การจำลองการขับขี่ยานพาหนะแบบเสมือนจริง

THE SIMULATION OF DRIVING IN VIRTUAL REALITY

นายกิตติคุณ เชื้อหาญ
นายจักรพงศ์ ปิ่นทวีเกียรติ
นายนันทวัจน์ เหลืองอรุณ

ปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต
ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2544

THE SIMULATION OF DRIVING IN VIRTUAL REALITY

KITTIKUN CHUEHARN
JAKKAPHONG PINTAVEGRAED
NUNTAWAT LUANGAROON

A SPECAIL PROJECT SUBMITTED IN PARTAIL FULFILLMENT
OF THE REQUIREMENT FOR THE DEGREE OF BACHELOR OF SCIENCE
DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE
FACULTY OF SCIENCE
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
ACADEMIC YEAR 2001

หัวข้อปัญหาพิเศษ	การจำลองการขับขี่ยานพาหนะแบบเสมือนจริง	
	THE SIMULATION OF DRIVING IN VIRTUAL REALITY	
ชื่อนักศึกษา	นาย กิตติคุณ เชื้อหาญ	41056003
	นาย จักรพงศ์ ปิ่นทวีเกียรติ	41056009
	นาย นันทวัจน์ เหลืองอรุณ	41056052
ภาควิชา	คณิตศาสตร์และวิทยาการคอมพิวเตอร์	
สาขาวิชา	วิทยาการคอมพิวเตอร์	
อาจารย์ที่ปรึกษา	อาจารย์ชาญชัย ดีอ่วม	

ภาควิชาคณิตศาสตร์ประยุกต์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้นับปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์ ประจำปีการศึกษา 2544

คณะกรรมการสอบ	ลายมือชื่อ
ประธานกรรมการ อาจารย์วีระชัย ตันยะสิทธิ์	
กรรมการ ผู้ช่วยศาสตราจารย์ธีรวัฒน์ ประกอบผล	
กรรมการและอาจารย์ที่ปรึกษา อาจารย์ชาญชัย ดีอ่วม	

(ผู้ช่วยศาสตราจารย์ไพรบูลย์ พันธรักษ์พงษ์)
หัวหน้าภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์

ลิขสิทธิ์ของภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

หัวข้อปัญหาพิเศษ	การจำลองการขับขี่ยานพาหนะแบบเสมือนจริง		
ชื่อนักศึกษา	นาย กิตติคุณ เชื้อหาญ	รหัสนักศึกษา	41056003
	นาย จักรพงศ์ ปิ่นทวีเกียรติ	รหัสนักศึกษา	41056009
	นาย นันทวัจน์ เหลืองอรุณ	รหัสนักศึกษา	41056052
ปริญญา	วิทยาศาสตรบัณฑิต		
ภาควิชา	คณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์		
สาขาวิชา	วิทยาการคอมพิวเตอร์		
ปีการศึกษา	2544		
อาจารย์ที่ปรึกษา	อาจารย์ชาญชัย ดีอ่วม		

บทคัดย่อ

ในปัจจุบันอุตสาหกรรมเกมคอมพิวเตอร์ได้มีการพัฒนาไปอย่างรวดเร็วมาก เกมต่างๆที่ออกมาสู่ท้องตลาด เริ่มมีภาพและเสียงที่สมจริงมากขึ้น ทำให้ผู้เล่นได้รู้สึกเหมือนได้อยู่ในสถานการณ์จริง เช่น เกมขับรถ ขับเครื่องบิน เพราะนอกจากภาพที่เป็นแบบสามมิติแล้ว การควบคุมก็เป็นไปอย่างอิสระมากขึ้น นอกจากนี้ยังมีระบบเสียงเข้ามาช่วยอีก สิ่งที่ผู้เล่นได้รับ นอกจากความสนุกสนานแล้ว ยังมีการฝึกไหวพริบ การตัดสินใจในเหตุการณ์เฉพาะหน้าอีกด้วย แต่น่าเสียดายที่โปรแกรมเกมคอมพิวเตอร์ส่วนใหญ่ยังคงเป็นของต่างประเทศ การผลิตเกมในไทยยังไม่ค่อยมีให้เห็นมากนัก

คณะผู้ศึกษาจึงได้เลือกศึกษาเกี่ยวกับการสร้างโปรแกรมเกมคอมพิวเตอร์ และได้เลือกสร้างเกมประเภทแข่งรถ แต่จะมีการนำเสนอที่แปลกไปจากเกมแข่งรถทั่วๆไป จะไม่แข่งในสนามแข่ง แต่จะแข่งบนถนนทั่วไปซึ่งมีรถคันอื่นๆสัญจรไปมาตามปกติ นอกจากได้ความรู้สึกที่แปลกใหม่แล้ว ยังได้ความรู้สึกเหมือนอยู่ในสถานการณ์จริงอีกด้วย เพราะเป็นถนนทั่วๆไปที่ใกล้เคียงกับที่เราคุ้นเคยในชีวิตประจำวัน และเป็นบรรยากาศ ทิวทัศน์ข้างทาง แบบที่เราคุ้นเคย นอกจากจะได้ความรู้สึกเหมือนเหตุการณ์จริงแล้ว ยังรู้สึกว่าเป็นเกมของคนไทย ที่ผลิตโดยคนไทยอีกด้วย

Special Project Title	THE SIMULATION OF DRIVING IN VIRTUAL REALITY	
Students	Mr. Kittikun Chuehan	41056003
	Mr. Jakkaphong Pintavegread	41056009
	Mr. Nuntawat Luangaroon	41056052
Degree	Bachelor's Degree of Science	
Department	Mathematics and Computer Sciences, Faculty of Science	
Programme	Computer Sciences	
Academic Year	2001	
Special Project Advisor	Lecturer Chanchai Dee-Uam	

ABSTRACT

The computer game industries are improving and developing very fast. The games that are being published have sounds and pictures that are more realistic then they have ever been, which make the players feel like their experiencing the real thing. Like in Car Simulation games. Not only has the picture improved, but also the controls give more freedom in controlling the game and the sound helps the player by letting the player use his sense of sound to help. The thing that the players will receive is not only the entertainment that game give, but they will be able to have a chance to improve they skill in making decisions and thing in situations that they may or may not find in the real world. But the down side of this story is that all of the computer games, which are in this field, were all developed by computer programmers from other countries. Hardly any were developed from Thai computed programmers.

The staff started by studying about developing computer games and then choices to build a Car Simulation game, but with something a little bit more special. The environment of the game won't be in a racing track but in the streets of the city instead, which will have surrounding by traffic. Not only will the players be impressed with the game, but they also will feel like having the experience of being in the real situation and see the view of the environment that is almost just like what the see every day. And all over that, they will be specially impressed that they are playing a game that was developed by Thai programmers.

สารบัญ

หน้า

กิตติกรรมประกาศ

ขอกราบขอบพระคุณบิดา มารดา ที่ได้ให้ความสนับสนุนทางด้านกำลังใจและทุนทรัพย์ จนการทำปัญหาพิเศษครั้งนี้สำเร็จด้วยดี

ขอขอบพระคุณ อาจารย์ชาญชัย ดีอ่วม อาจารย์ผู้รับผิดชอบปัญหาพิเศษฉบับนี้ที่กรุณาให้คำแนะนำและเป็นที่ปรึกษาในการแก้ปัญหาต่างๆ รวมทั้งเป็นผู้ตรวจสอบความถูกต้องของปัญหาพิเศษฉบับนี้

ขอขอบพระคุณอาจารย์ ธีรวัฒน์ ประกอบผล ที่กรุณาแนะนำเพิ่มเติมเกี่ยวกับปัญหาต่างๆ และมอบความเป็นกันเอง เอาใจใส่ ทำให้คณะผู้จัดทำมีกำลังใจและไม่ท้อแท้

ขอขอบพระคุณอาจารย์วีระชัย ตันยะสิทธิ์ สำหรับข้อคิดเห็นและข้อแนะนำที่ดี

ขอบคุณเพื่อนๆทุกคนที่คอยให้กำลังใจ และคอยช่วยเหลือในด้านต่างๆ

คณะผู้จัดทำ

มีนาคม 2545

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญรูป

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

สารบัญตาราง

ตารางที่ **หน้า**

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในปัจจุบันอุตสาหกรรมเกมคอมพิวเตอร์ได้มีการพัฒนาและเปลี่ยนแปลงไปอย่างรวดเร็ว เทคโนโลยีที่พัฒนาอยู่ทุกวัน ส่งผลให้นักพัฒนาสามารถสร้างเกมที่มีระบบภาพและเสียงได้ดีขึ้นเรื่อยๆ มีความเป็นสามมิติ ทำให้โลกของเกม ใกล้เคียงกับโลกแห่งความเป็นจริงมากขึ้น ผู้เล่นได้รู้สึกเหมือนอยู่ในสถานการณ์จริง ในฐานะนักพัฒนา เทคนิคและวิธีการสร้างระบบเกมนับว่าเป็นสิ่งที่น่าสนใจ น่าศึกษามาก เพราะระบบเกมต้องมีการพัฒนาในหลายๆด้าน ตั้งแต่การออกแบบระบบเกม การเขียนโปรแกรมติดต่อกับฮาร์ดแวร์ การแสดงผลแบบสามมิติ การใช้โปรแกรมอื่นๆในการสร้างภาพและเสียง สร้างภาพเคลื่อนไหว การเขียนโปรแกรมเพื่อโต้ตอบกับผู้เล่นในหลายๆลักษณะ และผลงานที่ได้ยังเป็นประโยชน์ต่อผู้ที่สนใจการพัฒนาเกม ในการนำไปศึกษาต่อและเกิดแรงบันดาลใจให้ผู้พัฒนาท่านอื่นที่ได้พบเห็น ก่อให้เกิดความคึกคักขึ้นในอุตสาหกรรมเกมคอมพิวเตอร์ในประเทศไทย

1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษา

1.2.1 เพื่อพัฒนาและศึกษาการเขียนโปรแกรมเกม 3 มิติแบบ Simulation

1.2.2 เพื่อศึกษาการเขียนโปรแกรมแสดงผล และควบคุมในแบบสามมิติ

1.3 สมมุติฐานของการศึกษา

1.3.1 ได้ศึกษาและเรียนรู้ขั้นตอนและวิธีการสร้างเกมจากการทำงานจริง

1.3.2 เกมสามารถแสดงผลได้ในรูปแบบสามมิติ โดยปรับเปลี่ยนมุมมองได้แม้ในระหว่างเล่น

1.3.3 สามารถเขียนโปรแกรมควบคุมการแสดงภาพแบบสามมิติ และตอบโต้กับผู้ใช้ได้

1.4 ทฤษฎีหรือแนวความคิดที่ใช้ในการศึกษา

ออกแบบและสร้างโมเดลต่างๆที่ใช้ในเกม ในรูปแบบของโมเดลสามมิติ โดยโปรแกรมที่ใช้ในการสร้างงานสามมิติ แล้วนำเข้ามาใช้ในเกม โดยเขียนโปรแกรมควบคุมการแสดงผลผ่าน Microsoft DirectX ให้ช่วยในการประมวลผลภาพ และการเคลื่อนไหว ออกมาเป็นแบบสามมิติ

1.5 ขอบเขตการศึกษา

1.5.1 ตัวโปรแกรมจะเป็นเกมแนว Simulation ประเภท racing

1.5.2 ฉากที่ใช้จะเป็นฉาก 3 มิติ และเป็นสภาพแวดล้อมภายในเมืองไทยที่คุ้นตา

1.5.3 ผู้เล่นสามารถเลือกรถ , บังคับรถได้อย่างอิสระ และสามารถเลือกมุมมองของรถที่ตัวเองถนัดได้

1.5.4 จะมีเวลาแสดงให้รู้ว่าใช้เวลาตั้งแต่เริ่มสตาร์ทจนถึงเส้นชัยใช้เวลาทั้งหมดเท่าไร

1.5.5 เมื่อผู้เล่นบังคับรถภายในฉากผ่านจุดที่ไม่สามารถจะผ่านไปได้ก็จะทำให้เกิดการชนขึ้น

1.6 ขั้นตอนการศึกษา

1.6.1 ศึกษาระบบและขั้นตอนการสร้างเกม

เริ่มศึกษาวิธีการ ขั้นตอนต่างๆ ในการสร้างเกม เทคโนโลยี เทคนิคต่างๆที่จะใช้พัฒนา

1.6.2 รวบรวมข้อมูลต่างๆ ที่เป็นประโยชน์ในการพัฒนา

รวบรวมเอาข้อมูลต่างๆ ที่เกี่ยวข้อง มีประโยชน์กับการพัฒนา ทั้งจากหนังสือ ตำราต่างๆ และเว็ปไซด์ หรือจากการพูดคุยสอบถามจากผู้รู้ หรือมีประสบการณ์

1.6.3 ศึกษาโปรแกรมที่ต้องการใช้

ศึกษาการใช้งานโปรแกรม เครื่องมือต่างๆ ที่จะนำมาใช้ในการพัฒนา เช่น Microsoft Visual C++, DirectX SDK และโปรแกรมทางด้านภาพและเสียง อาทิเช่นPhotoshop, 3DSMax, Cakewalk Pro Audio, Cool Edit

1.6.4 วิเคราะห์และออกแบบระบบเกม

วิเคราะห์ระบบงาน และแบ่งงานออกเป็นส่วนต่างๆ ลำดับการทำงาน จัดลำดับการทำงานความสำคัญของส่วนต่างๆ เพื่อให้การพัฒนาเป็นไปอย่างมีระบบ และผิดพลาดให้น้อยที่สุด

1.6.5 พัฒนาโปรแกรม

ดำเนินการเขียนโปรแกรมตามที่ได้ออกแบบไว้

1.6.6 ทดสอบและปรับปรุงโปรแกรม

ทดสอบโปรแกรมที่เขียน หาข้อผิดพลาด โดยอาจให้ผู้อื่นที่ไม่มีส่วนเกี่ยวข้องกับการพัฒนา มาทดสอบโปรแกรมและช่วยหาข้อผิดพลาด เพื่อนำมาปรับปรุงแก้ไขโปรแกรม

บทที่ 2
ทฤษฎีและวรรณกรรมที่เกี่ยวข้อง

2.1 โปรแกรม Microsoft Direct X

2.1.1 ความเป็นมา

ในสมัยก่อน การเขียนโปรแกรมให้ติดต่อกับอุปกรณ์ต่างๆ เป็นเรื่องยุ่งยาก โปรแกรมเมอร์จำเป็นที่จะต้องทำการศึกษาถึงคำสั่งต่างๆของฮาร์ดแวร์แต่ละรุ่น แต่ละยี่ห้อ ซึ่งมีจำนวนมากและเป็นเรื่องปริมาณที่ทำให้เสียเวลา และเมื่อฮาร์ดแวร์ตัวเก่าล้าสมัย และฮาร์ดแวร์ตัวใหม่ถูกผลิตออกมา โปรแกรมเมอร์ก็จำเป็นต้องทำการศึกษาถึงวิธีการใช้คำสั่งต่างๆของฮาร์ดแวร์ตัวใหม่ ซึ่งอาจจะแตกต่างจากตัวเก่าอย่างสิ้นเชิง หรือมีการเพิ่มเทคโนโลยีใหม่ๆเข้ามาในฮาร์ดแวร์ซึ่งเป็นเรื่องที่ทำความลำบากแก่ผู้พัฒนาเป็นอย่างมาก

ในวินโดว์ (ตั้งแต่ เวอร์ชัน 3.1 เป็นต้นมา) มีการใช้ไดรเวอร์ของฮาร์ดแวร์ เพื่อที่จะเข้าถึงอุปกรณ์ เป็นเหตุทำให้นักโปรแกรมเมอร์พบกับความลำบากในการเขียนโปรแกรมเพื่อติดต่อกับฮาร์ดแวร์ ในการพัฒนาโปรแกรมโปรแกรมเมอร์คาดหวังว่าในวินโดว์รุ่นใหม่ๆ จะมีชุดคำสั่งที่จะช่วยให้การเขียนโปรแกรมติดต่อกับอุปกรณ์ สามารถทำผ่านอินเตอร์เฟซตัวหนึ่งเพียงตัวเดียวได้ ซึ่งอินเตอร์เฟซตัวนี้จะทำการเรียกใช้ไดรเวอร์ต่างๆของอุปกรณ์นั้นๆเอง ซึ่งจะทำให้การเขียนโปรแกรมเป็นไปอย่างสะดวก เพราะเพียงทำการศึกษาถึงคำสั่งการเรียกใช้อินเตอร์เฟซก็เพียงพอที่จะติดต่อกับอุปกรณ์ต่างๆ รุ่นใดๆ ยี่ห้อใดๆ ได้เลยทันที

บริษัท Microsoft Corporation เล็งเห็นถึงความต้องการของโปรแกรมเมอร์ตรงจุดนี้ จึงได้ทำการผลิตชุดพัฒนา SDK สำหรับ Microsoft Windows ขึ้นมา โดยมีจุดประสงค์ที่จะช่วยให้การเข้าถึงฮาร์ดแวร์ต่างๆ เป็นเรื่องง่ายยิ่งขึ้นอีก ซึ่งไม่จำเป็นต้องไปทำการศึกษาการทำงานของ Hardware ตัวนั้นๆเหมือนสมัยก่อน โดยระยะแรกมีวัตถุประสงค์ เพื่อใช้พัฒนาเกมคอมพิวเตอร์โดยเฉพาะ จึงได้ตั้งชื่อว่า “ Game Kit ” ต่อมา ได้เพิ่มความสามารถที่ช่วยทางด้านอื่นๆด้วย เช่น มัลติมีเดีย ดังนั้นจึงได้ถูกพัฒนาและเรียกใหม่ว่า “ Microsoft Direct X ” และชุดพัฒนาจะมี SDK ต่อท้าย โดยปัจจุบันได้ออกมาถึงรุ่น 7.0 แล้ว

2.1.2 หลักการของ Direct X

สิ่งที่ Microsoft ได้คำนึงถึงเป็นสิ่งแรกของ Direct X ก็คือ จะต้องเป็นชั้นระหว่างแอปพลิเคชั่นกับฮาร์ดแวร์ ที่ต้องมีความสัมพันธ์กันมากเสมือนเป็นชั้นเดียวกัน เพราะต้องการให้แอปพลิเคชั่นสามารถทำงานกับฮาร์ดแวร์ได้อย่างรวดเร็ว ความจริงแล้วก่อนที่ Direct X จะถูกพัฒนาขึ้นมา ได้มีชุดคำสั่งของวินโดว์ซึ่งใช้ติดต่อกับการ์ดแสดงผลอยู่ก่อนแล้ว ซึ่งเรียกว่า GDI มีลักษณะการทำงานคล้าย Direct X ก็คือ สามารถติดต่อกับอุปกรณ์ ฮาร์ดแวร์ต่างๆได้โดยตรง โดยไม่ต้องทำการเรียกใช้

