิภาพโดยรวมลดลงได้ แบ่งได้ 3 รูปแบบ คือ

1) D3DFMT_D24S8 เป็นการร้องขอใช้ 32 บิท depth/stencil buffer โดย 24 บิทสำหรับ depth buffer และ 8 สำหรับ stencil buffer

2) D3DFMT_D24X4S4 เป็นการร้องขอใช้ 32 บิท depth/stencil buffer โดย 24 บิทสำหรับ depth buffer 4 สำหรับ stencil buffer และ อีก 4 บิทไม่ได้ใช้งาน

3) D3DFMT_D15S1 เป็นการร้องขอใช้ 16 บิท depth/stencil buffer โดย 15 บิทสำหรับ depth buffer และ 1 สำหรับ stencil buffer

2. ก่อนและหลังใช้งานจะต้องมีการ enable/disable การใช้ stencil buffer และ จะต้องมีการเคลียร์ทุกครั้งการเรียกใช้งานเป็นดังรูป

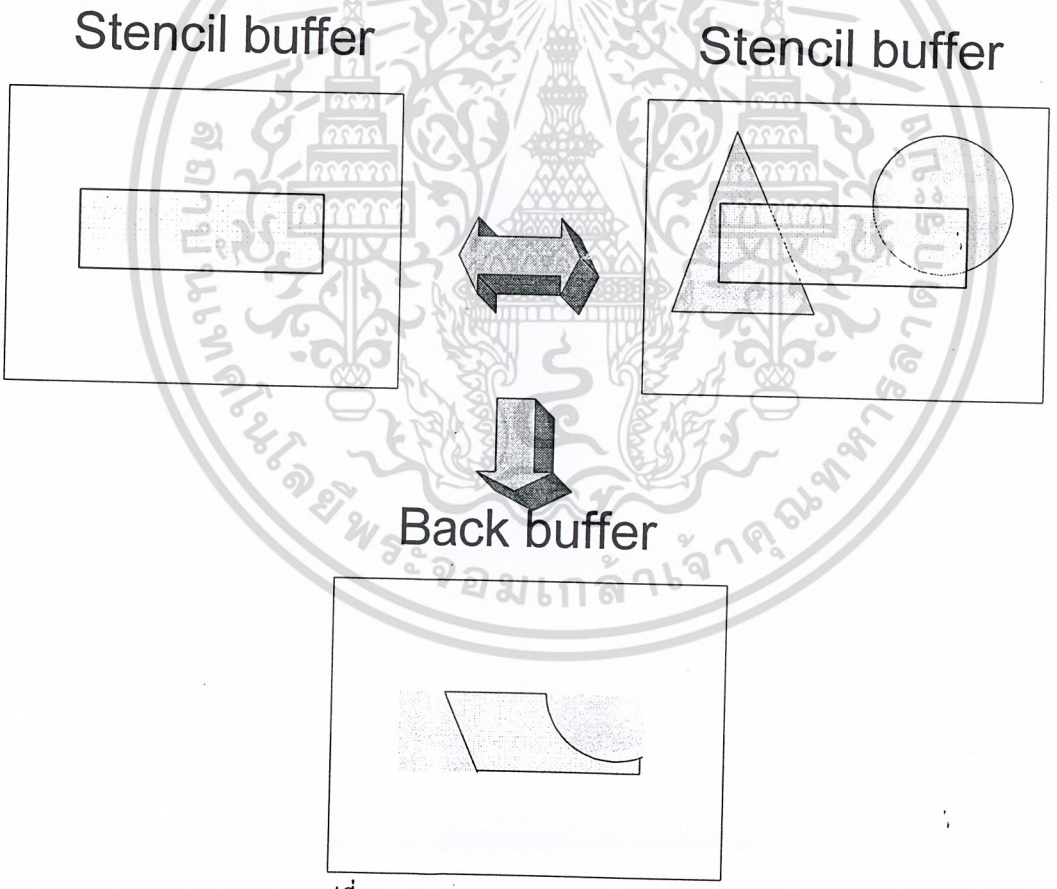
```
Device->SetRenderState(D3DRS_STENCILENABLE,    true);
... // do stencil work
Device->SetRenderState(D3DRS_STENCILENABLE,    false);
```

```
Device->Clear(0, 0,
     D3DCLEAR_TARGET | D3DCLEAR_ZBUFFER | D3DCLEAR_STENCIL,
     0xff000000, 1.0f, 0 );
```

รูปที่ 5.8 แสดงฟังก์ชั่นการใช้งาน และการเคลียร์ stencil buffer

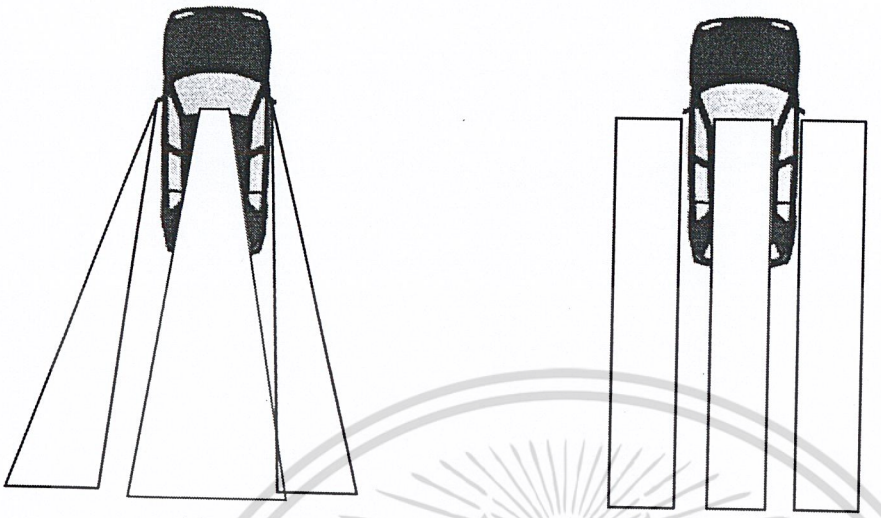
3. render ส่วนที่จะทำเป็นกระจกลงบน stencil buffer

4. render ส่วนที่จะสะท้อนซึ่งแต่ลพิกเซลที่จะแสดงผลของส่วนนี้จะต้องนำไปทำโอเปอเรชั่น and กับ stencil buffer เพื่อให้ render ออกมาเฉพาะในกระจกเท่านั้น



รูปที่ 5.9 แสดงการทำงานของ stencil buffer

และเนื่องจากการคูณด้วยเมททริกการสะท้อน viewing volume ที่ได้จึงเป็นรูปสี่เหลี่ยมซึ่งต่างจาก viewing volume ที่เกิดจากกระจกรถยนต์จริงๆ เพราะฉะนั้นภาพที่ได้อาจไม่เหมือนจริงบ้าง ดังรูป

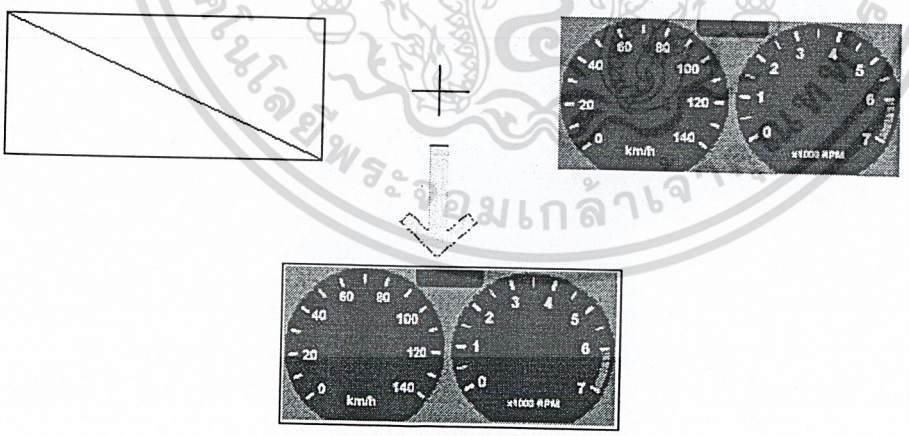


(a) (b)

รูปที่ 5.10 แสดง viewing volume ของกระจก ด้านซ้าย (a) เป็นกระจกในความเป็นจริง ด้านขวา (b) เป็นกระจกที่ใช้ในโครงงาน