ไดรเวอร์ของการ์ดแสดงผลเอง แต่ความสามารถจะช้ากว่า และความสามารถบางอย่างยังไม่สามารถทดแทนได้ เช่น หากโปรแกรมเมอร์ทำการเรียกคำสั่งที่ติดต่อกับฮาร์ดแวร์แสดงผล ซึ่งมีผลทำให้การทำงานเร็วมากขึ้น เพราะทำงานบนฮาร์ดแวร์ หากว่าเครื่องของผู้ใช้แอปพลิเคชั่นนั้นไม่มีฮาร์ดแวร์ดังกล่าว ถ้าเป็น GDI จะไม่สามารถทำงาน หรือใช้งานแอปพลิเคชั่นนั้นได้เลย แต่หากโปรแกรมเมอร์เรียกใช้ฮาร์ดแวร์ผ่านทาง Direct X เมื่อ Direct X ทำการตรวจสอบหาฮาร์ดแวร์ดังกล่าวและหาไม่เจอ ก็จะทำการจำลองฮาร์ดแวร์ตัวนั้นเหมือนกับว่ามีฮาร์ดแวร์ตัวนั้นอยู่ และใช้ อัลกอลิทึมที่กำหนดไว้ทำการแสดงผล และทำการแสดงผลแทนฮาร์ดแวร์ ตัวนั้น ซึ่งผู้ใช้สามารถเรียกใช้ ซอร์ฟแวร์แทนฮาร์ดแวร์ และยังสามารถใช้งานได้เหมือนเดิม

Direct X คือชุดของ Dynamic Link Library (DLL) ซึ่งสามารถทำให้โปรแกรมเมอร์เข้าถึงฮาร์ดแวร์ได้อย่างรวดเร็วโดยไม่ผ่านทางฮาร์ดแวร์ตามหลักการของวินโดว์

Direct X ยังเป็นมาตรฐานของซอร์ฟแวร์และ ฮาร์ดแวร์ในปัจจุบัน โดยบริษัทซอร์ฟแวร์ส่วนมากที่มีอุปกรณ์ที่เกี่ยวข้องกับ Direct X จะเขียนโปรแกรมให้เรียกใช้ Direct X และรวมไปถึงบริษัทที่ผลิตฮาร์ดแวร์ ก็ยังสนับสนุน Direct X โดยจะทำการเรียกใช้คำสั่งของ Direct X เพราะฉะนั้นเราสามารถที่จะมั่นใจได้ว่าDirect X จะเป็นที่แพร่หลายในอนาคต

Microsoft ได้ทำให้ Direct X มีคุณสมบัติพิเศษอย่างมาก เช่น เกมที่เขียนโดยใช้ Direct X 1.0 จะสามารถรันบนคอมพิวเตอร์ที่ติดตั้ง Direct X 3.0 หรือ Direct X 6.0 และยิ่งไปกว่านั้น Microsoft รู้ว่าเทคโนโลยี Direct X จะต้องหลุดออกจากหลักการ หากว่าถูกเขียนโดยไม่มีการมองการณ์ไกลและการวางแผน ดังนั้นจึงจำเป็นที่จะต้องใช้การใช้การโปรแกรมลักษณะของ Object Oriented ซึ่งสามารถปรับปรุงให้ดีขึ้นได้ และสามารถทำงานได้กับหลายๆภาษา และเป็นลักษณะของ Black Box เทคนิคเช่นนี้ถูกเรียกว่า Component Object Model (COM)

2.1.3 ส่วนประกอบของ Direct X

2.1.3.1 ทำความเข้าใจกับ HAL (The Hardware Abstraction Layer)

HAL คือซอร์ฟแวร์ระดับล่างสุดใน Direct X ซึ่งบรรจุไดรเวอร์ของฮาร์ดแวร์จากผู้ผลิตฮาร์ดแวร์รายนั้นๆ เพื่อที่จะทำการควบคุมฮาร์ดแวร์ได้โดยตรง ซอร์ฟแวร์ในชั้นนี้ให้ประสิทธิภาพสูงสุด เพราะเป็นชั้นที่ทำการติดต่อกับฮาร์ดแวร์โดย

2.1.3.2 ทำความเข้าใจกับ HEL (The Hardware Emulation Layer)

HEL จะวางอยู่ชั้นก่อนหน้า HAL โดยปกติ Direct X ถูกออกแบบมาให้ใช้ฮาร์ดแวร์ให้ได้เต็มประสิทธิภาพและด้วยความเร็วสูง หากว่ามีฮาร์ดแวร์ชนิดนั้นๆอยู่ แต่ Direct X ก็ยังสามารถทำงานได้อยู่ หากว่า ฮาร์ดแวร์ชนิดนั้นไม่มีอยู่ตัวอย่างเช่น หากเราต้องการสร้างภาพๆหนึ่งขึ้นมาซึ่งจำเป็นต้องเรียกใช้ฟังก์ชันบนฮาร์ดแวร์ชนิดนั้นได้โดยตรง แต่หากว่าฮาร์ดแวร์ชนิดนั้นไม่มีอยู่ในเครื่อง HEL จะเข้ามาทำงานในส่วนนี้ โดยการจำลองตัวเองเป็นฮาร์ดแวร์ตัวนั้นๆ และทำงานเคียงคู่กับ

HAL ซึ่งแน่นอนว่าทำให้การทำงานช้าลง เพราะเป็นการจำลองฮาร์ดแวร์ แต่ก็ยังทำให้แอปพลิเคชั่นนั้นทำงานได้ตามปกติได้

2.1.3.3 Direct Draw คืออะไร

อาจจะเรียกได้ว่า ส่วนนี้เป็นส่วนที่สำคัญที่สุดของ Direct X เพราะ Direct Draw สามารถที่จะทำการเรียกใช้งาน Video Card ด้วยความเร็วสูง โดยที่ Direct Draw รู้จักการ Set Mode ของจอภาพทุกๆโหมดไม่ว่าจะเป็น High-Resolution หรือจะเป็น True Color Modes แล้วยังสนับสนุนเรื่องการจัดการสี การกำหนดขอบเขตจอภาพ และการทำอนิเมชั่น

2.1.3.4 Direct 3D คืออะไร

ส่วนประกอบของ Direct X ชนิดนี้เป็นส่วนประกอบส่วนเดียวที่สามารถติดต่อกับอุปกรณ์แสดงผล 3 มิติได้ ทำให้สามารถเขียนโปรแกรมประเภท 3 มิติได้ เช่น เกมขับเครื่องบิน Simulation ต่างๆ

2.1.3.5 Direct Sound คืออะไร

ก่อนที่ยังไม่มี Direct X การเขียนโปรแกรมติดต่อกับกับการ์ดเสียงเป็นเรื่องที่ยากมาก ซึ่งการเขียนโปรแกรมที่อาศัยการ์ดเสียง และสร้างเสียงที่มีคุณภาพ ในสมัยก่อนมักจะว่าจ้างให้บริษัทที่ทำงานด้านนี้โดยเฉพาะเป็นผู้ออกแบบและทำการ Coding เช่นบริษัทเกม อาจเป็นบริษัทหนึ่ง แล้วจ้างบริษัทที่ทำระบบเสียงให้ทำการเพิ่มเสียงเพลงเข้าไปในตัวเกม เป็นต้น แต่ปัจจุบันเมื่อ Direct Sound ถือกำเนิดขึ้น สถานการณ์เช่นนั้นจึงหมดลงตามยุค เพราะ Direct Sound ลดขั้นตอนการเข้าถึงซาวด์การ์ดและทำงานได้กับซาวด์การ์ดทุกชนิด

2.1.3.6 Direct Sound 3D ทำงานอย่างไร

Direct Sound 3D อยู่บนพื้นฐานของ Direct Sound และทำงานได้กับระบบเสียง 3 มิติ ซึ่งจำเป็นต้องมีการคำนวณทางคณิตศาสตร์ Direct 3D จะทำการคำนวณและพิกัดตำแหน่งของเสียงในระบบ 3 มิติ ซึ่งจะทำให้เสียงที่ออกมาฟังสมจริงมากขึ้น

2.1.3.7 Direct Input คืออะไร

เป็นความสามารถใหม่ที่เพิ่งจะถูกเพิ่มเข้ามาใน Direct X 3.0 ซึ่งจะทำให้การรับข้อมูลทำได้หลากหลายขึ้นและทำได้โดยง่าย โดยรับจากคีย์บอร์ด , เมาส์ และ จอยสติก ซึ่งในเวอร์ชัน 5.0 ยังสนับสนุน จอยสติก แบบ Force Feed Back (สั่นสะเทือนได้) ในสมัยก่อน จอยสติกมีปุ่มเพียงเล็กน้อย แต่ในปัจจุบันมีมากมายหลายปุ่ม และมีหลากหลายยี่ห้อ ทำให้การติดต่อกับจอยสติกในแต่ละปุ่มต้องทำการศึกษาถึงจอยสติกชนิดนั้นๆอย่างลึกซึ้ง

2.1.3.8 Direct Play

เป็นส่วนประกอบที่ใช้ติดต่อกับอุปกรณ์ทางด้านเน็ตเวิร์ค การอ้างพอร์ตต่างๆหรือการสร้างเกมที่สามารถเล่นผ่านเครือข่ายได้

2.2 การเขียนโปรแกรมบน Windows

การเขียนโปรแกรมวินโดว์ อาจจะเป็นเรื่องที่ง่ายหรือยาก ขึ้นอยู่กับขนาดของโปรแกรมที่ต้องการทำ การเขียน Word Processor อาจต้องการความรู้ทางด้านการเขียนโปรแกรมที่มากพอสมควร แต่สำหรับการเขียนเกมโดยใช้ Direct X ต้องการเพียงเล็กน้อยเท่านั้น ในส่วนนี้จะอธิบายถึงการเขียนโปรแกรมบนวินโดว์ และพื้นฐานที่ควรจะรู้สำหรับการเขียนโปรแกรมบนวินโดว์เพื่อนำไปใช้ในการเขียนเกม

วินโดว์เป็นระบบปฏิบัติการแบบ Event-Based โดยแมสเสทเป็นเสมือนสัญญาณที่จะบอกให้วินโดว์ต้องทำอะไร หรือเกิดอะไร

ตัวอย่าง Windows Event เมื่อผู้ใช้ต้องการเปลี่ยนขนาดของ Window Application , วินโดว์ก็จะทำการส่งแมสเสทไปยังแอปพลิเคชั่นตัวนั้นว่าจะเปลี่ยนขนาดอย่างไร

Keyboard Event เมื่อผู้ใช้กดแป้นพิมพ์บนคีย์บอร์ด , แมสเสทของตัวแป้นพิมพ์ที่กดก็จะถูกส่งไปยังแอปพลิเคชั่น เพื่อกระทำการประมวลผลข้อมูลนั้น และแสดงผลข้อมูลนั้นออกมา เช่น แสดงเมนูย่อยจากเมนูบาร์หลัก

Drawing Event ถ้าวินโดว์ตัวใหม่ถูกแสดงออกมาทับตัวเก่าที่แสดงอยู่ หน้าจอจะต้องทำการวาดใหม่อีกครั้ง โดย Repaint Message จะกระทำการในส่วนนี้ เพราะฉะนั้นอาจจะกล่าวได้ว่า ทุกสิ่งทุกอย่างที่เกิดขึ้นในวินโดว์ก็คือแมสเสทนั่นเอง

ส่วนประกอบโดยทั่วไปในการสร้าง Windows Application ที่ต้องมีได้แก่

2.2.1 ทำไมต้องมี Win32 Libraries

เมื่อเราสร้าง Windows Application ขึ้นมาคอมไพเลอร์จะเป็นตัวจัดการเกี่ยวกับส่วนของไลบรารีให้เองทั้งหมด ยกเว้นเสียแต่ไลบรารีของแอปพลิเคชั่นที่เราต้องการไม่ได้ถูกตั้งค่าไว้ตั้งแต่เริ่มต้น เช่น เมื่อต้องการสร้าง แอปพลิเคชั่นเกี่ยวกับมัลติมีเดีย ซึ่งต้องการ Winmm.Lib Libraries แต่ในไดเรกเทอร์รี่ของ LIB ในตัวของ คอมไพเลอร์ไม่มี หรือไม่ได้ถูกกำหนดไว้ ผู้ใช้ก็ต้องเพิ่ม หรือกำหนดเข้าไปเอง เมื่อสร้างแอปพลิเคชั่นที่ใช้ Direct X ผู้ใช้ก็จะต้องเพิ่มไลบารีย์ของ Direct X เข้าไปในแอปพลิเคชั่นนั้นด้วย

2.2.2 The Include Files มีประโยชน์อย่างไร

โปรแกรมเมอร์ต้อง Include Files ที่จำเป็นสำหรับแอปพลิเคชั่นนั้นไว้ในตัวโปรแกรมด้วยเสมอ และโดยทั่วไปใน Windows Application จะต้องเริ่มต้นด้วย

```
# define win32_LEAN_AND_MEAN
# include < windows.h >
# include < windowsx.h >
```

และอาจจะเพิ่ม Include Files ที่จำเป็นสำหรับแอปพลิเคชั่นเข้าไปอีกได้ เช่น Stdio.h หรือ Math.h เป็นต้น

2.2.3 WinMain คืออะไร

เป็นฟังก์ชันหลักที่ Windows Application ทุกแอปพลิเคชั่นจำเป็นต้องกำหนดไว้ คล้ายๆกับฟังก์ชัน Main( ) ในภาษา C และภาษา C++ การประกาศ Prototype ของ WinMain( ) จะเป็นดังนี้

```
int WINAPI WinMain ( HINSTANCE hinstance ,
                     HINSTANCE hprevinstance ,
                     LPSTR lpcmline ,
                     Int ncmdshow ) ;
```

2.2.4 Windows Class จำเป็นอย่างไร

โปรแกรมเมอร์ที่ต้องการสร้าง Windows Application จำเป็นต้องกำหนด Windows Class ขึ้นมาด้วย ซึ่ง Windows Class จะอธิบายถึงคุณลักษณะโดยทั่วไปของ Windows ทั้งหมด โดยหลังจากโปรแกรมเมอร์ทำการสร้าง Windows Class ขึ้นมา แล้วโปรแกรมเมอร์ก็จะสามารถสร้างวินโดว์ได้ตาม Windows Class ที่กำหนดขึ้นมาได้ตามต้องการ และต่อไปนี้คือโครงสร้างของ Windows Class

```
typedef struct_WNDCLASS
{
    UINT        style ;          // style flags
    WNDPROC     lpfnWndProc ;    // pointer to event handler
    Int         cbClsExtra ;     // extra byte
    Int         cbWndExtra ;     // extra byte
    HANDLE      hInstance ;      // handle of instance
    HICON       hIcon ;          // handle to Icon
    HCURSOR     hCursor ;        // handle to cursor
    HBRUSH      hbrBackground    // handle to background brush
    LPCTSTR     lpszMenuName     // name of menu
    LPCTSTR     lpszClassName    // name of class
} WNDCLASS ;
    WNDCLASS wndclass            // a declaration of a window clas
```

2.2.5 Registering the Windows class คืออะไร

หลังจากที่ทำการสร้าง Windows Class ขึ้นมาแล้ว ขั้นต่อไปจะทำการ Register Class การ Register จะทำให้วินโดว์รู้ว่ามี Windows Class ที่สร้างขึ้นมา และสามารถให้โปรแกรมเมอร์สร้างวินโดว์ที่มีลักษณะตาม Windows Class ได้โดยชื่อ ASCII ของคลาสเพียงอย่างเดียว การ Register Class กระทำได้โดย

```
if ( RegisterClass ( &wndclass ) == 0 )
```

```
{
        // error
} // end if
```

2.2.6 การสร้าง Windows

การจะสร้างวินโดว์ขึ้นมาต้องทำการเรียกฟังก์ชัน , CreateWindow ( ) ที่มีค่าพารามิเตอร์ 11 ค่า การประกาศโปรโตไทต์จะทำได้ดังนี้

```
HWND CreateWindow (
        LPCTSTR     IpClassName ,    // pointer to class name ( string )
        LPCTSTR     IpWindowName , // pointer to window title ( string )
       DWORD  dwStyle ,        // windows style flags
       Int         x ,             // ตำแหน่งแนวแกน x ของวินโดวส์
       Int         y ,             // ตำแหน่งแนวแกน y ของวินโดวส์
       Int         nWidth ,        // ความกว้างของวินโดวส์
       Int         nHeight ,       // ความสูงของวินโดวส์
       HWND        hWndParent ,  // handle to parent ( usually NULL )
        HMENU      hMenu ,   // handle to menu ( usually NULL )
        HANDLE     hInstance ,     // handle to application instance
        LPVOID     IpParam    // pointer to startup creation data ( NULL)
) ;
```

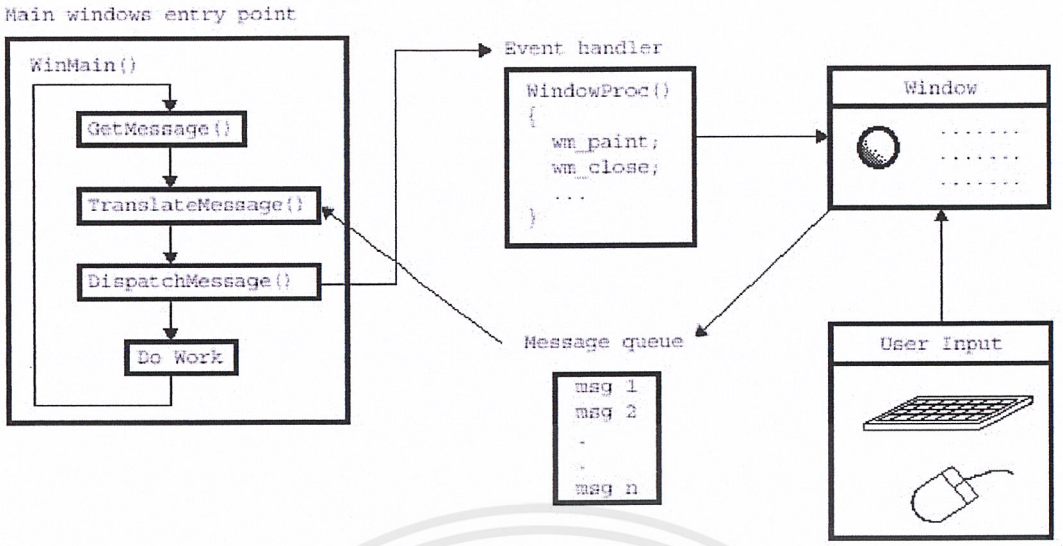
2.2.7 การแสดง Windows

หลังจากที่ทำการ Create Windows เสร็จแล้ว และต้องการแสดงผลออกมา ในกรณีที่ตั้งค่าพารามิเตอร์ของ dwStyle ไม่ใช่ WS_VISIBLE ต้องทำการเรียก Function ShowWindow ( ) เสมอ

```
BOOL ShowWindow ( HWND  hWnd ,     // handle of window
                  Int  nCmdshow ) ;    // show state of window
```

2.2.8 Event Loop คืออะไร

Main Event Loop คือลูปที่คอยรับ Message Record ที่เก็บไว้จาก Message Queue แล้วทำการแปลง แล้วส่งต่อไปยัง Event Handler



รูปที่ 2.1 The main event loop

โดยมีฟังก์ชันที่ทำหน้าที่ต่างกันอยู่ 3 ฟังก์ชันใน Event Loop นี้ได้แก่

2.2.8.1 ฟังก์ชัน GetMessage

เป็นฟังก์ชันที่ทำหน้าที่รับแมสเสทจาก Message Queue ถ้ากรณี Message Queue ว่าง ฟังก์ชันก็จะทำการรอจนกว่าจะมีแมสเสทเข้ามา

```
BOOL GetMessage ( LPMSG     lpMsg ,          // pointer to message structure
                  HWND      hWnd ,           // handle of window
                  UINT  wMsgFilterMin ,      // first message
                  UINT  wMsgFilterMax        // last message
                ) ;
```