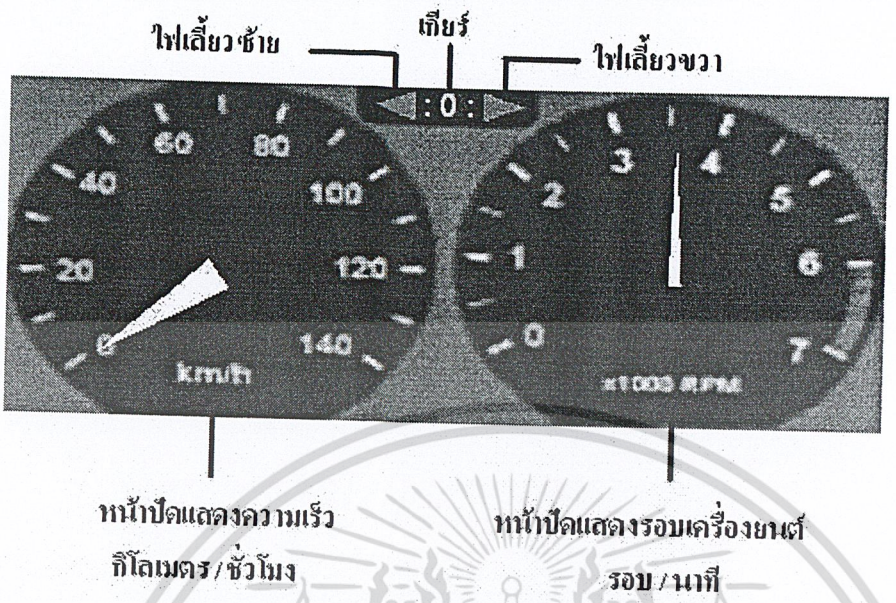
5.5 การทำแผงหน้าปัดรถ

เริ่มต้นด้วยการสร้างโพลีกอนรูปสามเหลี่ยมติดกัน 2 รูป และโหลดtextureที่จะใช้มาทำการแม็ป ดังรูป



รูปที่ 5.11 การ map แผง console ลงบน polygon สามเหลี่ยมสองรูป

จากนั้นจึงสร้างโพลีกอนรูปสามเหลี่ยมสีขาวเพื่อใช้แทนเข็ม , โพลีกอนรูปสามเหลี่ยมสีเขียวเพื่อใช้แทนไฟเลี้ยว



รูปที่ 5.12 ส่วนประกอบของ console หน้ารถ

การหมุนเข็มหน้าปัดจะหมุนอยู่บนแกน Z ซึ่งจะต้องสัมพันธ์กับความเร็วและค่ารอบเครื่องยนต์ แล้วจึงtranslateขึ้นมาในตำแหน่งบนหน้าปัด ดังรูป

รูปที่ 5.13 แสดงการทำ transformation ของเข็มวัดความเร็ว

5.6 การใช้งาน text

จะแบ่งการใช้งานหลักๆ ได้ 2 รูปแบบ คือ

1. ตัวอักษรแบบ 2 มิติ

เป็นการ render ตัวอักษรลงไปใน back buffer โดยตรง โดยจะต้องระบุตำแหน่งพิกเซล (X,Y) สี และรูปแบบตัวอักษร โดยในโครงงานได้ประยุกต์ใช้เพื่อแสดงผลเกียร์

2. ตัวอักษรแบบ 3 มิติ

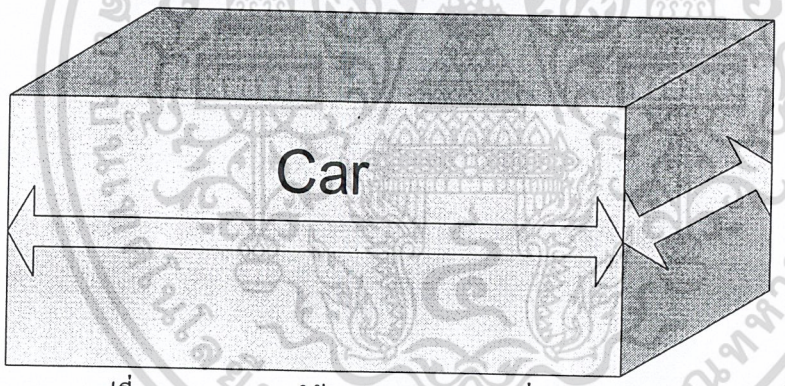
เป็นการ render ตัวอักษรลงไปใน world space

5.7 การตรวจสอบการชน

การตรวจสอบการชนระหว่างรถยนต์ซึ่งมีรูปร่างเป็นทรงสี่เหลี่ยมกับวัตถุอื่นๆรอบข้าง ในโครงงานนี้จะตรวจสอบการชนเฉพาะขอบกั้นรอบๆถนน ซึ่งทั้งหมดจะใช้ bounding box เป็นรูปทรงสี่เหลี่ยม

การใช้งาน function isHit() เพื่อตรวจสอบการชน ซึ่งจะ return ค่า Boolean เป็น true เมื่อมีการชน การตรวจสอบทำโดยเก็บข้อมูลตำแหน่งด้านหน้าข้างซ้าย ด้านหน้าข้างขวา ด้านหลังข้างซ้ายและด้านหลังข้างขวา และมีการอัปเดทข้อมูลตลอดทุกๆ การเคลื่อนไหวของรถ ซึ่งข้อมูลต่างๆของรถจะถูกเก็บอยู่ใน class CCar

การตรวจสอบรอบๆรถทำได้โดยใช้ ray ยิงรอบๆ ทั้ง 4 ด้าน ทั้งไปและกลับ และเมื่อ ray ยิงเข้าไปใน bounding box ของวัตถุอื่นก็ถือว่าชน ตามรูป



รูปที่ 5.14 แสดงการใช้ ray ยิงรอบๆตัวรถเพื่อใช้ตรวจสอบการชน

5.8 การจัดการกับเสียง

การจัดการกับเสียงจะใช้ class CMusicManager class CMusicSegment จะใช้ในการเปิดและโหลดไฟล์เสียงมาเก็บลงในเซ็กเมนต์ ซึ่งชนิดของไฟล์ที่สนับสนุนได้แก่ wav หรือ MIDI การเล่นเสียงแบ่งเป็น 2 ระดับ คือ

- Primary segment
- Secondary segment

5.9 การสอบวัดผลความรู้ด้านกฎจราจร

การสอบขอใบอนุญาตขับขี่รถยนต์ต้องสอบผ่านทั้งภาคทฤษฎีและภาคปฏิบัติ ผู้สอบต้องสอบข้อเขียนผ่านก่อน จึงสามารถสอบภาคปฏิบัติได้

5.9.1 คุณสมบัติผู้สอบขอใบอนุญาตขับขี่รถยนต์และกฎจราจรที่เกี่ยวข้อง

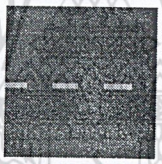
5.9.1.1 คุณสมบัติผู้สอบขอใบอนุญาตขับขี่รถยนต์

- ต้องมีอายุไม่ต่ำกว่า 18 ปีบริบูรณ์
- ไม่เป็นผู้ที่มีร่างกายบกพร่อง เช่น ตาบอด ตาบอดสี หรือ หูหนวก

5.9.1.2 กฎจราจรที่เกี่ยวข้อง

เครื่องหมายจราจรบนพื้นทางประเภทบังคับ

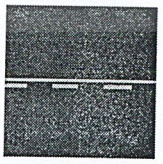
1) เส้นแบ่งทิศทางการจราจรปกติ



รูปที่ 5.15 เส้นแบ่งทิศทางการจราจรปกติ

เป็นเส้นแสดงการแบ่งแยกการจราจรของยวดยานที่มีทิศตรงกันข้าม ให้ขับรถไปตามด้านซ้ายของเส้นแบ่งทิศทางการจราจรยกเว้นในกรณีที่ต้องการแซงขึ้นหน้ารถคันอื่น

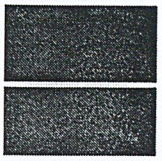
2) เส้นแบ่งทิศทางการจราจรห้ามแซงเฉพาะด้าน



รูปที่ 5.16 เส้นแบ่งทิศทางการจราจรห้ามแซงเฉพาะด้าน

ยวดยานที่ขับอยู่ด้านซ้ายของเส้นทึบ ห้ามมิให้ขับรถผ่านหรือคร่อมเส้นโดยเด็ดขาด ส่วนยวดยานที่ขับอยู่ทางด้านเส้นประ เมื่อเห็นว่าปลอดภัยสามารถแซงขึ้นหน้ารถคันอื่นหรือล้ำออกไปทางขวาของเส้นได้

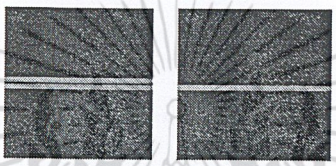
3) เส้นห้ามเปลี่ยนช่องจราจรหรือเส้นห้ามเปลี่ยนช่องเดินรถ



รูปที่ 5.17 เส้นห้ามเปลี่ยนช่องจราจรหรือเส้นห้ามเปลี่ยนช่องเดินรถ

แบ่งทางเดินรถหรือทางจราจรในทิศทางเดียวกันให้เป็นช่องทางเดินรถหรือช่องจราจร ให้ขับภายในช่องเดินรถห้ามขับผ่านหรือคร่อมเส้น

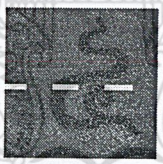
4) เส้นแบ่งทิศทางจราจรห้ามแซง



รูปที่ 5.18 เส้นแบ่งทิศทางจราจรห้ามแซง

ให้ขับรถไปตามด้านซ้ายของเส้น ห้ามมิให้ขับรถผ่านหรือคร่อมเส้นโดยเด็ดขาด

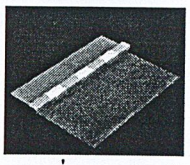
5) เส้นแบ่งช่องจราจรหรือเส้นช่องเดินรถปกติ



รูปที่ 5.19 เส้นแบ่งช่องจราจรหรือเส้นช่องเดินรถปกติ

แบ่งช่องเดินรถหรือทางจราจรที่มีทิศทางเดียวกัน ให้ขับรถภายในช่องจราจรหรือช่องเดินรถ ห้ามขับรถคร่อมเส้นเว้นแต่จะเปลี่ยนช่องจราจรหรือช่องเดินรถ

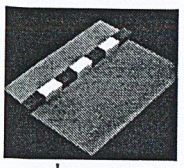
6) เครื่องหมายห้ามจอดรถ



รูปที่ 5.20 เครื่องหมายห้ามจอดรถ

ห้ามจอดรถทุกชนิดระหว่างแนวนั้น เว้นแต่การหยุดรับส่งคนหรือสิ่งของชั่วขณะ ซึ่งต้องกระทำโดยมิชักช้า

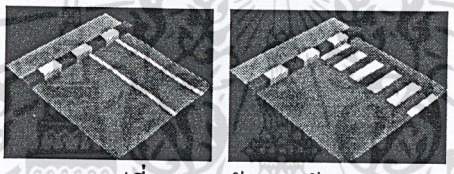
7) เครื่องหมายห้ามหยุดรถ



รูปที่ 5.21 เครื่องหมายห้ามหยุดรถ

ห้ามหยุดรถหรือจอดรถทุกชนิดตรงแนวนั้นเป็นอันขาด

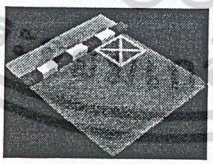
8) เส้นทางข้าม



รูปที่ 5.22 เส้นทางข้าม

ผู้ขับรถทุกชนิดจะต้องขับรถให้ช้าลงและพร้อมจะหยุดรถได้ทันท่วงที เมื่อมีคนข้ามถนนในเขตทางข้ามถนน ให้ผู้ขับรถหยุดรถก่อนถึงเส้นแนวหยุดหรือเส้นให้ทาง เมื่อคนเดินข้ามถนนไปแล้วจึงเคลื่อนรถต่อไปได้

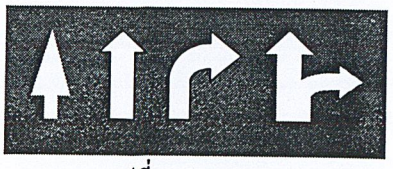
9) เส้นทแยงมุมห้ามหยุดรถ



รูปที่ 5.23 เส้นทแยงมุมห้ามหยุดรถ

ห้ามหยุดรถทุกชนิดภายในกรอบเส้นทแยง ห้ามหยุดรถ ยกเว้นรถที่หยุดเพื่อเลี้ยวขวา

10) ลูกศร



รูปที่ 5.24 ลูกศร

เมื่อมีลูกศรปรากฏอยู่ในช่องจราจรหรือช่องเดินรถใด ให้ผู้ขับรถที่อยู่ในช่องจราจรหรือช่องเดินรถนั้นปฏิบัติตามเครื่องหมาย

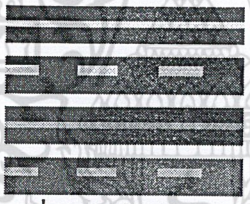
11) ข้อความบังคับบนพื้นทาง

รูปที่ 5.25 ข้อความบังคับบนพื้นทาง

ผู้ขับรถต้องปฏิบัติตามข้อความนั้นๆ

เครื่องหมายจราจรบนพื้นทางประเภทเตือน

1) เส้นขอบทาง



รูปที่ 5.26 เส้นขอบทาง

2) ข้อความเตือนหรือแนะนำบนพื้นทาง

รูปที่ 5.27 ข้อความเตือนหรือแนะนำบนพื้นทาง

ให้คนขับรถหรือคนเดินเท้าปฏิบัติตาม และระมัดระวังการใช้ช่องจราจรหรือช่องเดินรถให้ถูกต้อง