2.2.8.2 ฟังก์ชันTranslateMessage

เป็นฟังก์ชันที่ใช้แปลง Message Event ให้อยู่ในรูปพื้นฐานของอีเวนต์ เพื่อจะนำไปใช้ในส่วนของ Event Handler

```
BOOL TranslateMessage ( CONST  MSG*lpmsg ) ; // pointer to message
                                              structure
```

2.2.8.3 ฟังก์ชัน DispatchMessage

ทำการส่งแมสเสทหลังจากทำการ Translate Message ไปยัง Event Handler เพื่อทำการประมวลผลต่อไป

```
LONG DispatchMessage ( CONST  MSG*lpmsg ) ; // pointer to message
                                             structure
```

2.2.9 Event handler คืออะไร

Event handler หรือ WinProc คือฟังก์ชันที่ต้องเขียนเพื่อแฮนเดิล ทุกแมสเสท ที่แอปพลิเคชั่นนั้นสนใจ วินโดว์อาจจะส่งแมสเสทมามากมาย หลายอย่าง แต่อาจมีเพียง 1 แมสเสทที่เราสนใจจะแฮนเดิล การประกาศต้นแบบ ทำได้ดังนี้

```
LRESULT WINAPI WinProc ( HWND hWnd,      //the window
                         UINT msg,       //the message
                         WPARAM wParam,  //more info on message
                         LPARAM lParam ); //more info on message
```

2.3 Event Driven Programming คืออะไร

ในการเขียนโปรแกรมบนวินโดว์อย่างแรกที่ควรจะเข้าใจคือ ลักษณะของ Event Driven Programming ลักษณะของ Event Driven คือ โปรแกรมจะทำงานตามอีเวนต์ที่เกิดขึ้นโดยระบบปฏิบัติการ จะเป็นตัวคอยบอกโปรแกรมว่ามีอีเวนต์อะไรเกิดขึ้นบ้าง หากโปรแกรมได้รับอีเวนต์ที่โปรแกรมสนใจก็จะทำงานตามที่โปรแกรมเมอร์ได้กำหนดไว้

สำหรับอีเวนต์ในที่นี้ได้รวมถึงทุกอย่างที่เกิดขึ้น ทั้งการได้รับอินพุต เช่น การกดปุ่ม การปล่อยปุ่ม การคลิกเมาส์ เป็นต้น และอีเวนต์ เกี่ยวกับวินโดว์ เช่น การที่วินโดว์ได้รับโฟกัสการสั่งปิดวินโดว์ หรือแม้แต่การ Redraw Window ดังที่ได้กล่าวมาแล้ว ดังนั้น Event Driven Programming ก็คือการเขียนโปรแกรมที่จะทำงานโดยการ Check Message ที่ได้รับไปเรื่อยๆ แล้วทำงานตามที่กำหนดเมื่อมีแมสเสทที่โปรแกรมสนใจ

หากเปรียบกับการเขียนโปรแกรมโดยทั่วไป การทำงานจะเป็นการทำงานแบบเรียงลำดับตามผู้เขียนโปรแกรมได้เขียนไว้ แต่สำหรับ Event Driven Programming นั้นคือ ลักษณะที่ผู้ใช้เป็นผู้ควบคุมโปรแกรมมากกว่าที่โปรแกรมจะควบคุมให้ผู้ใช้ทำตาม ยกตัวอย่างเช่น ในการเขียนโปรแกรมบนดอส ถ้าโปรแกรมจะรับอินพุต 1 คีย์ โปรแกรมเมอร์อาจจะใช้ getc ( ) แต่ถ้าเป็น Event Driven Programming เมื่อมีการกดปุ่ม ระบบปฏิบัติการจะส่งแมสเสทบอกโปรแกรมว่าได้มีการกดแป้นพิมพ์ขึ้น

2.4 วิธีการแสดงภาพ

2.4.1 ความรู้ความเข้าใจเบื้องต้นของการแสดงภาพ

DirectDraw เป็นคอมโพแนนท์ส่วนหนึ่งใน Microsoft Direct X ซึ่งเป็นคอมโพแนนท์ส่วนที่สำคัญที่สุดใน Direct X เพราะว่า DirectDraw มีหน้าที่สำคัญในส่วนที่ใช้วาดหรือติดต่อโดยตรงกับ

ฮาร์ดแวร์ส่วนแสดงภาพ ซึ่งหากขาดส่วน DirectDraw ไป โปรแกรมจะไม่สามารถแสดงภาพได้เลย โดยต่อไปนี้จะกล่าวถึงส่วนที่สำคัญที่สุด

DirectDraw คือ ไลบรารีตัวหนึ่งนั่นเอง หรืออาจจะเรียกได้ว่าเป็น Include File ในภาษาซีขึ้นอยู่กับเวอร์ชันของ Direct X SDK (Software Developer Kit) ซึ่งจะถูกสร้างขึ้นมาให้รองรับภาษาต่างๆโดยหากเป็นของภาษาซีก็จะเป็นดังนี้

DDRAW.H และ DDRAW.LIB

เวลาเรียกใช้ก็ทำการ Include เข้าไปดังนี้

```
# include < ddraw.h >
```

2.4.2 การสร้าง DirectDraw Object

วินโดว์และ DirectDraw สามารถทำงานเข้ากันได้เป็นอย่างดี แต่การที่จะนำมาทำงานร่วมกันจำเป็นต้องมีวิธีการที่มีรูปแบบเฉพาะ โดยการที่เราจะสามารถนำ DirectDraw มาใช้ได้ เราต้องสร้าง DirectDraw Object ขึ้นมาเพื่อเป็นส่วนที่ใช้ติดต่อสื่อสารกับฮาร์ดแวร์ แต่ก่อนที่เราจะสามารถสร้าง DirectDraw Object ได้นั้น เราต้องทำการสร้าง Windows Application ขึ้นมาเป็น Working Windows เสียก่อน เมื่อมีวินโดว์ที่ใช้ทำงานแล้ว เราจึงจะสามารถสร้าง DirectDraw Object ได้ โดยใช้ฟังก์ชัน

```
HRESULT DirectDrawCreate ( GUID FAR*IpGUID , LPDIRECTDEAW
                                FAR*IpIpDD ,  Iunknown FAR* pUnkOuter ) ;
```

ฟังก์ชันนี้จะหมายถึง DirectDrawCreate จะสร้าง DirectDraw Object ไว้ที่ตำแหน่ง Address IpIpDD และจะได้การแสดงผลแบบ IpGUID (ถ้าใส่ค่าเป็น NULL จะได้การแสดงผลแบบ Default) ฟังก์ชันนี้จะทำงานสำเร็จหรือไม่นั้น จะแจ้งผ่านค่า HRESULT มาโดย HRESULT เป็นชนิดข้อมูลชนิดหนึ่งของ Direct X ซึ่งใช้สำหรับตรวจสอบการทำงานของฟังก์ชัน ซึ่งฟังก์ชันส่วนใหญ่ของ Direct X จะทำการคืนค่านี้กลับมา เพื่อตรวจสอบ ค่าที่บ่งบอกถึงความสำเร็จของฟังก์ชันก็คือ

DD_OK

หากฟังก์ชันคืนค่าอื่นๆที่ไม่ใช่ค่านี้ หมายถึงการทำงานของฟังก์ชันมีความผิดพลาด ซึ่งจะแจ้งข่าวสารข้อผิดพลาดมาทาง HRESULT เช่นกัน (แทนที่จะเป็น DD_OK)

<u>ตัวอย่าง</u>

```
LPDIRECTDRAW      Ipdd ;      // ทำการสร้างตัวแปรที่ใช้เก็บค่าแอตเดทของ
                                  DirectDraw Object
                              // ทำการสร้างออปเจคและตรวจสอบข้อผิดพลาด
If ( DirectDrawCreate ( NULL,&Ipdd,NULL ) != DD_OK )
{
        // ทำส่วนนี้เมื่อพบข้อผิดพลาด
```

}

เมื่อได้ทำการสร้างออปเจคของ DirectDraw เรียบร้อยแล้ววิธีการเรียกใช้ฟังก์ชันมีวิธี ดังนี้

Ipdd → Function ( );

เมื่อเราทำงานกับฟังก์ชันนั้นๆเสร็จเรียบร้อยแล้ว เราต้องทำการลบออปเจคนี้ออกจากหน่วยความจำได้โดย

Ipdd → Release ( );

2.4.3 การกำหนดค่า Cooperation Level ของ DirectDraw

การที่จะเขียนโปรแกรมเพื่อทำงานบนวินโดวส์ เราจำเป็นจะต้องทำการขอแบ่งทรัพยากรต่างๆ จากสิ่งแวดล้อมบนวินโดวส์ ซึ่ง DirectDraw ก็ได้ให้ฟังก์ชัน เพื่อใช้ในการกำหนดค่าความสำคัญและกำหนดระบบที่จะใช้โปรแกรมของเรา ดังนี้

HRESULT SetCooperativeLevel (HWND hWnd , DWORD dwFlags) ;

HWND hWnd // คือ แฮนเดิลของวินโดวส์หรือโปรแกรมที่กำลังทำงานด้วย โดยปกติแล้วเราเพียงแต่ทำการกำหนดค่าตัวแปรนี้แล้วคอมไพเลอร์จะทำการสร้างให้เองโดยอัตโนมัติ

DWORD dwFlags คือ ค่าที่จะกำหนดให้กับฟังก์ชัน มีดังตารางที่ 2.1

ค่า Flag	ความหมาย
DDSCL_ALLOWMODEX	อนุญาตให้ใช้ ModeX ได้ (320x200 , 320x240) โดยค่านี้จะต้องใช้คู่กับ DDSCL_EXCLUSIVE และ DDSCL_FULLSCREEN
DDSCL_ALLOWREBOOT	อนุญาตให้ใช้ Ctrl+Alt+Delete ในขณะที่อยู่ในโหมด Exclusive (fullscreen)
DDSCL_EXCLUSIVE	ทำการกำหนดให้โปรแกรมเป็นแบบ Exclusive (ให้โปรแกรมมีลำดับความสำคัญสูงสุด) โดยจะต้องทำงานคู่กับ DDSCL_FULLSCREEN
DDSCL_FULLSCREEN	ค่านี้จะบอกให้วินโดวส์อนุญาตให้เฉพาะ DirectDraw เท่านั้นที่เป็นผู้เขียนข้อมูลลงบนหน่วยความจำแสดงผลทั้งหมด
DDSCL_NORMAL	บ่งบอกให้วินโดวส์ทราบว่าโปรแกรมนี้จะเป็นเหมือน แอปพลิเคชั่นทั่วๆไป (เช่น Microsoft

	Word หรือ ACDSee) โดยหากเรากำหนดค่านี้แล้ว เราจะไม่สามารถกำหนดค่า DDSCL_ALLOWMODEX , DDSCL_EXCLUSIVE หรือ DDSCL_FULLSCREEN ได้
DDSCL_NOWINDOWCHANGES	บอกว่าโปรแกรมนี้ไม่สามารถที่จะมินิไมท์หรือรีสตอร์ โปรแกรมอื่นๆในขณะที่โปรแกรมนี้ทำงานอยู่ได้

ตารางที่ 2.1 ค่าCooperative Level ของ DirectDraw

ตัวอย่าง

Ipdd → SetCooperativeLevel (hwnd , DDSCL_NORMAL)

หมายถึงกำหนดให้โปรแกรมเราเป็นแบบแอปพลิเคชันทั่วๆไป

Ipdd → SetCooperativeLevel (hwnd , DDSCL_ALLOWMODEX
IDDSCL_FULLSCREEN
IDDSCL_EXCLUSIVE
IDDSCL_ALLOWREBOOT) ;

หมายถึงโปรแกรมเราสามารถใช้ ModeX ได้ และจะเป็นแบบ FullScreen ซึ่งแสดงผลเต็มหน้าจอ และมีความสำคัญสูงสุด และยังสามารถออกจากโปรแกรมได้โดยใช้ปุ่ม Ctrl+Alt+Delete

2.4.4 การกำหนดโหมดการแสดงผล

ความสามารถในการเปลี่ยน โหมดการแสดงผล คือ ความสามารถที่สำคัญที่สุดของ DirectDraw , ฟังก์ชันที่ใช้เปลี่ยน โหมดการแสดงผลคือ

HRESULT SetDisplayMode (DWORD dwWidth ,
DWORD dwHeight ,
DWORD dwBPP ,
DWORD dwRefreshRate ,
DWORD dwFlags) ;

โดยส่วนที่จำเป็นต้องกำหนดค่า มีสามส่วนด้วยกัน

dwWidth กำหนดความกว้างของการแสดงผลโดยมีหน่วยเป็นพิกเซล

dwHeight กำหนดความยาวของการแสดงผลโดยมีหน่วยเป็นพิกเซล

dwBPP กำหนดจำนวนข้อมูลสีต่อหนึ่งพิกเซล (8 , 16 , 32 bit)

นอกนั้นให้กำหนดค่าเป็น 0 ทั้งหมด เพราะคอมไพเลอร์จะทำการตรวจสอบค่าที่เหมาะสมให้เอง ทั้งหมดที่กล่าวมานี้เป็นส่วนที่ใช้สำหรับเริ่มการใช้งาน DirectDraw โดยสรุปได้ดังนี้

การเริ่มการใช้งาน DirectDraw มีสามขั้นตอนดังนี้

- ทำการสร้าง DirectDraw Object
- ทำการกำหนดระดับความสำคัญและการแบ่งใช้ทรัพยากรโดย SetCooperativeLevel
- ทำการกำหนดขนาดการแสดงผล โดยใช้ SetDisplayMode

2.4.5 การประยุกต์ใช้ DirectDraw

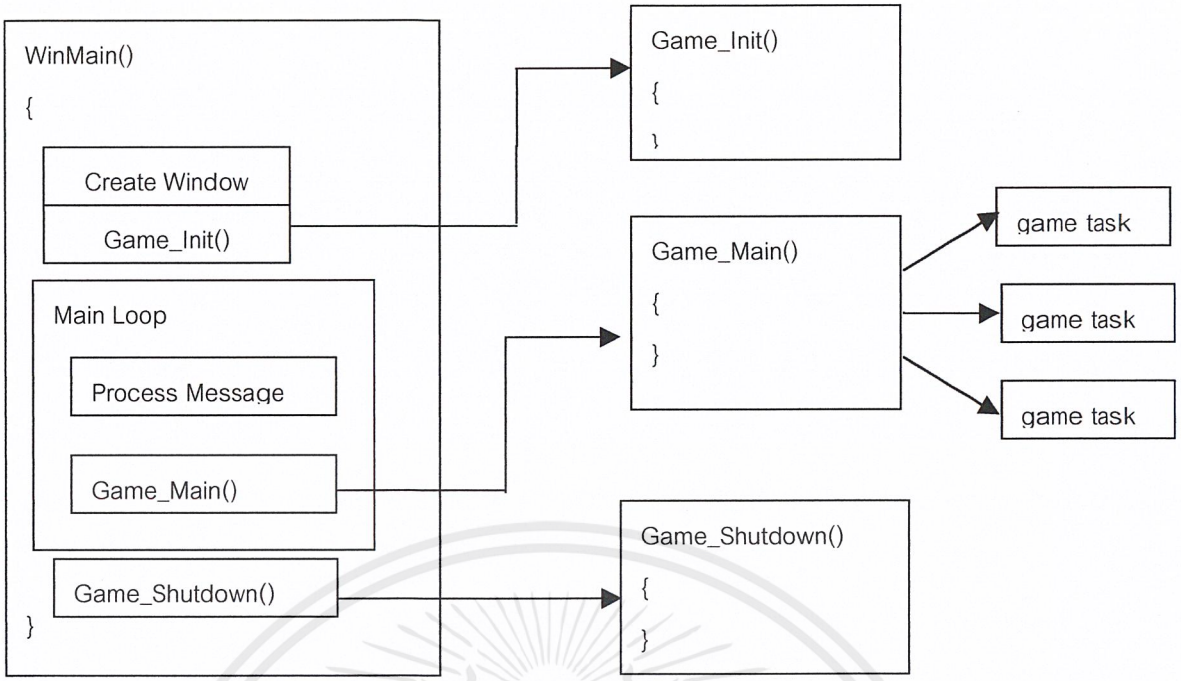
DirectDraw สามารถนำมาใช้ได้หลายๆด้าน แต่ด้านที่ถือว่านิยมที่สุดและนำ DirectDraw มาใช้มากที่สุดก็คือด้านการสร้างเกมคอมพิวเตอร์ ในปัจจุบัน เกมส่วนมากนำ DirectDraw มาใช้ในการสร้างอนิเมชั่น ซึ่ง DirectDraw สามารถเข้าถึงฮาร์ดแวร์ด้านการแสดงผลได้โดยตรง ดังนั้นการทำงานต่างๆจึงรวดเร็วมาก และผู้ใช้ไม่จำเป็นต้องเข้าไปยุ่งเกี่ยวกับคำสั่งในระดับฮาร์ดแวร์เลย

2.4.6 การวาดภาพด้วย DirectDraw

วิธีที่ใช้ในการวาดภาพ ก็คือการนำภาพที่เราได้วาดไว้แล้วจากโปรแกรมวาดภาพต่างๆ เช่น Adobe Illustrator หรือจะ Adobe Photoshop หรือเราอาจนำภาพมาจากโปรแกรมสร้างภาพสามมิติ เช่น 3Dstudio ก็ได้ แต่ภาพจะเป็นแบบ BMP โดยเราจะนำภาพที่เราได้สร้างไว้แล้วเก็บไว้ในหน่วยความจำ หรือที่เรียกว่าเซอร์เฟสแล้วทำการถ่ายเทข้อมูลไปยังหน่วยความจำที่การ์ดแสดงผล หลังจากนั้นจึงนำแสดงขึ้นจอภาพ

2.5 โครงร่างการเขียนเกม

หลังจากศึกษาการโปรแกรมที่เกี่ยวกับวินโดว์ไปแล้ว ในส่วนนี้จะกล่าวถึงโครงร่างการเขียนเกม (Game Console) ที่ถูกออกแบบมาเพื่อใช้ในการเขียนเกมโดยตรง โดย game console จะทำงานภายใน WinMain ดังแสดงตามรูปที่ 2.2



รูปที่ 2.2 โครงร่างการเขียนเกม

จุดประสงค์ของการสร้าง Game Console ก็เพื่อสร้างเกมที่รันบนวินโดว์และใช้ DirectX ได้ด้วยความสะดวก ดังนั้นในปัญหาพิเศษนี้ จะกระทำบน Game Console ซึ่ง Game Console แบ่งออกเป็น 3 ฟังก์ชันได้แก่ Game_Init(), Game_Shutdown() และ Game_Main()

2.5.1 Game_Init

มีหน้าที่กำหนดค่าเริ่มต้นให้แก่ระบบ เช่น ปรับเปลี่ยนโหมดการแสดงผล เตรียมบัฟเฟอร์ ตั้งค่าการ์ดเสียง เป็นต้น โดยจะทำงานเมื่อวินโดว์ได้ตั้งระบบเริ่มต้นไว้เรียบร้อยแล้ว

2.5.2 Game_Main

เป็นเสมือนฟังก์ชันหลัก โดยตัวโปรแกรมจะอยู่ในส่วนนี้ และอยู่ในลูปเพื่อประมวลผลไปเรื่อยๆ จนกว่าโปรแกรมจะทำงานเสร็จ หรือออกจากโปรแกรม

2.5.3 Game_Shutdown

ทำหน้าที่คืนทรัพยากรต่างๆให้กับระบบ เมื่อพร้อมที่จะจบการทำงาน จะถูกเรียกหลังจากที่โปรแกรมทำงานเสร็จ เตรียมที่จะปิดโปรแกรม