เครื่องหมายจราจรประเภทบังคับ

หยุด

ให้ทาง


ห้ามรถยนต์


ห้ามแซง


ห้ามเข้า


ห้ามกลับรถ


ห้ามเลี้ยวซ้าย


ห้ามเลี้ยวขวา


ให้รถสวนทางมาก่อน


ห้ามรถบรรทุก


ห้ามรถจักรยานยนต์


ห้ามรถยนต์สามล้อ


ห้ามรถสามล้อ


ห้ามรถจักรยาน


ห้ามล้อเลื่อนลากเข็น


ห้ามรถแทรกเตอร์


ห้ามรถจักรยานยนต์และรถยนต์


ห้ามรถจักรยานรถสามล้อ และล้อเลื่อนลากเข็น


ห้ามใช้เสียง


ห้ามคน


ห้ามจอดรถ


ห้ามหยุดรถ

หยุดตรวจ

จำกัดความเร็ว

ห้ามรถหนักเกินกำหนด

ห้ามรถกว้างเกินกำหนด

ห้ามรถสูงเกินกำหนด


ให้เดินรถทางเดียวไปข้างหน้า


ทางเดินรถทางเดียวไปทางซ้าย


ทางเดินรถทางเดียวไปทางขวา


ให้ชิดซ้าย


ให้ชิดขวา

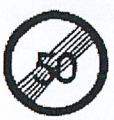

ให้ไปทางซ้ายหรือทางขวา


ให้เลี้ยวซ้าย


ให้เลี้ยวขวา

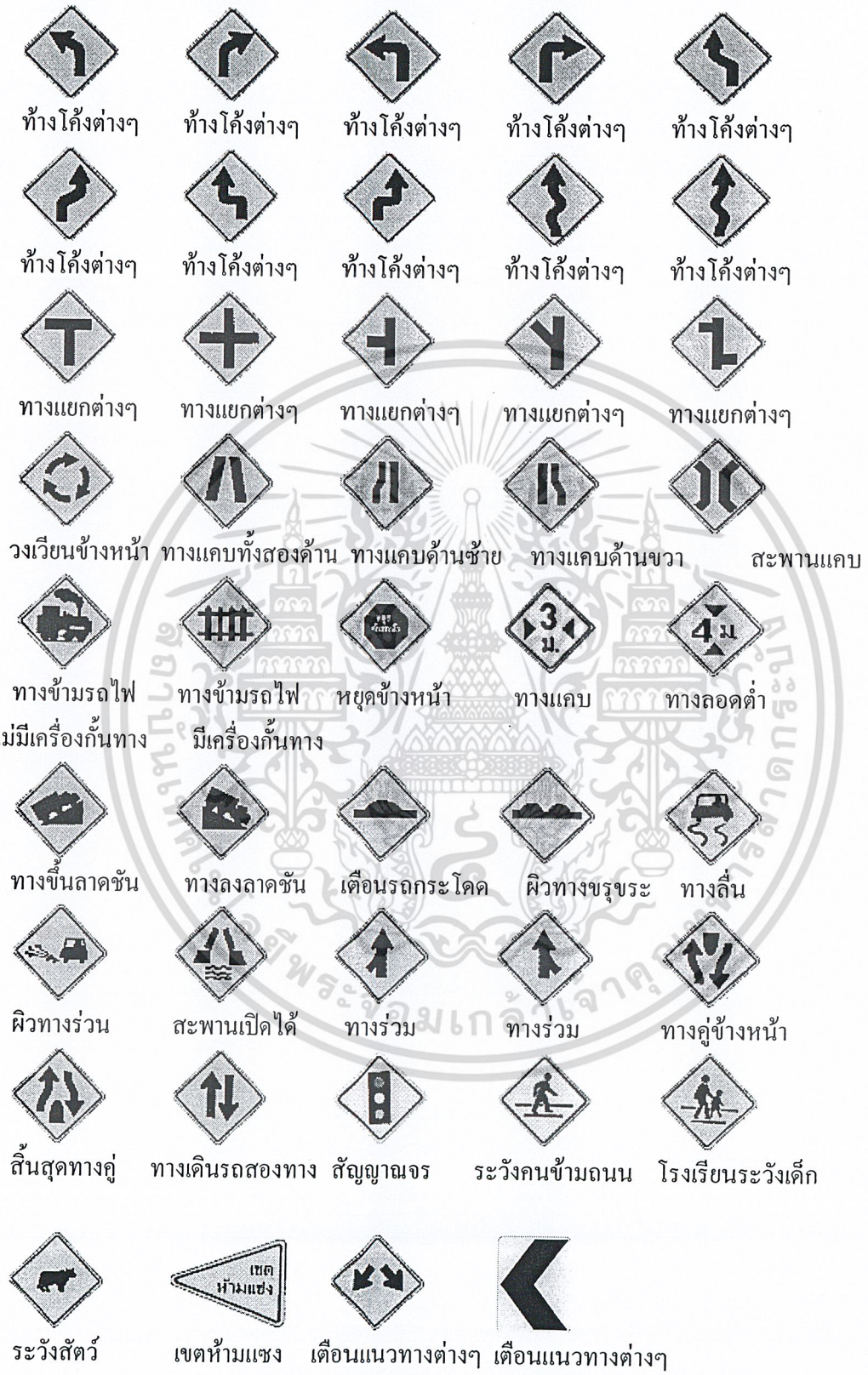

ให้เลี้ยวซ้ายหรือเลี้ยวขวา


วงเวียน


สุดเขตบังคับจำกัดความเร็ว

รูปที่ 5.28 เครื่องหมายจราจรประเภทบังคับ

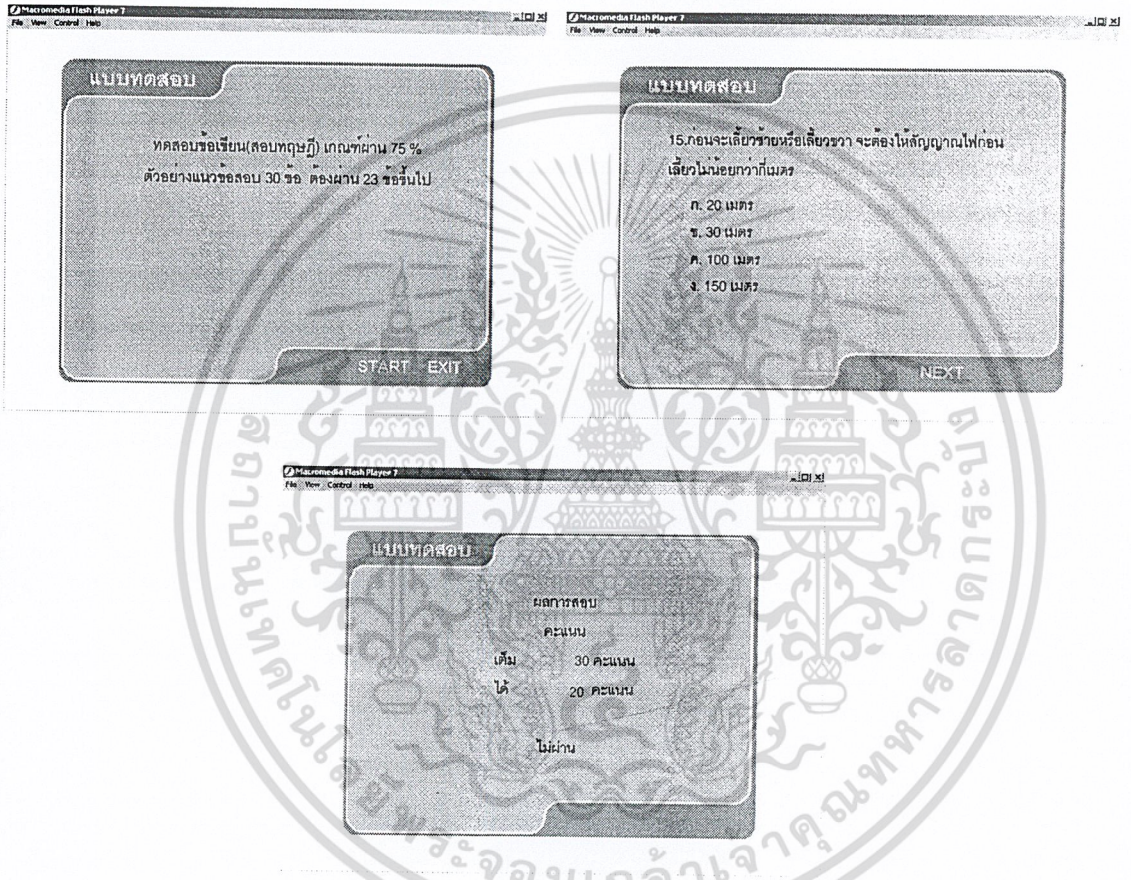
เครื่องหมายจราจรประเภทป้ายเตือน



รูปที่ 5.29 เครื่องหมายจราจรประเภทป้ายเตือน

5.9.2 ลักษณะข้อสอบและเกณฑ์การให้คะแนน

ในโครงงานนี้ได้จัดทำข้อสอบที่ใช้ในการสอบภาคทฤษฎี หรือ การสอบข้อเขียน เป็นข้อสอบปรนัย 30 ข้อ ต้องผ่านเกณฑ์ 75 เปอร์เซ็นต์ หรือ ได้ไม่ต่ำกว่า 23 ข้อ ดังเช่นเกณฑ์ที่ใช้ในการสอบจริง โครงงานนี้พัฒนาข้อสอบด้วยโปรแกรม Macromedia Flash MX ซึ่งสามารถประยุกต์ใช้บนเว็บหรือเป็นโปรแกรมประเภทไฟล์ Shockwave ซึ่งต้องติดตั้ง Flash Player ก่อนจึงสามารถเปิดได้ ในการพัฒนาต่อไปสามารถเพิ่มจำนวนชุดข้อสอบหรือใช้งานร่วมกับระบบฐานข้อมูลได้



รูปที่ 5.30 ตัวอย่างโปรแกรมข้อสอบ

บทที่ 6
สรุปและแนวทางการพัฒนา

6.1 บทสรุป

การดำเนินงานที่ผ่านมาเริ่มต้นจากแนวคิดที่จะเพิ่มทางเลือกให้กับประชาชนที่ไม่ได้รับความสะดวกในเรื่องของเวลาและสถานที่ในการฝึกหัดขับรถยนต์เพื่อให้สามารถฝึกหัดการขับรถภายในที่พักอาศัยได้อย่างสะดวก ผลการดำเนินงานโดยรวม ได้แก่

- การออกแบบ ใช้การติดต่อระหว่างอุปกรณ์ต่อพ่วงและคอมพิวเตอร์ผ่านเกมพอร์ตซึ่งง่ายต่อการประยุกต์ใช้
- เครื่องมือที่ใช้พัฒนา ในโครงงานได้ประยุกต์ใช้งานกราฟฟิคไลบรารี DirectX เพียงเบื้องต้นเท่านั้น แต่ DirectX ยังคงมีส่วนประกอบอีกมากซึ่งใช้งานได้รูปแบบซึ่งเหมาะสมในการศึกษาค้นคว้าต่อไป
- การทำงานของระบบ และการคำนวณต่างๆ ภายในโปรแกรม การเคลื่อนที่ในแนวราบทำงานได้ค่อนข้างสมบูรณ์ซึ่งสัมพันธ์กับปริมาณต่างๆ เช่น ความเร็ว ความเร่ง ซึ่งใช้สูตรการคำนวณที่ไม่ยาก การเคลื่อนที่บนสะพานอาจจะยังไม่สมจริงมากเพราะในการโหลดไฟล์ .X ที่เป็นสะพานนั้นเมื่อทำ transformation แล้วไม่สามารถวัดขนาดที่แน่นอนได้
- การแสดงผล เนื่องจากโปรแกรมใช้การคำนวณสูงและเวลาที่จำกัดในการพัฒนาทำให้ source code ยังไม่ได้รับการ optimize ที่เพียงพอจึงแสดงผลได้ไม่มากที่สุดเพียง 19 เฟรมต่อวินาที
- การใช้งานระบบ จะเหมือนกับการขับรถยนต์จริงเกือบทุกอย่างทำให้ผู้ที่เคยขับรถมาก่อนสามารถใช้ได้เร็ว
- ปัญหาและอุปสรรค
  - การใช้ตัวต้านทานต่อผ่านเกมพอร์ตให้สัญญาณขาเข้าที่ไม่ค่อยคงที่อาจให้ผลในการตรวจจับเกียร์ที่คาดเคลื่อน
  - การตั้งค่าแกนของเกมพอร์ตในแต่ละเครื่องต่างกับ เพราะฉะนั้นเมื่อมีการเปลี่ยนคอมพิวเตอร์ที่ใช้งานจะต้อง calibrate ใหม่ทุกครั้ง
  - การประมวลผลในโปรแกรมใช้การคำนวณสูงและยังไม่ผ่านการ optimize การแสดงผลจึงช้า

6.2 แนวทางการพัฒนา และ ข้อแนะนำ

เนื่องจากข้อจำกัดในเรื่องของเวลาจึงทำให้โครงงานนี้มีข้อจำกัดหลายอย่าง จึงมีข้อเสนอแนะเพื่อให้ผู้ที่สนใจได้พัฒนาต่อไป ได้แก่

- การตรวจสอบการชนของรถยังไม่ละเอียดมากพอ เพราะจะมีแค่รอบๆตัวรถแนวเดียวเท่านั้น เพราะฉะนั้นจึงควรจะเพิ่มการตรวจสอบในกรณีที่วัตถุลอดผ่านใต้ท้องรถ หรือ วัตถุลอยมาบนอากาศกระทบกับหลังคารถ

- การทำงานของโปรแกรมยังคงเป็นแบบง่ายๆ ดังจะสังเกตได้ว่าส่วนที่เคลื่อนไหวมีเฉพาะส่วนตัวรถและอุปกรณ์ที่อยู่ภายในรถเท่านั้น เพราะฉะนั้นสามารถที่จะนำเทคโนโลยีและหลักการทางด้านปัญญาประดิษฐ์มาร่วมด้วยได้เพื่อให้โปรแกรมสมบูรณ์ยิ่งขึ้น

- สามารถพัฒนาให้นอกเหนือจากการทดสอบการขอใบอนุญาตขับขี่ด้วยอุปกรณ์สำหรับขับเคลื่อนเสมือนจริงไปเป็นรูปแบบของเกมส์การแข่งรถที่เสมือนจริงได้