2.6 พื้นฐานกราฟฟิก 3 มิติ

หลังจากที่เราทำการสร้างหน้าจอทั้งแบบ Windowed Mode และ Fullscreen Mode แล้ว ขั้นต่อไปจะมาดูถึงการสร้างรูปภาพ 3 มิติ และนำรูปภาพ 3 มิติ ที่สร้างขึ้นไปใช้งานร่วมกับโปรแกรม ซึ่งในการสร้างรูปภาพ หรือการเขียนโปรแกรมในส่วนของการทำงานนี้ ก่อนอื่นจำเป็นจะต้องมีความรู้เกี่ยวกับพื้นฐานทางด้าน 3 มิติก่อน เพราะศัพท์บางคำที่ใช้อยู่ในโปรแกรม 3 มิติ จะถูกนำมาใช้

ร่วมกับการเขียนโปรแกรมเชื่อมต่อกับ Direct X ด้วย ซึ่งในบางครั้งถ้าหากไม่เรียนรู้พื้นฐานเหล่านี้ก่อนก็อาจจะเกิดปัญหาในการเขียนโปรแกรมที่ดีได้ โดยพื้นฐานทางด้าน 3 มิติที่ควรรู้ และมีความสำคัญในการเขียนโปรแกรมจะมีดังนี้

2.6.1 Vertex

จุด (point) ซึ่งแต่ละ Vertex จะเชื่อมต่อกันด้วย line และเมื่อทุกๆ Vertex มีการเชื่อมต่อกันหมดแล้วจะทำให้เกิดรูปทรง หรือ shape ขึ้น โดยลักษณะของ Vertex มีดังนี้

รูปที่ 2.3 แสดงลักษณะของจุด Vertex

และโครงสร้างของ Vertex มีดังนี้

```
Typedef struct vertex
{
        int     x, y, z, w;
}VERTEX;
```

และในส่วนของโครงสร้างของ shape มีดังนี้

```
Typedef struct shape
{
        int             numVerterts;
        VERTEX*         vertices;
}SHAPE;
```

2.6.2 Vector

เวกเตอร์คล้ายจุด (point) แต่ 1 เวกเตอร์จะประกอบด้วยค่า 3 ค่า คือ x, y, z โดยเวกเตอร์จะอธิบายถึงทิศทางและความเร็ว แต่ไม่รวมถึงตำแหน่งที่ตั้ง (location) ตามจริงแล้วเวกเตอร์ประกอบ

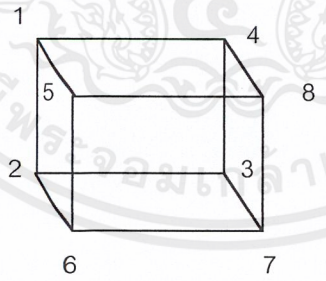
ด้วยค่าทั้งหมด 6 ค่าด้วยกัน คือ x, y, z ของจุดเริ่มต้น และค่า x, y, z ของจุดสิ้นสุด ซึ่งสิ่งเหล่านี้เองที่ทำให้เกิดทิศทางและความเร็ว แต่โดยทั่วไปแล้วมักใช้เวกเตอร์ที่มีประกอบด้วยค่า 3 ค่า คือ x, y, z ของจุดสิ้นสุด และมีจุดเริ่มต้นอยู่ที่จุดออร์ริจินคือ (0,0,0) นั่นเอง

เวกเตอร์กับ จุด (point) ต่างกันตรงที่ จุด (point) นั้นจะระบุถึงตำแหน่งที่ตั้ง แต่เวกเตอร์นั้นไม่สนใจ และในทางกลับกันที่ตั้งจะถูกใช้ในการกำหนดเวกเตอร์ แต่ เวกเตอร์ไม่ได้เป็นตัวกำหนดที่ตั้ง Unit Vector คือ เวกเตอร์ ที่มีความยาว = 1.0 และเป็นสับเซ็ตพิเศษของเวกเตอร์ Normal Vector เป็นชนิดของ Unit Vector ใช้ในการคำนวณแสงต่างๆของ face และ mesh โดย normal มี 2 ชนิด คือ

- Face normal คือ Unit Vector ของ face มักใช้ในการ shading สีของ face ที่แสดงออกมา ในด้านข้างของ face โดยทิศทางของ face normal ขึ้นอยู่กับ vertices และ handedness ของ coordinate system vector
- Vertex normal คือ Unit Vector ที่เชื่อมกับ vertex และสามารถถูกคำนวณได้โดยหาค่าเฉลี่ยของ face normal ซึ่ง share กับ vertex ,Vertex normal มักใช้ในการคำนวณหาค่า สีของ vertex แต่การใช้ face normal กับ vertex normal ขึ้นอยู่กับวิธีการเรนเดอร์ด้วย

2.6.3 Vertices

เซ็ตของจุดที่ใช้แทนตำแหน่งของวัตถุ เช่น face หรือ mesh ใน 3D Vertices เหล่านี้เป็นการสร้างบล๊อคที่ง่ายที่สุดในการสร้าง polygon mesh แต่ละ vertices อาจจะใช้แทนค่า3 ค่า ซึ่งแต่ละ vertices จะมีลักษณะเฉพาะของจุด หรือโคออร์ดิเนทในรูปทรง 3 มิติ



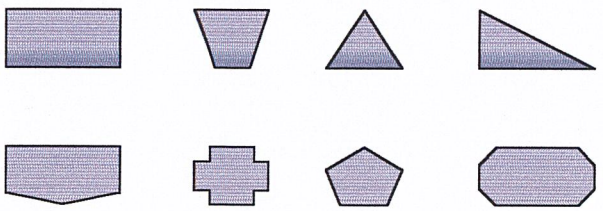
รูปที่ 2.4 แสดงลักษณะของ Vertices

2.6.4 Face

Set ของ vertices ที่มีตั้งแต่ 3 จุดขึ้นไป มาเชื่อมต่อกันในลักษณะของรูปทรงต่างๆ ในแนวระนาบ vertex เป็นตัวกำหนดมุมของ face ซึ่งทำให้ทุกๆ vertices ใน face จะต้องถูกกำหนดให้อยู่ในแนวระนาบ

2.6.5 Polygon

รูปทรงที่สร้างขึ้นโดยการวาดเส้นเชื่อมระหว่างเซ็ตของ vertices โดยที่ 1 โพลีกอนจะต้องมีอย่างน้อย 3 เส้น



รูปที่ 2.5 แสดงลักษณะของโพลีกอน

ในการวาดรูปที่มีส่วนโค้งเราไม่นิยมที่จะวาดเส้นโค้งนั้นโดยตรง แต่เรานิยมที่จะนำโพลีกอนหลายๆโพลีกอนมาเรียงต่อกันเพื่อให้ได้ส่วนโค้งมากกว่า ทั้งนี้เนื่องมาจากว่าคอมพิวเตอร์ถนัดที่จะคำนวณเส้นตรงมากกว่าเส้นโค้ง หากใช้เส้นโค้งโดยตรงการคำนวณจะทำได้ช้า และอาจเกิดข้อผิดพลาดได้ง่าย จึงนิยมที่จะให้โพลีกอนมาเรียงต่อกันมากกว่า ยิ่งใช้โพลีกอนจำนวนมากเท่าใดยิ่งทำให้ส่วนโค้งนั้นดูสมจริงมากเท่านั้น อีกทั้งยังประหยัดเวลาในการคำนวณของคอมพิวเตอร์ด้วย

2.6.6 Mesh คืออะไร

Mesh เป็นการรวมกันของ face ที่เชื่อมต่อกัน ซึ่ง 1 mesh สามารถมี face ได้ตั้งแต่ 1 face ขึ้นไป การรวมของ face นี้ทำให้ง่ายต่อการจัดการวัตถุในการทำอนิเมชั่น , material และ texture ชนิดของการรวมของ face มี

- Fan คือ กลุ่มของรูปทรงสามเหลี่ยม ซึ่งทุกรูปมีการใช้ vertex ร่วมกัน 1 จุด โดยจะกำหนดให้ vertex นั้นๆอยู่ระหว่างกลางของสามเหลี่ยมเหล่านั้น ชนิดของการรวม face แบบ fan นี้คล้ายกับแบบ strip คือมีการกำหนดค่า vertices 3 ค่าแรกสำหรับสามเหลี่ยมแรก จากนั้นในสามเหลี่ยมต่อไปก็เพียงแค่เพิ่มขึ้นมารูปละ 1 vertex เท่านั้น
- Strip คือ กลุ่มของรูปทรงสามเหลี่ยม ซึ่งแต่ละรูปจะมีการใช้เส้นร่วมกันกับสามเหลี่ยมส่วนหน้า ซึ่งก็หมายความว่า หลังจากที่มีการกำหนดค่า vertices 3 ค่าแรกสำหรับสามเหลี่ยมแรก จากนั้นในสามเหลี่ยมต่อไปก็เพียงแค่เพิ่มขึ้นมารูปละ 1 vertex นั้นเอง
- List คือ กลุ่มของรูปทรงสามเหลี่ยม ซึ่งทุกรูปไม่มีการใช้เส้น หรือ vertex ร่วมกันเลย กล่าวคือ ค่า 3 ของ vertices ทั้ง 3 ค่าของสามเหลี่ยมทุกรูปต้องกำหนดเองทั้งหมด

2.6.7 Texture

บิตแมพซึ่งเป็นแพทเทิลหรืออิมเมจ มักจะเก็บอยู่ในรูปแบบของ BMP, PCX หรือ GIF ซึ่งสามารถสร้างความสมจริงให้แก่วัตถุ คือเมื่อเราต้องการให้วัตถุที่เราสร้างขึ้นมีความสมจริง เราควรจะสร้างรายละเอียดให้กับพื้นผิวของวัตถุด้วยเช่นกัน เพื่อเพิ่มเติมคุณสมบัติหลักๆให้กับวัตถุ ยกตัว

อย่างเช่น เมื่อเราต้องการสร้างวัตถุที่เป็นลูกบอล นอกจากที่เราจะทำรูปทรงกลมแล้ว เราควรจะเพิ่มลายที่เป็นลายของลูกบอลเข้าไปเพื่อเพิ่มความสมจริงของลูกบอล เพราะถ้าหากเราปล่อยให้ลูกบอลที่เราสร้างขึ้นเป็นเพียงรูปทรงกลมแล้วละก็เราจะได้เพียงลูกบอลที่มีรูปร่างเหมือนลูกบอลที่ไร้จินตนาการเท่านั้น Texture coordinate ใช้กำหนดการเชื่อมต่อกันระหว่าง vertices ของ face กับพิกเซลของ การเชื่อมต่อกับบิตแมพ โดย texture coordinate นี้ใช้แทน 2 มิติของ coordinate system

Texture mapping คือการวาดรูปลงบนผิวหน้าของ face หรือโพลีกอน และในการทำ texture mapping นี้ต้องคำนึงถึงการคำนวณค่าต่างๆด้วย จึงต้องมีการกำหนดค่าของ vertices ด้วย จากการที่ texture coordinate กำหนดพิกเซลของ texture ที่จะวาดลงในส่วนของ face แล้วก็จะมี การ wrapping เพื่อ generate texture coordinate

สำหรับออปเจคนั้น ซึ่งการ wrapping นั้นมี 4 ชนิดคือ

- Flat wrap apply ไปสู่ face โดยตรง
- Cylindrical wrap ทำ wrap texture map ใน cyline รอบๆออปเจค
- Spherical wrap ทำ wrap texture map ใน sphere รอบๆออปเจค
- Chrome wrap ทำการ generate texture coordinate ไปยังออปเจคที่มีปฏิกิริยากับ texture

2.6.8 Light

แสงทำให้เกิดการเปลี่ยนแปลงของสีของ Vertices โดย Module ทำให้เกิด vertex normal เพราะสิ่งนี้ขึ้นอยู่กับมุมของแหล่งกำเนิดแสง โดยทั่วไปแล้วแหล่งกำเนิดแสงตามปกติจะมีแสงสีขาว เพราะเป็นการรวมกันอย่างหนาแน่นของสีทุกสี และโดยมากมักใช้รูปแบบของ RGB ในการกำหนดสีของแหล่งกำเนิดแสงใน 3D นั้นมีการตั้งค่า RGB ของแม่สีต่างๆดังนี้

แสงสีขาว เป็น 1, 1, 1

แสงสีแดง เป็น 1, 0, 0

แสงสีน้ำเงิน เป็น 0, 0, 1

นอกจากนี้เรายังสามารถใช้ 3 สี นี้ในการผสมสีได้

แหล่งกำเนิดแสงมี 3 ชนิด ได้แก่

- Ambient Light คือแหล่งกำเนิดแสงที่ง่ายที่สุด เพราะไม่ต้องมีการกำหนดตำแหน่งของแหล่งกำเนิดแสง และยังให้ความสว่างทั่วทุกออปเจค
- Point Light เป็นแหล่งกำเนิดแสงที่ทำการกระจายแสงไปทุกทิศทาง แต่ต้องระบุตำแหน่งของแหล่งกำเนิดแสง โดยไม่ต้องกำหนดทิศทางของแสง
- Directional Light เป็นแหล่งกำเนิดแสงที่มีประสิทธิภาพมากที่สุด เพราะเป็นแหล่ง กำเนิดแสงที่มีทิศทาง โดยต้องระบุตำแหน่งของแหล่งกำเนิดแสง

- Spot Light เป็นแหล่งกำเนิดแสงที่ต้องมีการระบุทั้งทิศทางและตำแหน่งของแหล่งกำเนิดแสง โดยการผลิตแสงจะเป็นรูปทรง

2.6.9 Device

ออปเจคที่สร้างขึ้นเพื่อใช้ในการวาดภาพจากมุมกล้อง โดย device มี 2 ชนิดหลักๆคือ

- Software device ซึ่งจะอนุญาตให้โปรแกรมสามารถวาดภาพโดยไม่ต้องนำไปคำนวณที่การ์ดจอแบบ 3D accelerated
- Hardware device สามารถทำงานบนคอมพิวเตอร์ที่มี 3D ฮาร์ดแวร์เท่านั้น และจะทำให้ Direct 3D สามารถเรียกใช้คุณสมบัติต่างๆในการเรนเดอร์ภาพได้อย่างเต็มที่ ทำให้จำนวนเฟรม/วินาทีมีค่ามากขึ้น

2.6.10 Viewport ใช้อย่างไร

อยู่ในส่วนของกล้อง โดยวิวพอร์ตนั้นจะรวมไปถึงตำแหน่งและทิศทางของฉาก(scene) จากการมองเห็น ซึ่งเรามักใช้วิวพอร์ตร่วมกับ field of view front & back clipping และ perspective tranformation

วิวพอร์ตประกอบด้วย

- Eyepoint คือจุดที่ตั้งกล้อง
- Lookat คือจุดที่กล้องมอง
- Upvector คือ vector ด้านบนของกล้อง

การดูออปเจคมี 2 ชนิด

- Perspective เป็นวิธีที่ทำให้รูปที่วาดลงบนฉาก(Screen) มีลักษณะที่ใกล้เคียงกับออปเจคจริงๆมากที่สุด เพราะจะสนใจทั้งจุด x, y, z ทำให้เกิดความลึกของภาพนั่นเอง
- Orthographic เป็นวิธีที่ทำให้รูปที่วาดลงบนฉาก(Screen) แบบง่ายๆจึงละเลยในส่วนของจุด z ไป ทำให้ภาพที่ได้คล้ายกับ 2 มิติ

2.6.11 การใช้ Animation ทำอย่างไร

การทำกราฟฟิก 3D นั้นจะไม่ดีถ้าหากขาดส่วนของอนิเมชั่นไป โดยการทำอนิเมชั่นให้สำเร็จได้นั้นขึ้นอยู่กับ motion attribute และ key framing โดยตำแหน่งของออปเจคถูกอธิบายโดยคีย์และแต่ละคีย์จะมีการกำหนดเวลาเพื่ออธิบายการเปลี่ยนแปลงของออปเจค

2.6.11.1 Motion attribute เป็นทิศทางการเคลื่อนที่ของอนิเมชั่น ซึ่ง motion attribute ประกอบด้วย

- translation (การเปลี่ยนตำแหน่ง)
- rotation (การหมุน)
- scale (การเปลี่ยนขนาด)

เรามักจะใช้ประโยชน์ motion attribute ในเรื่องของการเคลื่อนที่แบบซ้ำๆ

2.6.11.2 Key framing โดย key framing ในกราฟฟิก 3D นั้น หมายถึงการที่เรากำหนดตำแหน่งของออปเจคในฉาก(scene) ที่ key time นั้นในอนิเมชั่น และนอกจากนั้นยังรวมไปถึงจำนวนของเฟรมในอนิเมชั่น และการกำหนดคีย์ในเฟรมใน Direct 3D มี key framing 2 ชนิดคือ

- Linear keyframing ซึ่งหมายถึงการเคลื่อนที่ของอนิเมชั่นระหว่าง คีย์เฟรม โดยออปเจคจะเคลื่อนที่เป็นเส้นทางสั้นๆระหว่างคีย์เฟรม
- Spline-based keyframing ซึ่งหมายถึงการเคลื่อนที่ของอนิเมชั่นระหว่างคีย์เฟรม โดยออปเจคจะเคลื่อนที่เป็นเส้นโค้งระหว่างคีย์เฟรม

2.6.12 Camera คืออะไร

มุมกล้องที่ทำให้เราสามารถมองเห็นวัตถุจากทิศทางต่างๆ เปรียบเสมือนว่าเรากำลังนำกล้องไปตั้งไว้บริเวณพื้นที่ที่มีวัตถุดังกล่าว ดังนั้นการเปลี่ยนมุมกล้องก็จะทำให้ภาพที่ปรากฏบนจอมีลักษณะแตกต่างกันไป การสร้างวัตถุเพียงชิ้นเดียวอาจจะไม่มีปัญหาอะไรในการมองรูปภาพ แต่ถ้าเราสร้างวัตถุหลายๆชิ้น หลายรูปทรงใน 1 ภาพ การลำดับตำแหน่งของวัตถุ จึงเป็นสิ่งจำเป็นที่เราจะต้องฝึกให้คุ้นเคยและสามารถมองภาพให้เป็นมิติและสมจริงได้ไม่ว่าภาพนั้นจะแสดงผลในรูปแบบใด ดังนั้นลำดับเป็นเรื่องที่เราต้องแยกแยะให้ออก หากว่าเรายังไม่สามารถที่จะมองภาพที่เราสร้างขึ้นให้เป็นมิติได้ ก็ยากที่จะรู้ได้ว่าภาพหรือวัตถุชิ้นไหนอยู่หน้าหรืออยู่หลัง และอาจทำให้เราเกิดความสับสนในเรื่องของการลำดับวัตถุ และทำให้การสร้างภาพหรือวัตถุมีความยุ่งยากซับซ้อนมากยิ่งขึ้น จนไม่สามารถสร้างเกมที่มีความสมบูรณ์และมีประสิทธิภาพได้ดังนั้นในการเขียนเกมเราจึงควรศึกษาและทำความเข้าใจในการใช้มุมกล้อง รวมถึงเทคนิคพื้นฐานอื่นๆ เพื่อเพิ่มประสิทธิภาพให้กับเกมมากยิ่งขึ้น

2.7 วิธีการรับข้อมูลจาก Keyboard

DirectInput เป็นไลบรารีที่ช่วยให้โปรแกรมเมอร์ทำการติดต่อและเรียกใช้อุปกรณ์ในการรับอินพุทชนิดต่างๆได้อย่างมีประสิทธิภาพ DirectInput คลอบคลุมอุปกรณ์ในการรับอินพุมาตราฐานทุกชนิด อันประกอบไปด้วย

- คีย์บอร์ดมาตราฐาน
- เมาส์ทั้ง 2 และ 3 ปุ่ม
- จอยสติ๊ก จอยโยก ทั้งที่เป็นแบบอนาลอค และดิจิตอล
- อุปกรณ์ควบคุมการขับเคลื่อน ใช้สำหรับจำลองการขับรถทั้งที่เป็นแบบอนาลอค และดิจิตอล
- อุปกรณ์ที่ตอบสนองแรงสั่นสะเทือน จอยสติ๊กหรืออุปกรณ์อื่นๆที่สามารถคำนวณเพื่อทำให้เกิดแรงสั่นโต้ตอบกับผู้ใช้

2.7.1 การใช้ DirectInput

DiretcInput ประกอบไปด้วยรันไทม์ไลบรารี และ 2 คอมไพล์ไฟล์ DINPUT.LIB, DINPUT.H สำหรับโปรแกรมหรือโปรเจคที่ต้องการใช้ความสามารถของ DirectInput จะต้องเพิ่มไฟล์ตามที่ได้กล่าวมาข้างต้นไว้ในโปรเจคนั้นๆด้วย

ขั้นตอนในการใช้ DirectInput

- สร้างออปเจค DirectInput หลักด้วย DirectInputCreate( ).
- สร้างออปเจคของอุปกรณ์อินพุทด้วย CreateDevice( ).
- กำหนดลักษณะในการใช้งานของแต่ละอุปกรณ์ด้วย SetCooperativeLevel( ).
- กำหนดรูปแบบข้อมูลของแต่ละอุปกรณ์ด้วย SetDataFormat( ). และทำการกำหนดคุณสมบัติพิเศษของอุปกรณ์ต่างๆด้วย SetProperty( ).
- ทำการจองอุปกรณ์ต่างๆด้วย Acquire( ).
- รับข้อมูลจากแต่ละอุปกรณ์ที่ได้ทำการจองไว้แล้วด้วย GetDeviceState( ).