- เทคโนโลยีด้านเสียงเป็นการประยุกต์ใช้อย่างง่ายๆ ต่อไปสามารถที่จะพัฒนาระบบเสียง 3 มิติเข้ามาใช้ด้วยได้

- การใช้ Game Port ในการเชื่อมต่อ ทำให้ไม่สามารถทำการตอบสนองทาง Hardware ให้กับผู้ใช้ได้ เนื่องจากตัว Port เองใช้สำหรับรับค่าเพียงอย่างเดียว จึงทำการตอบสนองได้เพียงบน Software เท่านั้น นอกจากนี้การนำชุดอุปกรณ์ไปต่อกับคอมพิวเตอร์เครื่องอื่น ต้องมีการ Calibrate ค่าใหม่ทุกครั้ง

- สัญญาณอินพุตที่รับผ่านทางขาที่เป็น Analog-To-Digital Converter ทั้ง 4 ขา มีอาการสั่นไหวเล็กน้อยอยู่เสมอ ดังนั้นจึงไม่สามารถนำความแตกต่างของสัญญาณทั้งหมดมาใช้ได้ วิธีแก้ไขคือ เปลี่ยนมาใช้การตรวจสอบการเปลี่ยนแปลงค่าเป็นช่วงๆ แทนที่จะตรวจสอบค่าโดยละเอียดทุกค่า

บรรณานุกรม

Frank D. Luana Introduction to 3D GAME Programming with DirectX 9.0
Wordware Publishing, Inc 2003
Peter Walsh Advance 3D Game Programming with DirectX 9.0
Wordware Publishing, Inc 2003
Andre LaMothe Programming Role Playing Games with DirectX 8.0
Premier Press, Inc 2002

ภาคผนวก

1. สาเหตุที่เลือกใช้เครื่องมือชนิดต่างๆ

1.1 สาเหตุที่เลือกใช้ DirectX ในการพัฒนา

จุดเริ่มต้นของ DirectX เริ่มต้นมาจาก Microsoft ได้พัฒนาเครื่องมือสำหรับผู้พัฒนาเกมส์ ชื่อว่า The Game SDK ซึ่งเป็นเวอร์ชั่นแรกของ DirectX และได้พัฒนาให้สามารถใช้ความสามารถของ win32 API ได้ เช่น การทำมัลติเทรด การใช้ TCP/IP สแตก และมี User Interface มากมาย

สำหรับผู้ที่ไม่เคยใช้ก็ยังมี OpenGL ซึ่งเป็นกราฟฟิก API ซึ่งเป็นของบริษัท Silicon Graphics แต่ก็มีข้อเสียหลายอย่าง เช่น

- OpenGL ประกอบด้วยฟังก์ชั่นการทำงานมากมาย การใช้งานอย่างง่ายๆ แต่จำเป็นจะต้องจัดการสิ่งต่างๆมากมายเพื่อให้แน่ใจว่าโปรแกรมสามารถทำงานได้ถูกต้อง

- เนื่องจากการมี device driver ทำให้แต่ละบริษัทที่ผลิตการ์ด 3 มิติ จะต้องสร้างขึ้นมาเพื่อสนับสนุนการใช้งานทุกๆ ฟังก์ชั่นการทำงานของ OpenGL ด้วยการผสมผสานกับ OpenGL device driver ซึ่งสิ่งนี้เป็นการยากที่จะสร้างขึ้นมาให้ถูกต้องสมบูรณ์และครบถ้วน เพราะฉะนั้นประสิทธิภาพที่ได้จากอุปกรณ์ฮาร์ดแวร์จึงต่างกันออกไปขึ้นอยู่กับคุณภาพของ driver ต่างจาก DirectX ที่สามารถเข้าถึงอุปกรณ์ฮาร์ดแวร์ที่แตกต่างกันได้อย่างรวดเร็ว

- ฟังก์ชั่นการทำงานของ OpenGL ได้มาจากการประชุมตกลงกันของกรรมการ ขณะที่ DirectX นั้นควบคุมโดย Microsoft ซึ่งเหตุผลข้อนี้จะดีหรือไม่ก็ขึ้นอยู่กับแต่ละคนด้วย

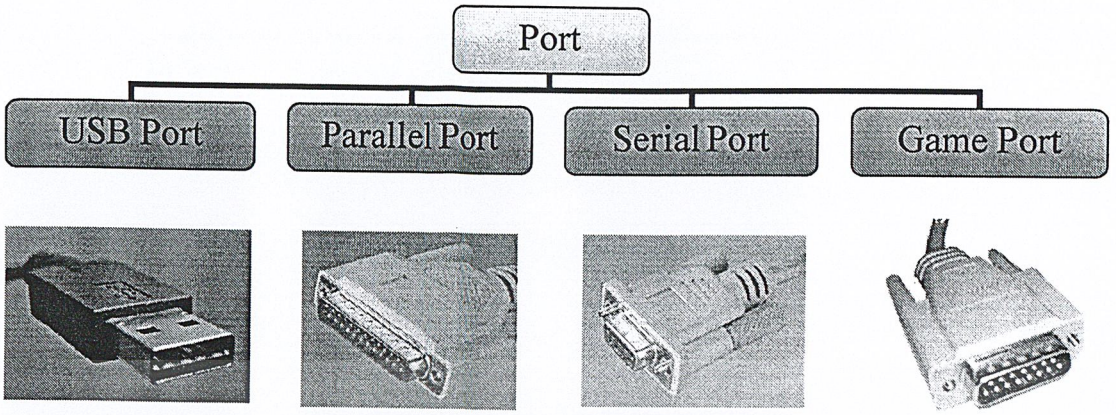
1.2 สาเหตุที่เลือกใช้ 3D Studio Max

3D Studio Max หรือ 3ds Max เป็นโปรแกรมสร้างกราฟฟิก 3 มิติ ที่ได้รับความนิยมเป็นอย่างมาก และนำไปใช้กับงานหลายประเภท เช่น งานออกแบบ, โฆษณา, ภาพยนต์และเกมส์ เป็นต้น จึงนำมาสร้างวัตถุและภาพแอนิเมชั่น 3 มิติ เพื่อใช้ในโครงงาน

เนื่องจากการนำวัตถุ 3 มิติมาใช้กับ Direct X ต้องเปลี่ยนชนิดไฟล์ จากไฟล์ของ 3ds Max (.3ds) เป็น X ไฟล์ (.x) จึงใช้โปรแกรม Quick 3D เพื่อแปลงชนิดไฟล์

1.3 สาเหตุที่เลือกใช้ Game Port

Port มาตรฐานที่นิยมใช้เชื่อมต่ออุปกรณ์ทั่วไปในคอมพิวเตอร์ล้วนแต่สามารถประยุกต์ใช้นำมาติดต่อกับ Joystick ได้แทบทั้งสิ้น ซึ่ง Port ที่สามารถใช้ติดต่อกับ Joystick ในปัจจุบันมีดังนี้



รูปแสดงพอร์ตที่สามารถนำมาใช้งานกับ Joystick

ซึ่งแต่ละ Port ก็จะมีรูปแบบการเชื่อมต่อแตกต่างกันออกไป และมีข้อดีข้อเสียแตกต่างกัน คุณสมบัติของแต่ละ Port มีดังนี้

USB Port เป็นพอร์ตที่ปัจจุบันนิยมใช้กันมาก มีอัตราการรับส่งข้อมูลด้วยความเร็วสูง และง่ายต่อการติดตั้ง เพราะมีคุณสมบัติ Plug & Play สามารถใช้งานได้ทันทีไม่ต้องลง Driver นำมาใช้งานกับอุปกรณ์ได้แทบทุกชนิด แต่การสร้างอุปกรณ์ที่เชื่อมต่อด้วย USB จะซับซ้อนกว่าทุกพอร์ตชนิดอื่นทั้งหมด

Parallel Port เป็นพอร์ตที่เรียกว่าพอร์ตขนาน นิยมใช้เป็นพอร์ตสำหรับ Printer ในสมัยก่อน สามารถทั้งรับและส่งข้อมูลได้ในตัว แต่จะรับส่งข้อมูลในลักษณะ Digital เป็นหลัก จึงไม่เหมาะที่จะนำมาใช้ทำโครงงาน ไม่จำเป็นต้องอาศัย Micro Controller ในการทำงาน แต่ต้องเขียน Driver ทำงานเอง

Serial Port เป็นพอร์ตที่เรียกว่าพอร์ตอนุกรม เป็นพอร์ตที่นิยมใช้มากในสมัยก่อน ส่งข้อมูลได้ที่อัตราสูงสุด 115,200 บิต / วินาที สามารถนำมาประยุกต์ใช้เป็น Joystick ได้ แต่ต้องมี Micro Controller ในการควบคุม สามารถรับส่งข้อมูลได้ทั้ง 2 ทิศทาง นอกจากนี้ต้องเขียนโปรแกรมรับส่งข้อมูลทั้งฝั่ง Joystick กับทั้งฝั่งคอมพิวเตอร์

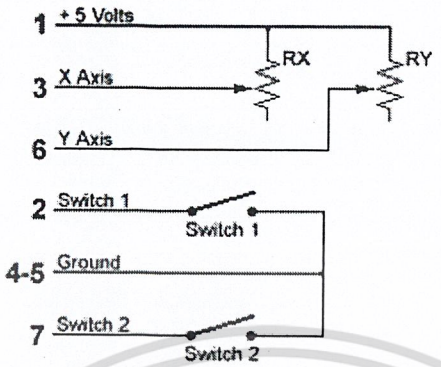
Game Port เป็นพอร์ตที่ออกแบบมาเพื่อใช้สำหรับติดต่อกับ Joystick โดยเฉพาะ ง่ายต่อการใช้งานไม่ต้องมี Micro Controller และไม่ต้องมีวงจรเป็นพิเศษ ตัวพอร์ตมี Analog-To-Digital Converter ในตัวถึง 4 ตัวและ รับการต่อสวิตช์ได้อีก 4 ตัว ไม่จำเป็นต้องมีการใช้ Driver เพิ่มเติม และไม่ต้องเขียนโปรแกรมใดๆ สามารถนำมาใช้ทำโครงงานได้เป็นอย่างดี แต่ข้อจำกัดคือ สามารถส่งจาก Joystick เข้าคอมพิวเตอร์ได้อย่างเดียว แต่คอมพิวเตอร์ส่งกลับมาไม่ได้

2 ชนิดของ Joystick

ปัจจุบันมีการนำ Joystick เข้ามาใช้ร่วมกับ Multimedia กันอย่างแพร่หลาย ซึ่ง Joystick จะแบ่งออกเป็น 2 ประเภทหลักๆ ได้แก่

2.1 Analog Joystick

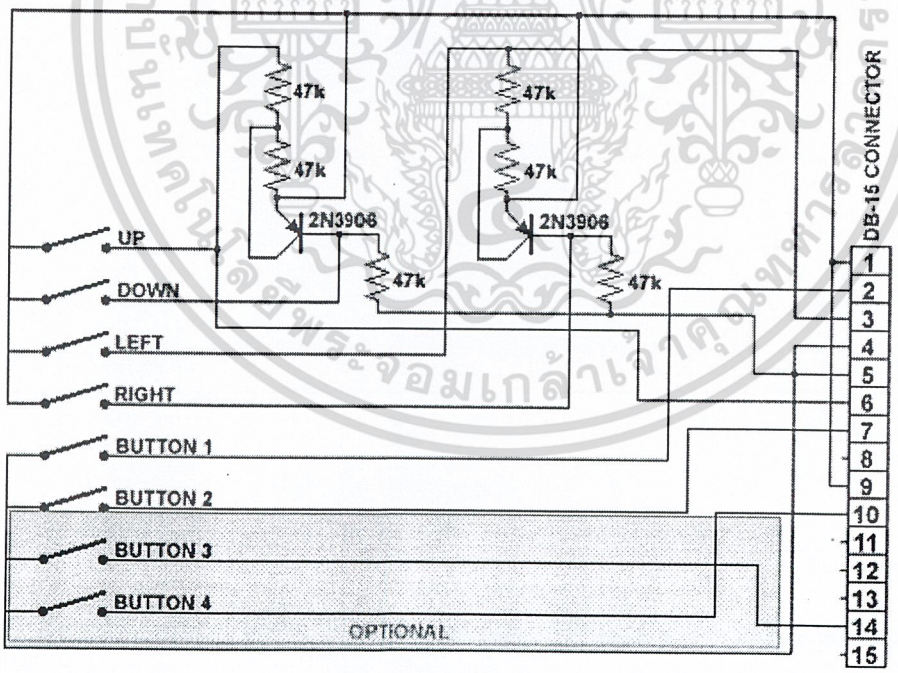
เป็น Joystick แบบดั้งเดิม ใช้ตั้งแต่ในสมัย PC รุ่นแรกๆ ประกอบด้วย Switch และ Potentiometer 100k ชนิด Linear อย่างละ 2 ตัว



รูปแบบการต่อ Analog Joystick

2.2 Digital Joystick

Joystick ชนิดนี้เริ่มต้นใช้กันในสมัย ทศวรรษที่ 80 โดยบริษัท Nintendo ซึ่งได้นำ Digital Switch เข้ามาแทนในส่วนการทำงานของ Potentiometer ซึ่งได้นิยมใช้กันมาจนถึงปัจจุบัน ซึ่งให้ค่าได้เที่ยงตรงแม่นยำมากกว่า เนื่องจากนำ Switch ที่ต่อวงจร Digital เข้ามาแทนการใช้ Potentiometer



รูปการต่อ Digital Joystick

เนื่องจาก Joystick ชนิด Digital นั้นบังคับทิศทางโดยการกด Switch ที่เชื่อมต่ออยู่กับวงจร Digital ซึ่งสามารถตรวจสอบทิศทาง ขึ้น, ลง, ซ้าย, ขวา ได้เพียงทิศทางละ 1 ระดับ จึงไม่เพียงพอต่อการใช้งานสำหรับ

โครงงานนี้ เนื่องจากโครงงานนี้ต้องการอุปกรณ์ที่สามารถให้ค่า Output ได้หลากหลายค่า จึงจำเป็นต้องใช้การเชื่อมต่อ Joystick แบบ Analog ที่ใช้ตัวต้านทานปรับค่าได้

3 รายละเอียดของคลาส CD3DApplication

```
class CD3DApplication
{
protected:
    CD3DEnumeration   m_d3dEnumeration;
    CD3DSettings      m_d3dSettings;

    // Internal variables for the state of the app
    bool          m_bWindowed;
    bool          m_bActive;
    bool          m_bDeviceLost;
    bool          m_bMinimized;
    bool          m_bMaximized;
    bool          m_bIgnoreSizeChange;
    bool          m_bDeviceObjectsInited;
    bool          m_bDeviceObjectsRestored;

    // Internal variables used for timing
    bool          m_bFrameMoving;
    bool          m_bSingleStep;

    // Internal error handling function
    HRESULT DisplayErrorMsg( HRESULT hr, DWORD dwType );

    // Internal functions to manage and render the 3D scene
    static bool ConfirmDeviceHelper( D3DCAPS9* pCaps,
        VertexProcessingType vertexProcessingType, D3DFORMAT backBufferFormat );
    void    BuildPresentParamsFromSettings();
    bool    FindBestWindowedMode( bool bRequireHAL, bool bRequireREF );
    bool    FindBestFullscreenMode( bool bRequireHAL, bool bRequireREF );
```

```
    HRESULT ChooseInitialD3DSettings();
    HRESULT Initialize3DEnvironment();
    HRESULT HandlePossibleSizeChange();
    HRESULT Reset3DEnvironment();
    HRESULT ToggleFullscreen();
    HRESULT ForceWindowed();
    HRESULT UserSelectNewDevice();
    void    Cleanup3DEnvironment();
    HRESULT Render3DEnvironment();
    virtual HRESULT AdjustWindowForChange();
    virtual void UpdateStats();

protected:
    // Main objects used for creating and rendering the 3D scene
    D3DPRESENT_PARAMETERS m_d3dpp;       // Parameters for CreateDevice/Reset
    HWND            m_hWnd;          // The main app window
    HWND            m_hWndFocus;     // The D3D focus window (usually same as m_hWnd)
    HMENU           m_hMenu;         // App menu bar (stored here when fullscreen)
    LPDIRECT3D9     m_pD3D;          // The main D3D object
    LPDIRECT3DDEVICE9 m_pd3dDevice;  // The D3D rendering device
    D3DCAPS9        m_d3dCaps;       // Caps for the device
    D3DSURFACE_DESC m_d3dsdBackBuffer; // Surface desc of the backbuffer
    DWORD           m_dwCreateFlags; // Indicate sw or hw vertex processing
    DWORD           m_dwWindowStyle; // Saved window style for mode switches
    RECT            m_rcWindowBounds; // Saved window bounds for mode switches
    RECT            m_rcWindowClient; // Saved client area size for mode switches

    // Variables for timing
    FLOAT           m_fTime;         // Current time in seconds
    FLOAT           m_fElapsedTime;  // Time elapsed since last frame
    FLOAT           m_fFPS;          // Instanteous frame rate
    TCHAR           m_strDeviceStats[90];// String to hold D3D device stats
    TCHAR           m_strFrameStats[90]; // String to hold frame stats
```

```
    // Overridable variables for the app
    TCHAR*          m_strWindowTitle;   // Title for the app's window
    DWORD           m_dwCreationWidth;  // Width used to create window
    DWORD           m_dwCreationHeight; // Height used to create window
    bool        m_bShowCursorWhenFullscreen; // Whether to show cursor when fullscreen
    bool        m_bClipCursorWhenFullscreen; // Whether to limit cursor pos when fullscreen
    bool        m_bStartFullscreen;  // Whether to start up the app in fullscreen mode

    // Overridable functions for the 3D scene created by the app
    virtual HRESULT ConfirmDevice(D3DCAPS9*,DWORD,D3DFORMAT)   { return S_OK; }
    virtual HRESULT OneTimeSceneInit()              { return S_OK; }
    virtual HRESULT InitDeviceObjects()             { return S_OK; }
    virtual HRESULT RestoreDeviceObjects()          { return S_OK; }
    virtual HRESULT FrameMove()                     { return S_OK; }
    virtual HRESULT Render()                        { return S_OK; }
    virtual HRESULT InvalidateDeviceObjects()       { return S_OK; }
    virtual HRESULT DeleteDeviceObjects()           { return S_OK; }
    virtual HRESULT FinalCleanup()                  { return S_OK; }

public:
    // Functions to create, run, pause, and clean up the application
    virtual HRESULT Create( HINSTANCE hInstance );
    virtual INT     Run();
    virtual LRESULT MsgProc( HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam );
    virtual void    Pause( bool bPause );
    virtual         ~CD3DApplication()              { }

    // Internal constructor
    CD3DApplication();
};
```