การอ่านข้อมูลของแต่ละอุปกรณ์จะถูกอ่านเป็นเรคคอร์ด แต่ละเรคคอร์ดจะถูกแปลงเป็นข้อมูลเพื่อให้โปรแกรมเมอร์นำข้อมูลต่างๆเหล่านี้ไปใช้ แต่ละอุปกรณ์ก็จะมีรูปแบบของเรคคอร์ดที่แตกต่างกัน ถึงแม้ว่าจะมีรูปแบบที่แตกต่างกัน แต่เรคคอร์ดเหล่านี้ก็มีลักษณะใกล้เคียงกันมาก ทำให้เกิดการสร้างข้อมูลอินพุทที่เป็นแบบหลักๆขึ้น

โปรแกรมเมอร์สามารถร้องขอข้อมูลอินพุทจาก DirectInput ได้ 2 วิธีคือ

- แบบทันทีทันใด (Immediate Input) ข้อมูลที่ได้จะเป็นสถานะปัจจุบันของอุปกรณ์อินพุทขณะนั้น
- แบบมีการพักข้อมูล (Buffered Input) ข้อมูลจะถูกเก็บเป็นดาต้าเบสของเหตุการณ์ที่เกิดขึ้นตั้งแต่การร้องขออินพุทครั้งสุดท้าย

ในการใช้ DirectInput โปรแกรมเมอร์ไม่ต้องกังวลเกี่ยวกับ แมสเสท หรือการตอบสนองกับผู้ใช้ว่าจะเป็นไปอย่างเชื่องช้า เพราะ DirectInput จะทำงานโดยตรงกับไดรเวอร์ของอุปกรณ์อินพุทนั้นๆ

2.7.2 การติดต่อกับ Keyboard ผ่านทาง DirectInput

2.7.2.1 DirectInput Object จะสร้างออปเจคหลักเพื่อใช้ในการจัดการ ติดต่อกับอุปกรณ์อินพุทชนิดอื่น โดยออปเจคหลักนี้ถูกอ้างถึงด้วย

```
LPDIRECTINPUT        lpdi;
```

ทำการสร้างออปเจคหลักด้วยฟังก์ชั่น DirectInputCreate( ) โดยมีรูปแบบของฟังก์ชั่นดังนี้

```
HRESULT DirectInputCreate(
                    HINSTANCE       hinst,
                    DWORD           dwversion,
```

LPDIRECTINPUT *lplpDirectInput,

LPUNKNOWN punkOuter);

ฟังก์ชั่นจะคืนค่า DI_OK(DirectInput Okey) ถ้าการเรียกใช้ฟังก์ชั่นนี้ผ่านและจะคืนค่าเป็นผิดพลาดอื่นๆถ้าฟังก์ชั่นนี้ไม่ผ่าน สำหรับค่าพารามิเตอร์ต่างๆของ DirectInputCreate มีดังนี้

hinst ใช้อ้างถึงแอพพลิเคชั่นที่ต้องการใช้งาน DirectDraw ให้แน่ใจว่าค่า hinst นี้ถูกส่งค่ามาจากค่า hinstance ใน WinMain( )

dwversion กำหนดเวอร์ชั่นของ DirectInput ว่าจะใช้เวอร์ชั่นใด มีไว้เพื่อให้โปรแกรมเมอร์ที่ต้องการใช้ DirectInput เวอร์ชั่นเก่า แต่โดยทั่วไปจะถูกกำหนดค่าเป็น DIRECTINPUT_VERSION ซึ่งหมายความว่า DirectInput ที่จะใช้เป็นเวอร์ชั่นล่าสุด

lplpDirectInput เป็นพอยเตอร์ที่อ้างอิงไปยังออปเจคหลักของ DirectInput

punkOuter ถูกกำหนดเป็น NULL เสมอ

<u>ตัวอย่าง</u> การสร้างออปเจคหลักของ DirectInput

```
LPDIRECTINPUT lpdi;
DirectInputCreate(hinstance, DIRECTINPUT_VERSION, &lpdi, NULL);
```

เมื่อไม่ต้องการใช้งานออปเจคหลักของ DirectInput ก็ทำการเคลียร์ค่าด้วยฟังก์ชั่น Release( ) เช่น

lpdi → Release( )

2.7.2.2 ทำการสร้างออปเจคของคีย์บอร์ด ด้วยฟังก์ชั่น CreateDevice( ) ฟังกัชั่นนี้จะใช้ เมื่อต้องการสร้างออปเจคของอุปกรณ์อินพุทใดๆ รูปแบบการใช้งานฟังก์ชั่น CreateDevice( ) มีดังนี้

```
HRESULT CreateDevice(
        REFGUI                  rguid,
        LPDIRECTINPUTDEVICE     lplpDiDev,
        LPUNKNOWN               punkOuter);
```

ค่าพารามิเตอร์ต่างๆของ DirectInput มีดังนี้

rguid คือค่า GUID(Globally Unique Identifier) เป็นหมายเลขของอุปกรณ์อินพุทที่ต้องการสร้าง โดยปกติจะใช้ฟังก์ชั่น EnumDevice( )เพื่อหาค่า GUID ของอุปกรณ์อินพุทนั้นๆแต่ใน DirectInput มีการกำหนดค่า GUIDของคียบอรด์และเมาส์นั้นๆไว้เรียบร้อยแล้ว โดยจะมีการกำหนดค่าเป็น

- GUID_SysKeyboard สำหรับคีย์บอร์ดมาตรฐาน
- GUID_SysMouse สำหรับเมาส์มาตรฐาน

อย่างไรก็ตามการใช้ค่า GUID ที่ถูกประกาศไว้แล้วนี้จำเป็นต้องเพิ่ม #define INIGUIC ไว้บนส่วนของทุกไฟล์ C/C++ ที่มีการประกาศ OBJBASE.H เพื่อคอมไพเลอร์จะรวบรวมข้อมูลเกี่ยวกับ COM และค่า GUID ที่ได้ถูกประกาศไว้แล้ว

lplpDiDev เป็นตำแหน่งที่พอยเตอร์ชี่เพื่ออ้างอิงถึงอุปกรณ์อินพุทชนิดนั้นๆ

punkOuter ถูกกำหนดให้เท่ากับ NULL เสมอ

ตัวอย่าง การสร้างออปเจคของคีย์บอร์ด

```
#define INITGUID
#include <OBJBASE.H>
#include <DINPUT.H>
//ทำการสร้างออบเจ็คหลักตามที่ได้กล่าวมาในขั้นก่อนหน้านี้หลังจากสร้างออบเจ็คหลักแล้วจึงทำการสร้างออบเจ็คของคียบอร์ด
        LPDIRECINPUTDEVICE          lpdikey;
        lpdi → CreateDevice(GUID_SysKeyboard, &lpdikey , NULL);
```

2.7.2.3 กำหนดลักษณะการใช้งานของคีย์บอร์ด

แต่ละอุปกรณ์ก็จะมีลักษณะการใช้งานเป็นของตัวเอง SetCooperativeLevel( ) มีรูปแบบการเรียกใช้ดังนี้

```
HRESULT SetCooperativeLevel(
        HWND        hwnd,
        DWORD       dwFlag);
```

ค่าพารามิเตอร์ต่างๆของ SetCooperativeLevel( ) มีดังนี้

- hwnd ใช้อ้างถึงวินโดว์ของแอพพลิเคชั่นนั้นๆ
- dwFlag เป็นค่ามี่กำหนดลักษณะของอุปกรณ์อินพุทนั้นๆมีทั้งหมด 4 ค่า

ค่าของ dwflag	ความหมาย
DISCL_BACKGROUND	กำหนดให้แอพพลิเคชั่นสามารถใช้อุปกรณ์อินพุทนั้นๆได้แม้ว่า แอพพลิเคชั่นจะอยู่ในโหมดแบล็คกราวด์
DISCL_FOREGROUND	แอพพลิเคชั่นไม่สามารถเรียกใช้อุปกรณ์อินพุทนั้นๆได้ในโหมดแบล็คกราวด์ เพราะอุปกรณ์อินพุทจะถูกปล่อยทันที เมื่อแอพพลิเคชั่นเปลี่ยนเป็นโหมดแบล็คกราวด์
DISCL_EXCLUSIVE	หลังจากทำการจองอุปกรณ์อินพุทนั้นๆแล้ว จะมีได้เพียงแอพพลิเคชั่นเดียว ที่ร้องขอ DISCL_EXCLUSIVE ส่วนแอพพลิเคชั่นอื่นจะต้องร้องขอการใช้งานอุปกรณ์อินพุทแบบ DISCL_NONEXCLUSIVE ได้
DISCL_NONEXCLUSIVE	แอพพลิเคชั่นสามารถใช้งานอุปกรณ์อินพุทร่วมกันได้

ตารางที่ 2.2 ค่า dwFlag ลักษณะการใช้งานของคีย์บอร์ด

สำหรับเกมที่ใช้โหมดแบบ Fullscreen จะมีการกำหนดลักษณะของอุปกรณ์อินพุทเป็นแบบ DISCL_BACKGROUND และ DISCL_NONEXCLUSIVE

ตัวอย่าง การใช้ฟังก์ชั่น SetCooperativeLevel( )

lpdikey → SetCooperativeLevel(hwnd,DISCL_BACKGROUND |

DISCL_NONEXCLUSIVE);

2.7.2.4 กำหนดรูแแบบของคียบอร์ด

เพื่อกำหนดว่า ข้อมูลที่จะรับเข้ามาจากคียบอร์ดมีรูปแบบอย่างไรด้วยฟังก์ชั่น SetDataFormat( ) โดยมีรูปแบบการเรียกใช้ดังนี้

HRESULT SetDataFormat(LPCDIDATAFORMAT lpdf);

- SetDataFormat มีพารามิเตอร์เพียงตัวเดียวคือ lpdf พารามิเตอร์นี้จะใช้โครงสร้างของ DIDATAFORMAT ตามที่ได้แสดงด้างล่างนี้

```
Typedef struct {
        DWORD       dwSize;      //ขนาดของโครงสร้างรูปแบบนี้หน่วยเป็นไบต์
        DWORD       dwObjSize;   //ขนาดของ DIOBJECTDATAFORMAT เป็นไบต์
        DWORD       dwFlags;     //กำหนดการรายงานผลว่าเป็นรูปแบบ Reletive หรือว่า
                                 Absolute
        DWORD       dwDataSize; //ขนาดของ packet ข้อมูล
        DWORD       dsNumObjs;   //จำนวนออปเจคกำหนดให้เป็นขนาดของอาร์เรย์
        LPDIOBJECTTDATAFORMAT  rgodf;  //พอยเตอร์ชี้ไปยังอาร์เรย์ออปเจค
                } DIDATAFORMAT, *LPDIDATAFORMAT;
```

แต่ DirectInput ได้กำหนดรูปแบบของข้อมูลวำหรับอุปกรณ์อินพุทมาตรฐานไว้เรียบร้อยแล้ว นั่นคือ

- c_dfDIKeyboard กำหนดรูปแบบข้อมูลสำหรับคีย์บอร์ดมาตรฐาน
- c_dfDIMouse กำหนดรูปแบบข้อมูลสำหรับเมาส์มาตรฐาน
- c_dfDIJoystick กำหนดรูปแบบข้อมูลสำหรับจอยสติ้กมาตรฐาน

ดังนั้นโปรแกรมเมอร์สามารถใช้รูปแบบข้อมูลที่ถูกกำหนดขึ้นไว้แล้ว ถ้าแอพพลิเคชั่นนั้นใช้อุปกรณ์อินพุทมาตรฐาน

ตัวอย่าง การใช้ฟังก์ชั่น SetDataFormat

lpdikey → SetDataFormat (&c_df DIKeyboard);

2.7.2.5 ทำการจองคีย์บอร์ดด้วยฟังก์ชั่น Acquire( ) มีรูปแบบการเรียกใช้ดังนี้

```
HRESULT          Acquire( );
```

ตัวอย่าง การใช้ฟังก์ชั่น Acquire( )

lpdikey → Acquire( );

2.7.2.6 ทำการอ่านข้อมูลจากคีย์บอร์ด

สำหรับลักษณะข้อมูล ของคีย์บอร์ดที่ถูกกำหนดโดย DirectInput มีลักษณะเป็น state data นั่นหมายความว่าเมื่ออ่านข้อมูลขึ้นมาจากคีย์บอร์ด ข้อมูลที่ได้จะเป็น

สถานะปัจจุบันของคีย์บอร์ด สำหรับการอ่านข้อมูลจากคีย์บอร์ดจะใช้ฟังก์ชั่น GetDeviceState( ) ฟังก์ชั่นนี้จะอ่านค่าสถานะปัจจุบันของคียบอร์ดขึ้นมา หรือสิ่งที่คีย์บอร์ดได้ทำครั้งสุดท้ายใน input loop โดยมีรูปแบบการเรียกใช้ฟังก์ชั่นดังนี้

```
HRESULT GetDeviceState (
        DWORD cbData,
```

LPVOID lpvData);

ตัวอย่าง การใช้ฟังก์ชั่น GetDeviceState( )

UCHAR keystate[256];

Lpdikey → GetDeviceState(256,keystate);

DirectInput จะมีค่าคงที่ของแต่ละคีย์บนคีย์บอร์ดเพื่อให้โปรแกรมเมอร์สามารถตรวจสอบสถานะของคีย์บนคีย์บอร์ดได้ว่าเป็นอย่างไร โดยค่าคงที่นี้จะขึ้นต้นด้วย DIK_ แล้วตามด้วยค่าคีย์ตามตารางที่ 2.3

ค่า	ความหมาย
DIK_ESCAPE	ปุ่ม ESC
DIK_0 – DIK_9	หมายเลข 0-9
DIK_A – DIK_Z	A ถึง Z
DIK_RETURN	ปุ่ม Enter
DIK_LCONTROL	ปุ่ม Ctrl ทางซ้ายมือ
DIK_RCONTROL	ปุ่ม Ctrl ทางขวามือ
DIK_SPACE	ปุ่ม Spacebar
DIK_F1- DIK_F12	ปุ่ม F1 ถึง F12
DIK_UP	ปุ่มลูกศรชี้ขึ้น
DIK_DOWN	ปุ่มลูกศรชี้ลง
DIK_LEFT	ปุ่มลูกศรชี้ไปทางซ้าย
DIK_RIGHT	ปุ่มลูกศรชี้ไปทางขวา
DIK_PRIOR	ปุ่ม PageUp
DIK_NEXT	ปุ่ม PageDown

ตารางที่ 2.3 ค่าคงที่ของแต่ละคีย์บนคีย์บอร์ด

หลังจากทำการอ่านค่าสถานะของคีย์บอร์ดแล้ว จะสามารถตรวจสอบได้ว่าคีย์ที่สนใจมีสถานะเป็นอย่างไรด้วยการตรวจสอบค่าบิตของคีย์นั้นๆ ถ้าคีย์ถูกกดค่าบิต ox80 จะมีค่าเป็น 1 แต่ถ้าคีย์นั้นๆไม่ถูกกด หรือคีย์นั้นๆถูกปล่อยไปแล้วค่าบิต ox80 จะมีค่าเป็น 0

ตัวอย่าง การตรวจสอบค่าของคีย์

int ship_x = 100 , ship_y = 100;

UCHARkeystate[256];

Lpdikey → GetDeviceState(256,keystate)

If(keystate[DIK_RIGHT] & ox80) ship_x++;

If(keystate[DIK_DOWN] & ox80) ship_y++;

จากตัวอย่างเป็นการตรวจสอบว่าคีย์ลูกศรชี้ลง หรือคีย์ลูกศรชี้ไปด้านขวาถูกกดหรือไม่

บทที่ 3
การออกแบบและการพัฒนาโปรแกรม

ขั้นตอนทั้งหมดในงานวิจัยจะเกิดขึ้นไม่ได้หากขาดการออกแบบการวางแผนที่ดี และมีเป้าหมายที่ชัดเจน การดำเนินงานทั้งหมดจะถูกกระทำภายใต้แผนที่วางเอาไว้อย่างรัดกุม เพื่อความสำเร็จที่วางเอาไว้ตามเป้าหมายในแต่ละช่วง แผนงานทั้งหมดมีดังนี้

1. ขั้นตอนการออกแบบ ซึ่งอาจแบ่งย่อยได้ดังนี้

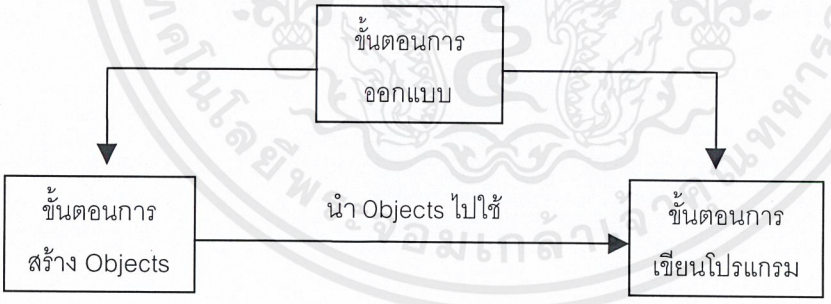
- การออกแบบระบบเกม
- การออกแบบและสร้างออปเจคในเกม
- การออกแบบอินเตอร์เฟซที่ใช้ติดต่อกับผู้เล่น