2 รายละเอียดของคลาส CMyD3DApplication

```
class CMyD3DApplication : public CD3DApplication
{
    BOOL            m_bLoadingApp;      // TRUE, if the app is loading
```

```
CD3DFont*           m_pFont;            // Font for drawing text

CInputDeviceManager*  m_pInputDeviceManager;  // DirectInput device manager
DIACTIONFORMAT        m_diafGame;         // Action format for game play
LPDIRECT3DSURFACE9    m_pDIConfigSurface;   // Surface for config'ing DInput devices
UserInput             m_UserInput;        // Struct for storing user input

FLOAT               m_fSoundPlayRepeatCountdown; // Sound repeat timer
CMusicManager*      m_pMusicManager;      // DirectMusic manager class
CMusicSegment*      m_pBounceSound;       // Bounce sound

FLOAT               m_fWorldRotX;         // World rotation state X-axis
FLOAT               m_fWorldRotY;         // World rotation state Y-axis

    bool                                  m_bClearSW;
    bool                                  m_bClearSW2;


    //////////////////////////
    // Additional variables //
    //////////////////////////

    // Object from class
    CCar*                                 m_pCar;                    //
Represent car parameters
    CCamera*                              m_pCamera;

    // Vertex buffer
    IDirect3DVertexBuffer9* m_pTerrainVB;
    IDirect3DVertexBuffer9* m_pSkyboxVB;
    IDirect3DVertexBuffer9* m_pMirrorVB;
    IDirect3DVertexBuffer9*  m_pConsoleVB;

    // Xfiles
```

```
    ID3DXMesh*              m_pWall[25];        // The Wall
    CD3DMesh*               m_pCarbody;         // The
Car body
    CD3DMesh*               m_pBridge;          // The
bridge
    CD3DMesh*               m_pHouse1;          // The 1st
house
    CD3DMesh*               m_pGrandStand;
    CD3DMesh*               m_pWhiteBlock;
    CD3DMesh*               m_pRedBlock;
    CD3DMesh*               m_pPalace;
    CD3DMesh*               m_pConcreteBlock;
    CD3DMesh*               m_pWireFence;
    CD3DMesh*               m_pFence1;
    CD3DMesh*               m_pGate;
    CD3DMesh*               m_pWall1;

    // Bounding box
    BoundingBox             m_sBoundingWall[25];    // The Wall
    BoundingBox             m_sBoundingBridge;      // The Bridge

    // Texture
    IDirect3DTexture9*  m_pTopSkyboxTexture;
    IDirect3DTexture9*  m_pFrontSkyboxTexture;
    IDirect3DTexture9*  m_pLeftSkyboxTexture;
    IDirect3DTexture9*  m_pRightSkyboxTexture;
    IDirect3DTexture9*  m_pBackSkyboxTexture;
    IDirect3DTexture9*  m_pConsoleTexture;
    IDirect3DTexture9*  m_pGrassTexture;
    IDirect3DTexture9*  m_pShortRoadTexture;
    IDirect3DTexture9*  m_pCornerRoadTexture;
    IDirect3DTexture9*  m_pRoadTexture;
    IDirect3DTexture9*  m_pLeftRightTexture;
```

```
    // Others
    DWORD                   m_dwAppState;       // Current
state the app is in
    D3DXVECTOR3             m_vPos;             // Current
position
  TCHAR m_strProfilePath[MAX_PATH];             // Use for set current
directory
  TCHAR m_strCurrentWorkingDir[MAX_PATH];
    FLOAT                   m_fRotationY;       // Current
direction
    FLOAT                   m_fBridgeAngle;
    INT                     m_iTimeDelay;
    BOOL                    m_bPass;

    BOOL                    EngineCheck;        // For Checking
ON/OFF Engine completely
    INT                     OldGear;
    // For Checking Clutch is press before gear is changed
    FLOAT                   v;
    // real velocity to move car
    INT                     Accel;
    INT                     Brake;
    INT                     ClutchCounter;
    INT                     GearTemp;
    INT                     GearTemp2;
    INT                     Old1;
    INT                     Old2;
    INT                     Old3;
    BOOL                Flag;
    INT                     RPM;
    INT                     Direction;

protected:
  virtual HRESULT OneTimeSceneInit();
```

```
virtual HRESULT InitDeviceObjects();
virtual HRESULT RestoreDeviceObjects();
virtual HRESULT InvalidateDeviceObjects();
virtual HRESULT DeleteDeviceObjects();
virtual HRESULT Render();
virtual HRESULT FrameMove();
virtual HRESULT FinalCleanup();
virtual HRESULT ConfirmDevice( D3DCAPS9*, DWORD, D3DFORMAT );
VOID    Pause( bool bPause );

HRESULT RenderText();

HRESULT InitInput( HWND hWnd );
void    UpdateInput( UserInput* pUserInput );
void    CleanupDirectInput();

HRESULT InitAudio( HWND hWnd );

VOID    ReadSettings();
VOID    WriteSettings();

    //////////////////////////
    // Additional functions //
    //////////////////////////

    HRESULT RenderFrame();
    HRESULT RenderBackMirror();
    HRESULT RenderLeftMirror();
    HRESULT RenderRightMirror();
    VOID UpdateCarParam();
    bool ComputeBoundingBox(ID3DXMesh* mesh, BoundingBox* box);
     bool ComputeBoundingSphere(ID3DXMesh* mesh, BoundingSphere* sphere);
    bool isHit();
    bool OnBridge();
```

```
        void MovementOnBridge();
        void EngineOFF();
        bool CalAccel(int parameter);


public:
    LRESULT MsgProc( HWND hWnd, UINT msg, WPARAM wParam, LPARAM lParam );
    CMyD3DApplication();
    virtual ~CMyD3DApplication();

    HRESULT InputAddDeviceCB( CInputDeviceManager::DeviceInfo* pDeviceInfo, const
DIDEVICEINSTANCE* pdidi );
    static HRESULT CALLBACK StaticInputAddDeviceCB( CInputDeviceManager::DeviceInfo*
pDeviceInfo, const DIDEVICEINSTANCE* pdidi, LPVOID pParam );
    BOOL    ConfigureInputDevicesCB( IUnknown* pUnknown );
    static BOOL CALLBACK StaticConfigureInputDevicesCB( IUnknown* pUnknown, VOID*
pUserData );
};
```

3 รายละเอียดของคลาส CD3DFont

```
class CD3DFont
{
    TCHAR   m_strFontName[80];          // Font properties
    DWORD   m_dwFontHeight;
    DWORD   m_dwFontFlags;

    LPDIRECT3DDEVICE9       m_pd3dDevice; // A D3DDevice used for rendering
    LPDIRECT3DTEXTURE9      m_pTexture;  // The d3d texture for this font
    LPDIRECT3DVERTEXBUFFER9 m_pVB;       // VertexBuffer for rendering text
    DWORD   m_dwTexWidth;               // Texture dimensions
    DWORD   m_dwTexHeight;
    FLOAT   m_fTextScale;
    FLOAT   m_fTexCoords[128-32][4];
    DWORD   m_dwSpacing;                // Character pixel spacing per side
```

```
    // Stateblocks for setting and restoring render states
    LPDIRECT3DSTATEBLOCK9 m_pStateBlockSaved;
    LPDIRECT3DSTATEBLOCK9 m_pStateBlockDrawText;

public:
    // 2D and 3D text drawing functions
    HRESULT DrawText( FLOAT x, FLOAT y, DWORD dwColor,
                      const TCHAR* strText, DWORD dwFlags=0L );
    HRESULT DrawTextScaled( FLOAT x, FLOAT y, FLOAT z,
                            FLOAT fXScale, FLOAT fYScale, DWORD dwColor,
                            const TCHAR* strText, DWORD dwFlags=0L );
    HRESULT Render3DText( const TCHAR* strText, DWORD dwFlags=0L );

    // Function to get extent of text
    HRESULT GetTextExtent( const TCHAR* strText, SIZE* pSize );

    // Initializing and destroying device-dependent objects
    HRESULT InitDeviceObjects( LPDIRECT3DDEVICE9 pd3dDevice );
    HRESULT RestoreDeviceObjects();
    HRESULT InvalidateDeviceObjects();
    HRESULT DeleteDeviceObjects();

    // Constructor / destructor
    CD3DFont( const TCHAR* strFontName, DWORD dwHeight, DWORD dwFlags=0L );
    ~CD3DFont();
};
```