การสร้างรถ การสร้างฉาก ที่ใช้ในเกม ขั้นตอนนี้ใช้ระยะเวลาไม่มากนัก โดยทำการสร้างรถ ฉากเก็บไว้เป็นแหล่งข้อมูลเพื่อใช้ในเกม เมื่อจะต้องใช้ก็ไปดึงมาจากแหล่งข้อมูล ทำให้เร็วขึ้น

2. ขั้นตอนการเขียนโปรแกรม

ขั้นตอนนี้ใช้เวลานานที่สุดในตลอดระยะเวลาทำการวิจัย เพราะจำเป็นต้องศึกษารูปแบบวิธีการเขียนโปรแกรมที่เป็นลักษณะ real time และการเขียนโปรแกรมเชิงวัตถุ เพื่อให้ได้โปรแกรมที่เข้าใจง่าย มีขนาดไม่ใหญ่จนเกินไปและสามารถทำงานได้ถูกต้อง

สามารถสรุปขั้นตอนการวางแผนเป็นแผนภาพได้ดังรูปที่ 2.1



รูปที่3.1 ขั้นตอนการวางแผน

3.1 ขั้นตอนการออกแบบ

ขั้นตอนการออกแบบมีรายละเอียดดังนี้

3.1.1 การออกแบบระบบเกม

การออกแบบระบบเกมเป็นส่วนที่สำคัญที่สุดในขั้นตอนการออกแบบทั้งหมด เพราะระบบเกม คือ ทุกสิ่งทุกอย่างตั้งแต่แนวคิดของเกม การดำเนินเรื่องของเกมจนถึงการดำเนินเกมโดย การออกแบบมีรายละเอียดต่าง ๆ ดังนี้

การออกแบบทั่วไป

การออกแบบการรับข้อมูลต่าง ๆ

3.1.1.1 การออกแบบทั่วไป

การออกแบบทั่วไปเป็นการออกแบบที่เป็นรูปลักษณ์ของเกม โดยมีสิ่งต่างๆ ดังนี้ ตัวละครในเกม จะแบ่งเป็น 2 ลักษณะใหญ่ ๆ คือ สิ่งที่เคลื่อนไหวได้ กับสิ่งที่เคลื่อนไหวไม่ได้ ซึ่ง สิ่งที่เคลื่อนไหวได้ ก็คือ รถ ส่วนสิ่งที่เคลื่อนไหวไม่ได้ ก็คือ ตึก อาคาร ต้นไม้ เกมมีลักษณะเป็นเกมแนว simulation ประเภท Racing แบบ Real time คือ สามารถ สั่งการ และดำเนินการได้เลยโดยไม่ต้องรอ มุมมองที่ใช้ สามารถเปลี่ยนได้ เป็น 2 ลักษณะ แล้วแต่ความถนัดของผู้เล่น คือ มุมมองบุคคลที่ 1 (มุมมองจากในตัวรถ) และมุมมองบุคคลที่ 3 (มุมมองจากด้านท้ายของตัวรถ)

3.1.1.2 การออกแบบการรับข้อมูลต่างๆ

เป็นการออกแบบ คำสั่งต่างๆ ที่จะใช้ในการบังคับ ควมคุมรถ ซึ่งเป็นตัวละครในเกม และ ออกแบบรูปแบบการรับข้อมูล การสั่งการทั่วไป โดยใช้คีย์บอร์ด กดแป้นพิมพ์คีย์ต่าง ๆ เพื่อควบคุม รถ แบบ real time เกมแข่งรถในท้องตลาดในปัจจุบันโดยทั่วไปก็จะมีการควบคุมรถหลัก ๆ คือ เร่งความเร็ว ลดความเร็ว เลี้ยวซ้าย เลี้ยวขวา เพิ่มเกียร์ ลดเกียร์ ดังนั้นเกมที่ทำขึ้นมาจึงมีการควบคุม หลัก ๆ คล้ายกับเกมแข่งรถทั่ว ๆ ไป คือ จะมีการควบคุม ดังนี้

- เร่งความเร็ว
- เบรก
- เลี้ยวซ้าย
- เลี้ยวขวา
- เพิ่มเกียร์
- ลดเกียร์
- เปลี่ยนมุมมอง

โดยที่แต่ละคำสั่งมีรายละเอียดปลีกย่อยดังนี้

- **เร่งความเร็ว (Accelerator)**

แป้นพิมพ์ Z จะใช้แทนปุ่มเร่งความเร็ว ปุ่มนี้จะใช้เมื่อต้องการที่จะเพิ่มความเร็วของรถให้สูงขึ้น เมื่อกดปุ่ม Z ความเร็วจะค่อยๆสูงขึ้น แต่เมื่อปล่อยปุ่ม Z ความเร็วจะค่อยๆลดลงเรื่อยๆ ซึ่งอัตราในการเพิ่มความเร็ว จะขึ้นอยู่กับตำแหน่งเกียร์ และความเร็วรอบ(RPM) ในขณะนั้น รวมถึงค่าอื่น ๆ ที่จำเป็นในการคำนวณ เช่น ค่าแรงม้าของเครื่องยนต์

- **เบรก (Break)**

แป้นพิมพ์ X จะใช้แทนปุ่มเบรก ปุ่มนี้จะใช้เมื่อต้องการที่จะลดความเร็วของรถหรือต้องการที่จะหยุดรถ ซึ่งใช้ค่าสัมประสิทธิ์ความเสียดทานของถนน ที่กำหนดไว้ มาช่วยคำนวณด้วย

- **เลี้ยวซ้าย (Turn Left)**

แป้นพิมพ์ ← จะใช้แทนปุ่มเลี้ยวซ้าย ปุ่มนี้จะใช้เมื่อต้องการจะเลี้ยวรถไปทางด้านซ้าย ทำให้มุมมองและ ทิศทางการเคลื่อนที่ของรถ เปลี่ยนไป ความเร็วรถก็มีผลกับการเลี้ยว เช่น ถ้ารถจอดนิ่งอยู่กับที่ การเลี้ยวก็จะไม่เกิดผลใด ๆ

- **เลี้ยวขวา (Turn Right)**

แป้นพิมพ์ → จะใช้แทนปุ่มเลี้ยวขวา ปุ่มนี้จะใช้เมื่อต้องการจะเลี้ยวรถไปทางด้านขวา ทำให้มุมมองและ ทิศทางการเคลื่อนที่ของรถ เปลี่ยนไป ความเร็วรถก็มีผลกับการเลี้ยว เช่น ถ้ารถจอดนิ่งอยู่กับที่ การเลี้ยวก็จะไม่เกิดผลใด ๆ

- **เพิ่มเกียร์ (Shift Up)**

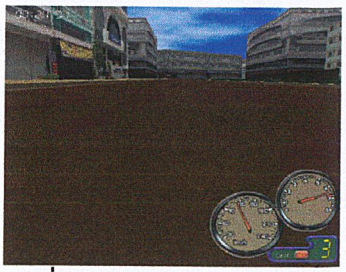
แป้นพิมพ์ ↑ จะใช้แทนปุ่มเพิ่มเกียร์ ปุ่มนี้จะใช้เมื่อต้องการจะเพิ่มเกียร์ให้สูงขึ้น 1 ตำแหน่ง

- **ลดเกียร์ (Shift Down)**

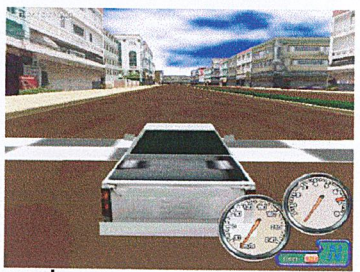
แป้นพิมพ์ ↓ จะใช้แทนปุ่มลดเกียร์ ปุ่มนี้จะใช้เมื่อต้องการจะลดเกียร์ให้ต่ำลง 1 ตำแหน่ง

- **เปลี่ยนมุมมอง (Change Viewport)**

แป้นพิมพ์ V จะใช้แทนปุ่มเปลี่ยนมุมมองของตัวรถ มุมมองของรถในเกมนี้จะมีอยู่ 2 มุมมองที่สามารถเลือกได้ คือ 1. มุมมองภายในตัวรถ มุมมองนี้จะไม่เห็นตัวรถของผู้เล่น แต่จะเสมือนว่าเราเป็นผู้ขับอยู่จริงๆ 2. มุมมองด้านท้ายของตัวรถ มุมมองนี้จะเห็นตัวรถทั้งคันจากด้านท้ายของตัวรถ จะเห็นการเคลื่อนที่ของรถ และการเลี้ยวของตัวรถที่เราควบคุมอยู่

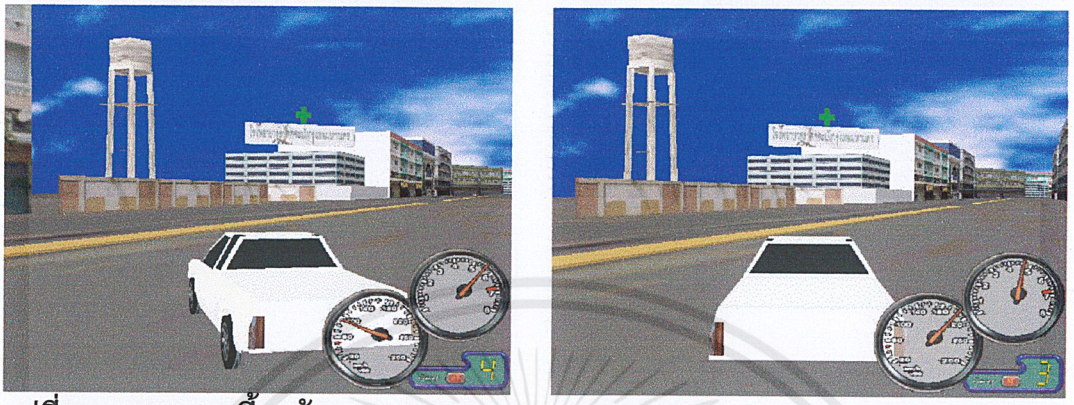


รูปที่3.2 มุมมองภายในตัวรถ

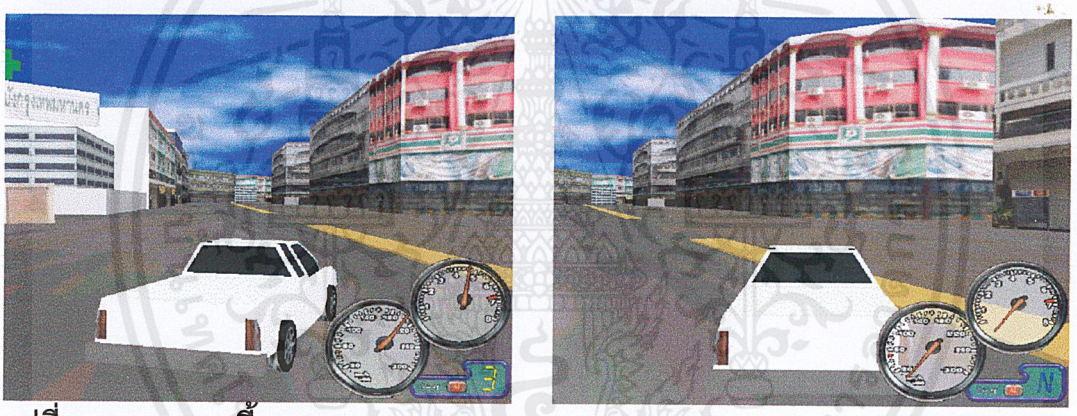


รูปที่3.3 มุมมองด้านท้ายของตัว

การควบคุมรถให้เลี้ยว เมื่อกดแป้นพิมพ์ควบคุมการเลี้ยว ตัวรถจะเบี่ยงไปตามทิศทางที่ควบคุมไปและฉากก็จะเปลี่ยนไปตามทิศทางของรถ และเมื่อปล่อยแป้นพิมพ์ ตัวรถจะค่อย ๆ คืนกลับมาสู่แนวตรง ดังรูป



รูปที่3.4 แสดงการเลี้ยวซ้าย



รูปที่3.5 แสดงการเลี้ยวขวา

3.1.2 การออกแบบและสร้างออปเจคในเกม

ออปเจคในเกมในที่นี้เราจะหมายถึง ตัวรถที่ใช้เล่น ตัวอาคารสิ่งก่อสร้าง เป็นต้น

3.1.2.1 ขั้นตอนการออกแบบรถ

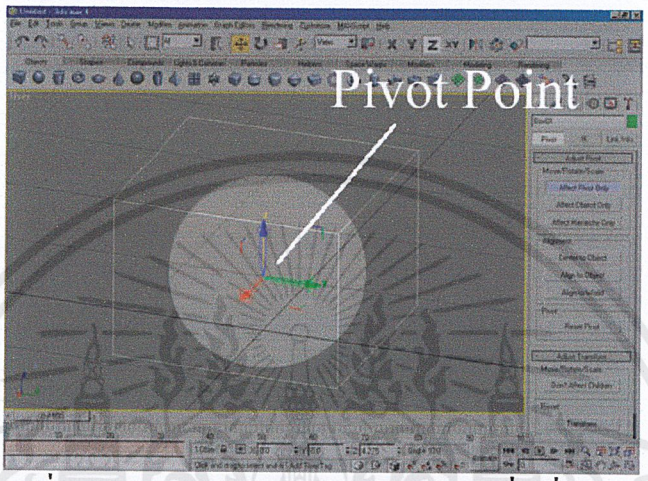
การออกแบบรถ เป็นขั้นตอนที่ใช้เวลาค่อนข้างมาก เนื่องจากเกมที่ทำนี้เป็นเกมแข่งรถ 3 มิติ ดังนั้น รถที่ออกแบบจึงต้องเป็นรถแบบ 3 มิติด้วย ซึ่งต้องใช้ความละเอียดสูงและต้องมองจากทุกมุมมองได้ การออกแบบจะออกแบบ โดยใช้โปรแกรม “3D studio Max 4” ซึ่งเป็นโปรแกรมที่นิยมใช้ในการออกแบบภาพ 3 มิติ กันอย่างแพร่หลาย มีขั้นตอนหลัก ๆ ดังนี้

หาแบบรถที่เราต้องการทำโดยอาจเป็นรถยนต์ รถกระบะ รถตู้ หรือแม้กระทั่งรถเมล์หรือรถบรรทุก

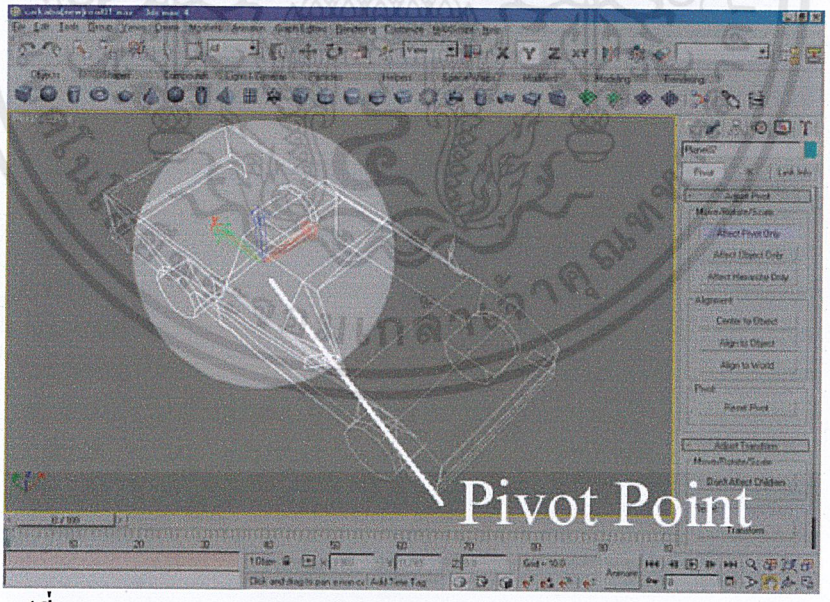
ศึกษาส่วนประกอบต่างๆของตัวรถที่เราจะสร้าง ซึ่งส่วนประกอบหลัก ๆ จะประกอบด้วย โครงรถ, ที่นั่ง, พวงมาลัย, กันชน, ประตู ฯลฯ

นำส่วนประกอบต่าง ๆ มาวาดบนกระดาษ เพื่อแยกส่วนสำคัญต่าง ๆ

นำส่วนประกอบที่ได้มาออกแบบโดยใช้โปรแกรม 3D studio Max 4
กำหนดจุด Pivot Point ให้กับตัวรถ โดยใช้โปรแกรม 3D studio Max 4 ซึ่งในการอ้าง Position X, Y, Z ของรถใน 3D World ก็หมายถึง จุด Pivot Point นั่นเอง และยังใช้เป็น
จุดหมุนของ Object เมื่อสั่งหมุน Object ด้วยคำสั่ง Rotate อีกด้วย ดังนั้น จึงกำหนด
จุด Pivot Point ไว้กึ่งกลางตัวรถ



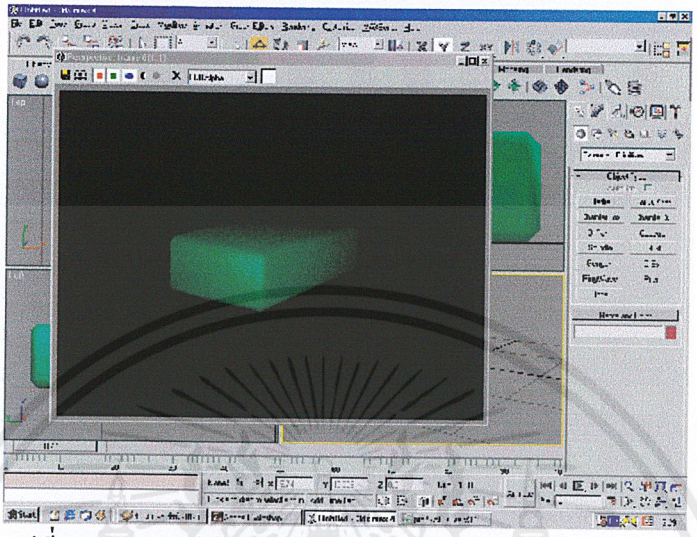
รูปที่3.6 จุด Pivot Point ของ Object รูปทรงสี่เหลี่ยม



รูปที่3.7 จุด Pivot Point ของ Object รถ

1) การสรางโค้รงแบบรถ

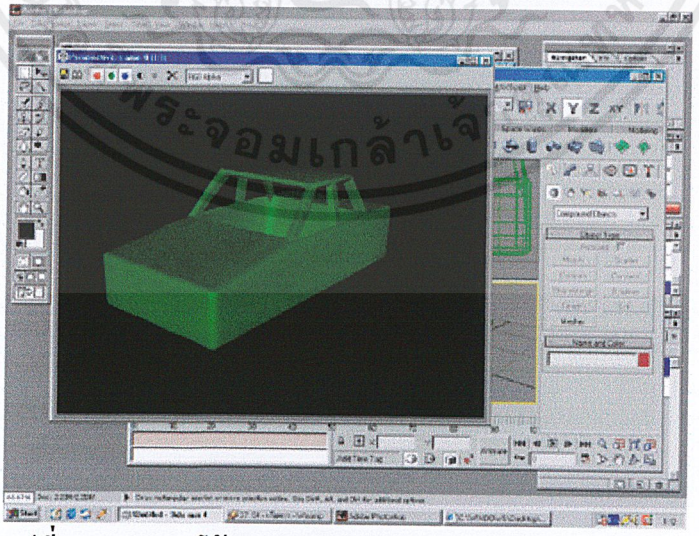
ขั้นตอนแรกในการสร้างตัวรถ จะต้องขึ้น โครงมาตรฐานของตัวรถ คือรูปทรงสี่เหลี่ยม (box) ตามขนาดของตัวรถก่อน ดังรูป



รูปที่3.8 การสรางโค้รงแบบรถ(1)

2) การสรางโค้รงแบบรถ

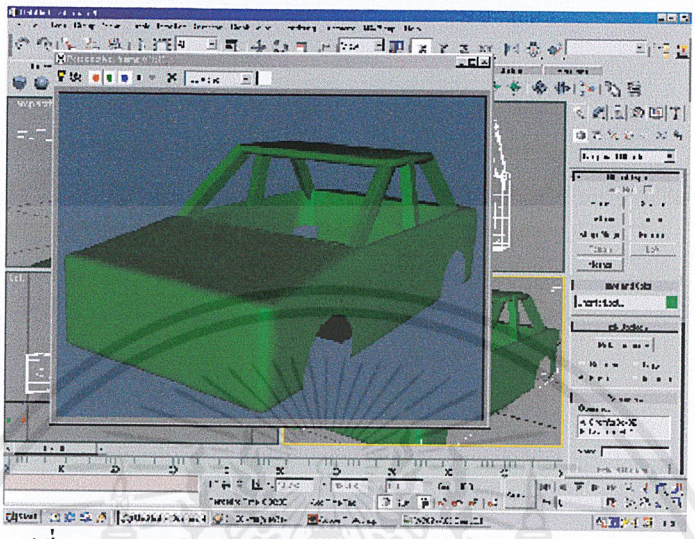
เพิ่มเติมหลังคารถ โดยใช้รูปทรงสี่เหลี่ยม มาประกอบกันเป็น โครงรถตามแบบที่มี (ให้เหมือนแบบหรือใกล้เคียงที่สุด) ในที่นี้จะเป็นรถยนต์ ถ้าเป็นรถกระบะหรือรถตู้ก็จะต้องเพิ่มเติมหลังคาตามแบบตัวรถนั้น ๆ



รูปที่3.9 การสรางโค้รงแบบรถ(2)

3) การสรางโค้รงแบบรถ

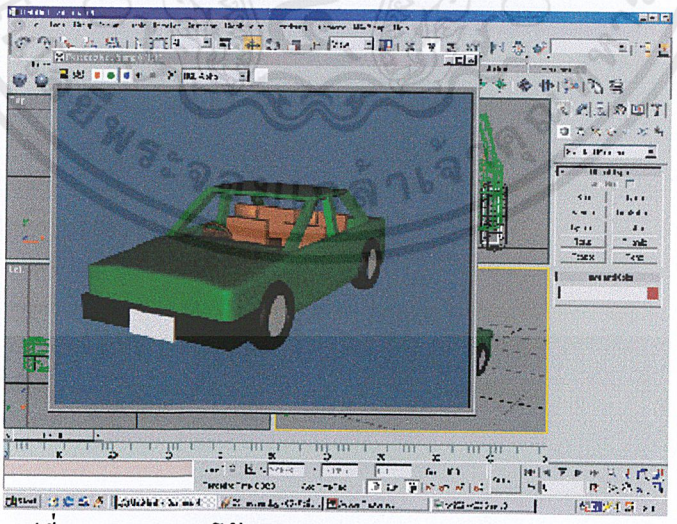
นำรูปทรงกลม มาทำการตัดกับตัวรถเพื่อเป็นช่องล้อรถ โดยใช้คำสั่งบูลีนจะได้ช่องล้อรถ ทำเหมือนกันทั้งสี่ล้อ



รูปที่3.10 การสรางโค้รงแบบรถ(3)

4) การสรางโค้รงแบบรถ

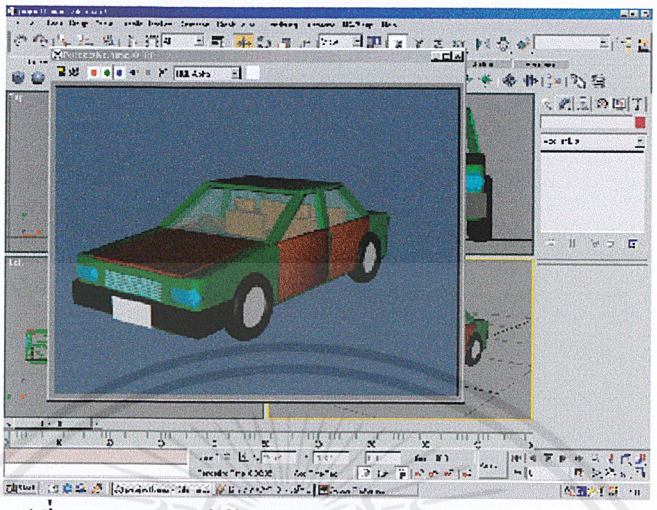
เพิ่มเติม ที่นั่ง, ล้อทั้ง4, กันชนหน้า, กันชนหลัง, พวงมาลัยรถ, ป้ายทะเบียนรถ (จะเริ่มเห็นเป็นรถ แต่ยังขาดรายละเอียดที่สมจริง)



รูปที่3.11 การสรางโค้รงแบบรถ(4)

5) การสรางโค้รงแบบรถ

เพิ่มเติมมิติต่าง ๆ ให้กับตัวรถ โดยการเพิ่มกระโปรงหน้ารถ, ประตู, ไฟหน้า ฯลฯ

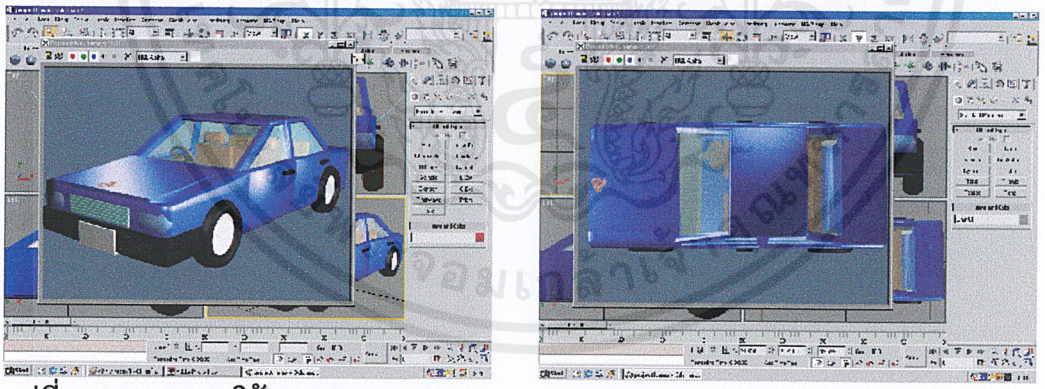


รูปที่3.12 การสรางโค้รงแบบรถ(5)

6) การสรางโค้รงแบบรถ

ปรับแต่งสีให้กับตัวรถและส่วนประกอบต่าง ๆ

เพิ่ม texture ให้กับพื้นผิววัตถุ ก็จะได้แบบโครงสร้างรถที่ไว้ใช้ในตัวเกม



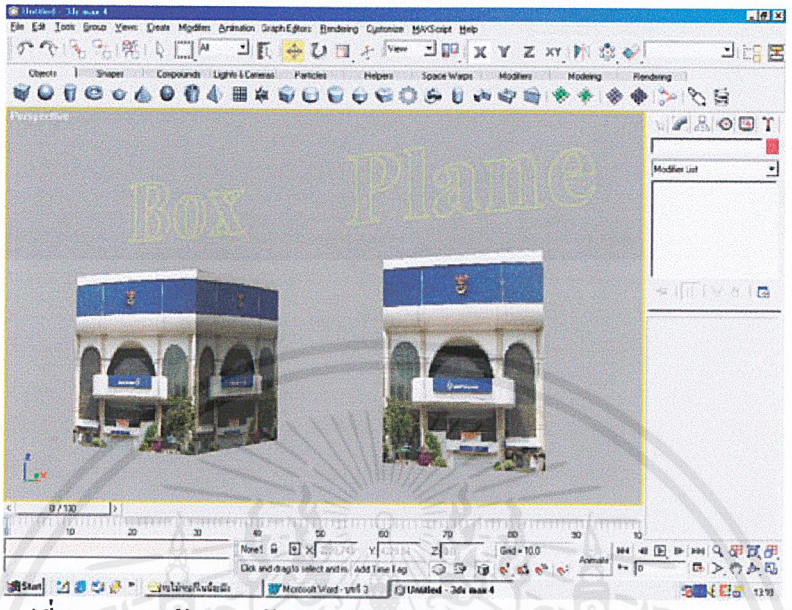
รูปที่3.13 การสรางโค้รงแบบรถ(6)

3.1.2.2 การออกแบบฉากแผนที่ที่ใช้ในเกม

ในส่วนของฉากแผนที่ที่ใช้ในเกมนั้น จะแสดงถึงสถานที่ต่าง ๆ ที่รถของเราจะเคลื่อนที่ผ่าน เช่น ตึก ทางเท้า ต้นไม้ ฯลฯ

เนื่องจากฉากในเกมนั้นมีขนาดที่ใหญ่มาก การที่จะใช้รูปทรงสี่เหลี่ยม (BOX) มาใช้แทนแต่ละตึกนั้น จะทำให้สิ้นเปลืองหน่วยความจำมาก ซึ่งมีผลทำให้ความเร็วในการแสดงผลของเกม

ลดลงไปด้วย ดังนั้นจึงใช้ผืนระนาบ (PLANE) มาแทนตัวตึก (โดยที่ 1 BOX จะประกอบไปด้วยกันทั้งหมด 6 ด้าน แต่ PLANE จะประกอบไปด้วยด้านเพียงด้านเดียว)



รูปที่3.14 การสร้างฉากด้วย BOX กับ PLANE

เมื่อสร้าง PLANE มาแล้ว ก็จึงนำ TEXTURE ของตึกที่เราทำมาจากโปรแกรม PHOTO SHOP มาวางบนระนาบ PLANE เราก็จะได้ตึกที่มีความสมจริงเหมือนกับที่เราใช้ BOX เช่นกัน

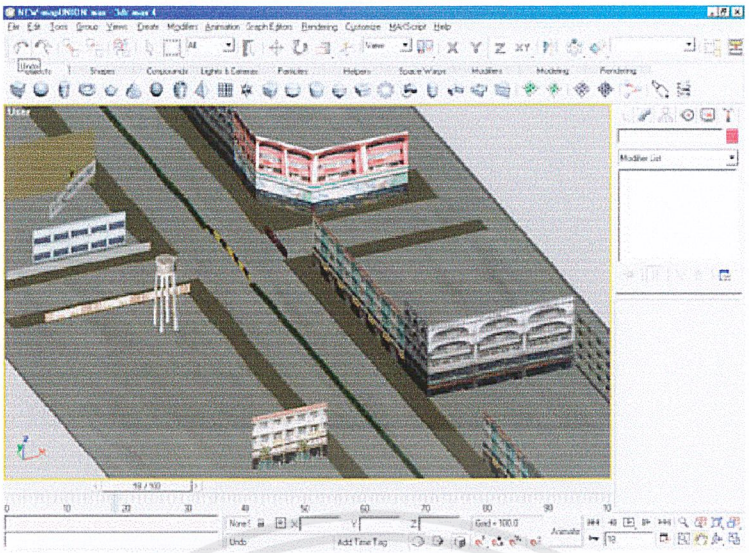
เมื่อได้ตึกที่สร้างจาก PLANE แล้วเราจึงนำ PLANE ที่ทำไว้ มาวางตามฉากแผนที่ที่เราต้องการ ซึ่งจะมีหลักการตัดสินใจว่าจะวาง PLANE ไว้ตำแหน่งไหนบ้าง โดยพิจารณาจากว่าผู้เล่นจะสามารถมองเห็นด้านใดของตึกได้บ้าง

ถ้าตึกที่จะวางนั้นอยู่ระหว่างตึกอื่น ๆ ผู้เล่นจะไม่สามารถมองเห็นด้านข้างของตัวตึกได้ ดังนั้นตึกนี้ใช้ PLANE ที่ด้านหน้าตึกเพียงที่เดียวก็เพียงพอ

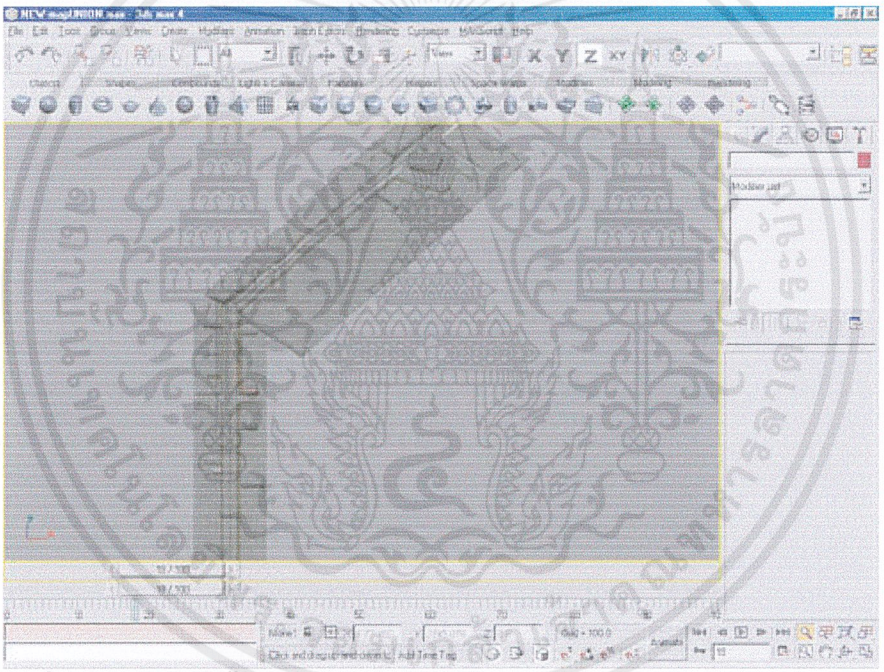
ตึกที่ตั้งอยู่ที่มุมของตึก ผู้เล่นก็จะสามารถมองเห็นได้ 2 ด้าน คือ ด้านหน้า และด้านข้างของตัวตึก ดังนั้นตึกนี้ก็จะต้องประกอบไปด้วย 2 PLANE

ตึกที่ตั้งอยู่ตึกเดียวไม่มีตึกอื่น ๆ อยู่ด้านข้าง ตามปกติก็น่าที่จะใช้เพียง 2 PLANE เท่านั้น แต่ก็ต้องคิดเผื่อกรณีที่รถของผู้เล่นย้อนกลับมาจากฝั่งตรงข้าม ดังนั้นจึงจำเป็นที่จะต้องเพิ่ม PLANE ของอีกด้านหนึ่งขึ้นมาด้วย

สรุปแล้วจะเห็นได้ว่าถึงแม้ตึกที่สร้างนั้น จะมองเห็นได้รอบตัวตึกก็ยังคงใช้ PLANE เพียง 4 ด้านเท่านั้น แทนที่จะต้องใช้ BOX ซึ่งด้านบนกับด้านล่างของตัวตึกผู้เล่นไม่สามารถมองเห็นได้ ดังนั้นจึงเป็นการลดขนาดของฉากแผนที่ลงได้อีกมากทีเดียว



รูปที่3.15 ฉากที่สร้างโดยโปรแกรม 3ds max



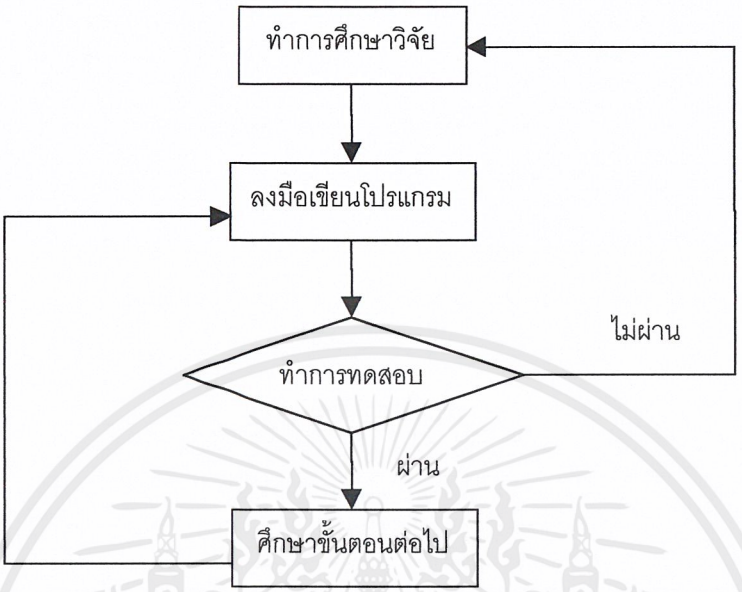
รูปที่3.16 แสดงฉากทั้งหมดในเกม

3.2 ขั้นตอนการเขียนโปรแกรม

ขั้นตอนนี้ใช้เวลานานที่สุดในตลอดระยะเวลาการทำวิจัย เพราะจำเป็นต้องศึกษารูปแบบวิธีการเขียนโปรแกรมที่เป็นลักษณะ Real time และการเขียนโปรแกรมแบบเชิงวัตถุ เพื่อให้ได้โปรแกรมที่ทำงานได้ถูกต้อง เข้าใจง่าย และมีขนาดกะทัดรัด

ส่วนใหญ่แล้วขั้นตอนต่าง ๆ จะมีการทำงานแบบเส้นตรง หมายถึงเมื่อทำเสร็จงานหนึ่งก็จะทำงานต่อไปโดยไม่มีการวนมาที่งานนั้นใหม่ แต่ขั้นตอนการเขียนโปรแกรมมีการทำงานที่ต้องวน

มาศึกษาเรื่อย ๆ และทำการเขียนโปรแกรม หากยังเขียนไม่เสร็จ ก็จะต้องทำการศึกษาเพิ่มเติมใหม่ ดังรูปที่ 3.17



รูปที่3.17 ขั้นตอนการเขียนโปรแกรม

โปรแกรมจะถูกแบ่งเป็นส่วนย่อย ๆ หลายส่วนดังนี้

ส่วนการแสดงผล

ส่วนการติดต่ออุปกรณ์เพื่อรับอินพุท

ส่วนของการติดต่อการ์ดเสียงและการแสดงเสียง

การกำหนดตำแหน่งของ Object ใน 3D World Space

การกำหนดขอบเขตการชนของฉาก

หลักการตรวจสอบการชน

เหตุผลหลักในการแบ่งงานออกเป็นหลาย ๆ ส่วน เพื่อต้องการจัดแบ่งหน้าที่การทำงานให้เป็นประเภทเดียวกัน ทำให้ง่ายต่อการตรวจสอบ แก้ไข หรือเพิ่มเติม หลักการ แนวความคิด และรูปแบบของข้อมูลในแต่ละส่วนการทำงาน

3.2.1 ส่วนการแสดงผล

ในส่วนการแสดงผลนี้ จะรวมฟังก์ชัน ที่ติดต่อกับการ์ดจอ, การกำหนดรูปแบบการแสดงผล, ความละเอียด, จำนวนสี, จำนวน backbuffer, การสร้าง device สำหรับวาดออกหน้าจอ รวมถึงการ render object 3 มิติด้วย