4 รายละเอียดของคลาส CInputDeviceManager

```
class CInputDeviceManager
{
public:
```

```
    struct DeviceInfo
    {
        LPDIRECTINPUTDEVICE8 pdidDevice;
        LPVOID           pParam;
    };

    typedef HRESULT (CALLBACK
*LPDIMANAGERCALLBACK)(CInputDeviceManager::DeviceInfo* pDeviceInfo, const
DIDEVICEINSTANCE* pdidi, LPVOID);

private:
    BOOL            m_bCleanupCOM;
    HWND            m_hWnd;
    TCHAR*          m_strUserName;

    LPDIRECTINPUT8      m_pDI;
    DeviceInfo*         m_pDevices;
    DWORD               m_dwMaxDevices;
    DWORD               m_dwNumDevices;
    DIACTIONFORMAT      m_diaf;

    LPDIMANAGERCALLBACK  m_AddDeviceCallback;
    LPVOID          m_AddDeviceCallbackParam;

public:
    // Device control
    HRESULT AddDevice( const DIDEVICEINSTANCE* pdidi, LPDIRECTINPUTDEVICE8
pdidDevice );
    HRESULT GetDevices( DeviceInfo** ppDeviceInfo, DWORD* pdwNumDevices );
    HRESULT ConfigureDevices( HWND hWnd, IUnknown* pSurface, VOID* pCallback, DWORD
dwFlags, LPVOID pvCBParam );
    VOID UnacquireDevices();
    VOID SetFocus( HWND hWnd );
```

```
    // Construction
    HRESULT SetActionFormat( DIACTIONFORMAT& diaf, BOOL bReenumerate );
    HRESULT Create( HWND hWnd, TCHAR* strUserName, DIACTIONFORMAT& diaf,
LPDIMANAGERCALLBACK AddDeviceCallback, LPVOID pCallbackParam );

    CInputDeviceManager();
    ~CInputDeviceManager();
};
```

5 รายละเอียดของคลาส CMusicManager

```
class CMusicManager
{
protected:
    BOOL                m_bCleanupCOM;
    IDirectMusicLoader8*      m_pLoader;
    IDirectMusicPerformance8* m_pPerformance;
    IDirectSound3DListener*   m_pDSListener;
    DS3DLISTENER        m_dsListenerParams;          // Listener properties

public:
    CMusicManager();
    ~CMusicManager();

    inline IDirectMusicLoader8*      GetLoader()     { return m_pLoader; }
    inline IDirectMusicPerformance8* GetPerformance() { return m_pPerformance; }
    inline IDirectSound3DListener*   GetListener()    { return m_pDSListener; }

    IDirectMusicAudioPath8* GetDefaultAudioPath();

    HRESULT Initialize( HWND hWnd, DWORD dwPChannels = 128, DWORD dwDefaultPathType =
DMUS_APATH_DYNAMIC_STEREO, LPDIRECTSOUND pDS = NULL );

    HRESULT SetSearchDirectory( const TCHAR* strMediaPath );
    VOID    CollectGarbage();
```

```
    VOID    StopAll();

    HRESULT CreateSegmentFromFile( CMusicSegment** ppSegment, TCHAR* strFileName,
                                   BOOL bDownloadNow = TRUE, BOOL bIsMidiFile = FALSE );
    HRESULT Create3DSegmentFromFile( C3DMusicSegment** ppSegment, TCHAR* strFileName,
                                   BOOL bDownloadNow = TRUE, BOOL bIsMidiFile = FALSE,
                        IDirectMusicAudioPath8* p3DAudioPath = NULL );
    HRESULT CreateScriptFromFile( CMusicScript** ppScript, TCHAR* strFileName );

    HRESULT CreateChordMapFromFile( IDirectMusicChordMap8** ppChordMap, TCHAR*
strFileName );
    HRESULT CreateStyleFromFile( IDirectMusicStyle8** ppStyle, TCHAR* strFileName );
    HRESULT GetMotifFromStyle( IDirectMusicSegment8** ppMotif, TCHAR* strStyle, TCHAR*
wstrMotif );

    HRESULT CreateSegmentFromResource( CMusicSegment** ppSegment, TCHAR* strResource,
TCHAR* strResourceType,
                                   BOOL bDownloadNow = TRUE, BOOL bIsMidiFile = FALSE );

    VOID Set3DParameters( FLOAT fDistanceFactor, FLOAT fDopplerFactor, FLOAT fRolloffFactor );
};
```

6 รายละเอียดของคลาส CMusicSegment

```
class CMusicSegment
{
protected:
    IDirectMusicSegment8*    m_pSegment;
    IDirectMusicLoader8*     m_pLoader;
    IDirectMusicPerformance8* m_pPerformance;
    IDirectMusicAudioPath8*   m_pEmbeddedAudioPath;
    BOOL                     m_bDownloaded;

public:
    CMusicSegment( IDirectMusicPerformance8* pPerformance,
```

```
            IDirectMusicLoader8* pLoader,
            IDirectMusicSegment8* pSegment );
    virtual ~CMusicSegment();

    inline  IDirectMusicSegment8* GetSegment() { return m_pSegment; }
    HRESULT GetStyle( IDirectMusicStyle8** ppStyle, DWORD dwStyleIndex = 0 );

    HRESULT SetRepeats( DWORD dwRepeats );
    virtual HRESULT Play( DWORD dwFlags = DMUS_SEGF_SECONDARY,
IDirectMusicAudioPath8* pAudioPath = NULL );
    HRESULT Stop( DWORD dwFlags = 0 );
    HRESULT Download( IDirectMusicAudioPath8* pAudioPath = NULL );
    HRESULT Unload( IDirectMusicAudioPath8* pAudioPath = NULL );

    BOOL    IsPlaying();
};
```

7 รายละเอียดของคลาส CCar

```
class CCar
{
public:
        CCar(){ m_fWheelDirection = 0; m_fVelocity = 0.00f; m_iAccel = 0; m_iGear = 0; EngineState =
false; m_iRPM = 0; m_fDirection = 0;};
public:
        D3DXVECTOR3 m_vFrontLeft;
        D3DXVECTOR3 m_vFrontRight;
        D3DXVECTOR3 m_vBackLeft;
        D3DXVECTOR3 m_vBackRight;
        D3DXVECTOR3 m_vCurrentPos;
        FLOAT      m_fWheelDirection;
        FLOAT           m_fVelocity;
        int                     m_iAccel;
        int                     m_iGear;
        int                     m_iRPM;
```

```
        BOOL            m_fDirection;
    CD3DMesh*   m_pCar;
        BOOL    EngineState;
};
```

8 รายละเอียดของคลาส CCamera

```
class CCamera
{
public:

        CCamera();
        ~CCamera();

        void strafe(float units); // left/right
        void fly(float units);    // up/down
        void walk(float units);   // forward/backward

        void pitch(float angle); // rotate on right vector
        void yaw(float angle);   // rotate on up vector
        void roll(float angle);  // rotate on look vector

        void getViewMatrix(D3DXMATRIX* V);
        void getPosition(D3DXVECTOR3* pos);
        void setPosition(D3DXVECTOR3* pos);

        void getRight(D3DXVECTOR3* right);
        void getUp(D3DXVECTOR3* up);
        void getLook(D3DXVECTOR3* look);
public:
        D3DXVECTOR3 m_vRight;
        D3DXVECTOR3 m_vUp;
        D3DXVECTOR3 m_vLook;
        D3DXVECTOR3 m_vPos;
};
```

9 รายละเอียดของคลาส CD3DMesh

```
class CD3DMesh
{
public:
    TCHAR           m_strName[512];

    LPD3DXMESH      m_pSysMemMesh;   // SysMem mesh, lives through resize
    LPD3DXMESH      m_pLocalMesh;    // Local mesh, rebuilt on resize

    DWORD           m_dwNumMaterials; // Materials for the mesh
    D3DMATERIAL9*   m_pMaterials;
    LPDIRECT3DTEXTURE9* m_pTextures;
    bool            m_bUseMaterials;

public:
    // Rendering
    HRESULT Render( LPDIRECT3DDEVICE9 pd3dDevice,
                    bool bDrawOpaqueSubsets = true,
                    bool bDrawAlphaSubsets = true );

    // Mesh access
    LPD3DXMESH GetSysMemMesh() { return m_pSysMemMesh; }
    LPD3DXMESH GetLocalMesh()  { return m_pLocalMesh; }

    // Rendering options
    void    UseMeshMaterials( bool bFlag ) { m_bUseMaterials = bFlag; }
    HRESULT SetFVF( LPDIRECT3DDEVICE9 pd3dDevice, DWORD dwFVF );

    // Initializing
    HRESULT RestoreDeviceObjects( LPDIRECT3DDEVICE9 pd3dDevice );
    HRESULT InvalidateDeviceObjects();

    // Creation/destruction
```

```
    HRESULT Create( LPDIRECT3DDEVICE9 pd3dDevice, TCHAR* strFilename );
    HRESULT Create( LPDIRECT3DDEVICE9 pd3dDevice, LPDIRECTXFILEDATA pFileData );
    HRESULT Destroy();

    CD3DMesh( TCHAR* strName = _T("CD3DFile_Mesh") );
    virtual ~CD3DMesh();
};
```