สำหรับการแสดงผล object 3 มิติ ได้แยกออกมาเป็น class ไว้ใน file OmeshObject.h และ OmeshObject.cpp โดยจะประกอบไปด้วยตัวแปรที่จำเป็น ในการ render object 3 มิติ ดังนี้

```
class OMeshObject
{
private :
        FLOAT x, y, z, Radius;
        // Our mesh object in system
        D3DMATERIAL8*           m_pMeshMaterials; // Materials for our mesh
        LPDIRECT3DTEXTURE8*     m_pMeshTextures;// Textures for our mesh
        DWORD                   m_dwNumMaterials;// Number of mesh materials
        LPD3DXBUFFER            pD3DXMtrlBuffer;
D3DXMATERIAL*           d3dxMaterials;
        LPDIRECT3DDEVICE8       m_pd3dDevice;
        LPD3DXMESH              m_pMesh;

public :
        OMeshObject(void);
        ~OMeshObject(void);

        HRESULT Create(LPDIRECT3DDEVICE8 g_pd3dDevice,
                TCHAR* strFileName);
        HRESULT Create(LPDIRECT3DDEVICE8 g_pd3dDevice,
                TCHAR* strFileName,
                TCHAR* strTextureFileName);
        VOID SetupWorldSpace(FLOAT _x, FLOAT _y, FLOAT _z);
        HRESULT Render(void);
        LPD3DXMESH GetSysMemMesh() { return m_pMesh; }
        VOID Destroy(void);
};
```

ภายใน class ประกอบไปด้วย ตัวแปรที่จำเป็นสำหรับ object 3 มิติ ได้แก่ ตำแหน่ง x, y, z ตัวแปรสำหรับจัดการกับ object 3 มิติที่โหลดเข้ามา จัดการกับ textures และ device ที่จะใช้ในการ render object นี้ ส่วน method ก็จะประกอบไปด้วย method ที่จะใช้ create object โดยมี overload method ในกรณีที่จะใช้ texture อื่นที่กำหนดมา map ลงบน object นี้ และ method ในการกำหนดตำแหน่งของ object และสุดท้าย method render ที่จะใช้ในการวาด object

ดังนั้น object ที่ไม่มีการเคลื่อนไหว เช่น ถนน ตึก ต้นไม้ ฯลฯ จะประกาศให้เป็น instance ของ class นี้ ส่วนรถที่เป็นตัวละคร ที่เราสามารถบังคับได้ จะแยกออกมาเป็น class ใหม่ โดยสืบทอด (inherite) มาจาก class OmeshObject นี้ เพราะเป็น object 3 มิติเหมือนกัน มีการใช้ตัวแปรเช่นเดียวกัน แต่มีส่วนที่เพิ่มเติมขึ้นมาก็คือ การควบคุมตัวรถต่าง การเร่ง เบรก เลี้ยวซ้าย เลี้ยวขวา ก็จะต้องมีตัวแปรและ method เพิ่มขึ้นมาเพื่อที่จะจัดการในการทำงานส่วนนี้

ส่วนการสร้าง object ของ direct3D version 8.0 และ สร้าง device เพื่อใช้ในการแสดงผล รวมถึงการติดต่อกับการ์ดแสดงผล เพื่อเลือกโหมดแสดงผลที่ดีที่สุดมาใช้ และกำหนดค่าเริ่มต้นให้กับโปรแกรม รวมอยู่ใน method InitD3D ดังนี้

```
HRESULT InitD3D( HWND hWnd )
{
// Create the D3D object.
        if( NULL == ( g_pD3D = Direct3DCreate8( D3D_SDK_VERSION ) ) )
                return E_FAIL;

        // Get the current desktop display mode, so we can set up a back
        // buffer of the same format
        D3DDISPLAYMODE d3ddm;
        if( FAILED( g_pD3D->GetAdapterDisplayMode( D3DADAPTER_DEFAULT,
&d3ddm ) ) )
                return E_FAIL;

const D3DFORMAT FormatFullscreenArray[] =
        {
                // X == non used bit, A == alpha
                D3DFMT_R5G6B5,      //16 bits
                D3DFMT_X1R5G5B5,  //16 bits
```

```
            D3DFMT_A1R5G5B5,  //16 bits
            D3DFMT_X8R8G8B8,  //32 bits
            D3DFMT_A8R8G8B8,  //32 bits
        };
        const INT NumFullscreenFormat = sizeof(FormatFullscreenArray) / sizeof
(FormatFullscreenArray[0]);

        INT iFormat;
        for (iFormat=0;iFormat<NumFullscreenFormat;iFormat++)
        {
            if ( SUCCEEDED(g_pD3D->CheckDeviceType(
                D3DADAPTER_DEFAULT, D3DDEVTYPE_HAL,
                FormatFullscreenArray[iFormat],
                FormatFullscreenArray[iFormat], FALSE )))
            {
                g_d3dFormatFullscreen = FormatFullscreenArray[iFormat];
                break;
            }
        }

        const D3DFORMAT FormatTextureArray[] =
        {
            D3DFMT_A1R5G5B5,
            D3DFMT_A4R4G4B4,
            D3DFMT_A8R8G8B8,
        };
        const INT NumTextureFormat = sizeof(FormatTextureArray) /
sizeof(FormatTextureArray[0]);

        for (iFormat=0;iFormat<NumTextureFormat;iFormat++)
        {
            if ( SUCCEEDED(g_pD3D->CheckDeviceFormat(
```

```
                    D3DADAPTER_DEFAULT, D3DDEVTYPE_HAL,
                    d3ddm.Format, 0, D3DRTYPE_TEXTURE,
                    FormatTextureArray[iFormat] )))
            {
                    g_d3dFormatTexture = FormatTextureArray[iFormat];
                    break;
            }
    }

// Set up the structure used to create the D3DDevice. Since we are now
    // using more complex geometry, we will create a device with a zbuffer.
    D3DPRESENT_PARAMETERS d3dpp;
    ZeroMemory( &d3dpp, sizeof(d3dpp) );
    d3dpp.Windowed = FALSE;
    d3dpp.BackBufferWidth = 640;
    d3dpp.BackBufferHeight  = 480;
    d3dpp.BackBufferCount = 1;
    d3dpp.BackBufferFormat = D3DFMT_R5G6B5;
    d3dpp.SwapEffect = D3DSWAPEFFECT_DISCARD;
    d3dpp.EnableAutoDepthStencil = true;
    d3dpp.AutoDepthStencilFormat = D3DFMT_D16;
    d3dpp.FullScreen_RefreshRateInHz = D3DPRESENT_RATE_DEFAULT;
    d3dpp.FullScreen_PresentationInterval = D3DPRESENT_INTERVAL_TWO;

    // Create the D3DDevice
    if( FAILED( g_pD3D->CreateDevice( D3DADAPTER_DEFAULT,
D3DDEVTYPE_HAL, hWnd,
                    D3DCREATE_SOFTWARE_VERTEXPROCESSING,
                    &d3dpp, &g_pd3dDevice ) ) )
    {
            return E_FAIL;
    }
```

```
        g_pd3dDevice->SetTextureStageState( 0 ,D3DTSS_COLORARG1,
D3DTA_TEXTURE );
        g_pd3dDevice->SetTextureStageState( 0 ,D3DTSS_COLORARG2, D3DTA_DIFFUSE
);
        g_pd3dDevice->SetTextureStageState( 0 ,D3DTSS_COLOROP,
D3DTOP_MODULATE );
        g_pd3dDevice->SetTextureStageState( 0 ,D3DTSS_ALPHAARG1, D3DTA_TEXTURE
);
        g_pd3dDevice->SetTextureStageState( 0 ,D3DTSS_ALPHAARG2, D3DTA_DIFFUSE
);
        g_pd3dDevice->SetTextureStageState( 0 ,D3DTSS_ALPHAOP,
D3DTOP_MODULATE );
        g_pd3dDevice->SetTextureStageState( 0 ,D3DTSS_MINFILTER, D3DTEXF_LINEAR
);
        g_pd3dDevice->SetTextureStageState( 0 ,D3DTSS_MAGFILTER,
D3DTEXF_LINEAR );

        //Set Light
        D3DXVECTOR3 vecDir;
        D3DLIGHT8     light;
        ZeroMemory( &light, sizeof(D3DLIGHT8) );
        light.Type = D3DLIGHT_DIRECTIONAL;
        light.Diffuse.r = 0.05f;
        light.Diffuse.g = 0.05f;
        light.Diffuse.b = 0.05f;
        vecDir = D3DXVECTOR3(0.0f, 0.0f, 0.0f) - D3DXVECTOR3(0.0f,0.5f,0.8f);
        D3DXVec3Normalize((D3DXVECTOR3*)&light.Direction, &vecDir );
        light.Range = 1000.0f;

        // Turn on the zbuffer
        g_pd3dDevice->SetRenderState( D3DRS_ZENABLE, TRUE );
```

```
        g_pd3dDevice->SetRenderState( D3DRS_LIGHTING, TRUE );
        g_pd3dDevice->SetRenderState( D3DRS_AMBIENT, 0x00ffffff );
        g_pd3dDevice->LightEnable(0, TRUE );
g_pd3dDevice->SetLight(0, &light );

        return S_OK;
}
```

โดยเมื่อตรวจสอบการ์ดแสดงผลเรียบร้อยแล้ว จะทำการสร้าง device โดยค่าต่าง ๆ เช่น ความละเอียดโหมดการแสดงผล, จำนวนสี, จำนวน backbuffer, เป็น windowed โหมด หรือ fullscreen โหมด จะถูกกำหนดอยู่ในตัวแปร d3dpp ประเภท D3DPRESENT_PARAMETERS จาก นั้นเมื่อได้ device แล้ว จะทำการกำหนดค่าเริ่มต้นต่าง ๆ เช่น สร้างแสง, กำหนดสถานะให้ textures ที่จะใช้, กำหนดให้มีการใช้ z-buffer คือ มีระยะความลึก (แกน Z) เพื่อให้พร้อมที่จะวาด object 3 มิติต่อไป

3.2.2 ส่วนการติดต่ออุปกรณ์เพื่อรับอินพุท

ในส่วนนี้จะรวมฟังก์ชันที่ทำการติดต่อกับอุปกรณ์อินพุท โดยผ่านทาง DirectInput โปรแกรมเกมที่พัฒนาขึ้นมานี้จะใช้คีย์บอร์ดในการรับอินพุทเป็นหลัก

ฟังก์ชั่นการทำงานในส่วนการติดต่ออุปกรณ์เพื่อรับอินพุท

InitDI

ProcessKBInput

FreeDirectInput

หลักการและการทำงานของฟังก์ชั่น

InitDI

ฟังก์ชั่นนี้จะทำการติดต่อกับ DirectInput เพื่อเตรียมอุปกรณ์รับอินพุทให้พร้อมใช้งาน

```
HRESULT InitDI( HWND hWnd ,LPDIRECT3DDEVICE8 g_pd3dDevice)
{
HRESULT hr;

if( FAILED(hr = DirectInput8Create( GetModuleHandle(NULL), DIRECTINPUT_VERSION,
                IID_IDirectInput8, (VOID**)&g_pDI, NULL )))
```

```
{
return hr;
}

if( FAILED(g_pDI->CreateDevice( GUID_SysKeyboard , &g_pKeyboard, NULL )))
return hr;


if( FAILED( hr = g_pKeyboard->SetDataFormat( &c_dfDIKeyboard ) ) )
return hr;

hr = g_pKeyboard->SetCooperativeLevel( hWnd, DISCL_FOREGROUND |
                                       DISCL_NONEXCLUSIVE);//DISCL_EXCLUSIVE );
if( FAILED(hr) )
return hr;

DIPROPDWORD dipdw;
dipdw.diph.dwSize          = sizeof(DIPROPDWORD);
dipdw.diph.dwHeaderSize    = sizeof(DIPROPHEADER);
dipdw.diph.dwObj     = 0;
dipdw.diph.dwHow     = DIPH_DEVICE;

if( FAILED( hr = g_pKeyboard->SetProperty( DIPROP_BUFFERSIZE, &dipdw.diph ) ) )
return hr;

return S_OK;
}
```

ProcessKBInput

ฟังก์ชันนี้ทำหน้าที่รับค่าจาก Keyboard และตรวจสอบ Input ที่รับเข้ามา แล้วประมวลผลสถานะการทำงาน เช่น User กดปุ่มเร่งความเร็วรถ ค่าของตัวแปร Accelerator ใน structure Controls จะ set เป็น TRUE หากปล่อยปุ่มเมื่อไร ค่าในตัวแปร Accelerator จะถูก set เป็น FALSE

ทันที ดังนั้น ในแต่ละรอบของ loop ประมวลผล สามารถใช้การควบคุมหลาย ๆ อย่างพร้อมกันได้ เช่น เร่งความเร็วไปพร้อม ๆ กับการเลี้ยวซ้าย

```
VOID WINAPI ProcessKBInput()
{
#define KEYDOWN(name, key) (name[key] & 0x80)

char    buffer[256];
HRESULT  hr;

g_pKeyboard->Acquire();
hr = g_pKeyboard->GetDeviceState(sizeof(buffer),(LPVOID)&buffer);
if FAILED(hr)
{
// If it failed, the device has probably been lost.
// Check for
// and attempt to reacquire it here.
return;
}

Controls.Accelerate      = buffer[DIK_Z] && 0x80;
Controls.TurnLeft                = buffer[DIK_LEFT] && 0x80;
Controls.TurnRight       = buffer[DIK_RIGHT] && 0x80;
Controls.Break           = buffer[DIK_X] && 0x80;
Controls.ShiftUp                 = buffer[DIK_UP] && 0x80;
Controls.ShiftDown       = buffer[DIK_DOWN] && 0x80;

// Thrust or stop the car
}
```

FreeDirectInput

เมื่อต้องการยกเลิกการติดต่อกับ Direct Input จะต้องทำการลบออปเจคของ DirectInput ทิ้ง เพื่อคืนค่าหน่วยความจำกลับสู่ระบบ

```
VOID FreeDirectInput()
{
// Unacquire the device one last time just in case
// the app tried to exit while the device is still acquired.
if( g_pKeyboard )
g_pKeyboard->Unacquire();

// Release any DirectInput objects.
g_pKeyboard->Release();
g_pKeyboard = NULL;
g_pDI->Release();
g_pDI = NULL;
}
```

3.2.3 ส่วนของการติดต่อการ์ดเสียงและการแสดงเสียง

ในส่วนนี้เราจะติดต่อกับการ์ดเสียงโดยผ่านทาง Direct Sound แต่เนื่องจากการใช้ Direct Sound มีข้อจำกัดในเรื่องของการเลือกไฟล์เสียง เพราะ Direct Sound จะเล่นได้เฉพาะไฟล์ WAV ดังนั้นเราจึงทำการสร้างคลาสซึ่งใช้ติดต่อกับการ์ดเสียง โดยพัฒนามาจากคลาสที่ใช้ใน Direct Sound ชื่อว่าคลาส CAudio ซึ่งสามารถเล่นไฟล์เสียงได้ทุกนามสกุลทำให้ง่ายต่อการเลือกใช้

ฟังก์ชันที่ที่ใช้ในส่วนนี้มีดังนี้

CAudio_Init

CAudio_Cleanup

Play_Sound

Stop_Sound

Load_Sound

Setloop_Sound

Resetloop_Sound

การทำงานของแต่ละฟังก์ชัน

3.2.3.1 CAudio_Init

ฟังก์ชันทำการเตรียมให้ระบบพร้อมที่จะใช้ Direct Sound โดยทำการสร้าง Direct Sound ออปเจค กำหนดลักษณะการทำงานของ Direct Sound

3.2.3.2 CAudio_Cleanup

ทำการยกเลิกการใช้ Direct Sound ยกเลิกออปเจคของ Direct Sound เพื่อคืนหน่วยความจำกลับสู่ระบบ

3.2.3.3 Load_Sound

ทำการโหลดไฟล์เสียงที่จำเป็นต้องใช้ในโปรแกรมทั้งหมด เข้าสู่หน่วยความจำ

3.2.3.4 Play_Sound

ทำการเล่นเสียงที่ถูกโหลดเข้ามาในหน่วยความจำ

3.2.3.5 Stop_Sound

ใช้เมื่อต้องการหยุดเสียงที่กำลังเล่นอยู่ในขณะนั้น

3.2.3.6 Setloop_Sound

กำหนดการเล่นเสียงตั้งแต่เริ่มต้นจนจบซ้ำไปเรื่อย ๆ

3.2.3.7 Resetloop_Sound

ยกเลิกการเล่นเสียงซ้ำ ๆ ที่เล่นอยู่ในขณะนั้น

3.2.4 การกำหนดตำแหน่งของ Object ใน 3D World Space

3D Object ทุกตัว ล้วนแล้วแต่มีตำแหน่งในแกน x, y และ z เป็นของตัวเอง ซึ่ง Object ในที่ไม่มีการเคลื่อนที่ เช่น ตึก อาคาร ฯลฯ สร้างขึ้นจาก class OMeshObject ซึ่ง ตัวแปร x,y และ z ใน class นี้ ประกาศเป็นตัวแปรประเภท private ไม่สามารถอ้างถึงได้โดยตรงจากภายนอก class ดังนั้นการกำหนดตำแหน่งของ Object จาก class OMeshObject นี้ทำได้โดย เรียกใช้ method SetupWorldSpace(FLOAT _x, FLOAT _y, FLOAT _z)

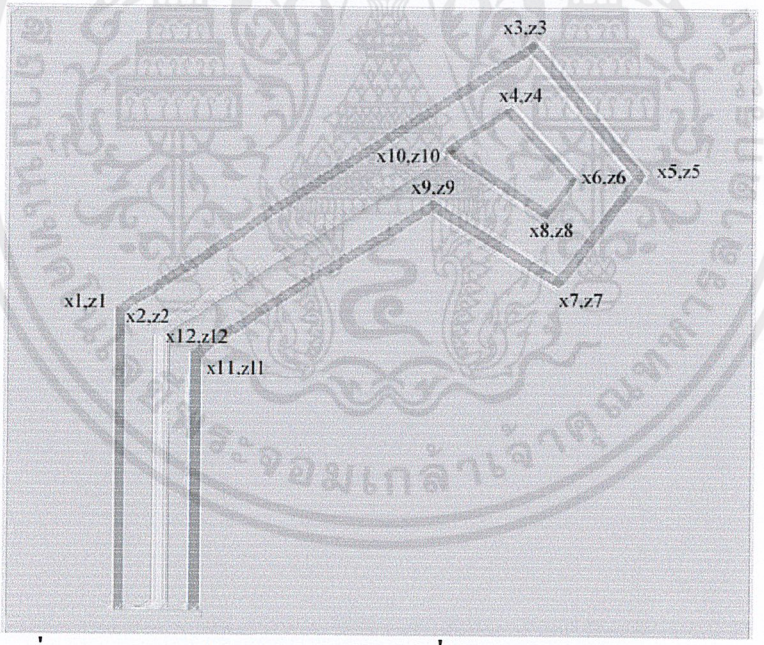
```
OMap = new OMeshObject();
if ( FAILED( OMap->Create(g_pd3dDevice, "mapreal.x")))
        return E_FAIL;
OMap->SetupWorldSpace(0,0,0);
```

ส่วนการกำหนดตำแหน่งของ Object รถ ซึ่งมีการเคลื่อนที่นั้น กำหนดเพียงแค่จุดเริ่มต้น ของรถเท่านั้น โดยใช้ method InitCar(FLOAT _x, FLOAT _Y, FLOAT _Z) หลังจากนั้น ตำแหน่งของรถจะเปลี่ยนไปตามการควบคุมของผู้เล่น โดยตำแหน่งใหม่จะเกิดจากการคำนวณของโปรแกรม

```
MyCar = new OPlayerCar();
if( FAILED( MyCar->Create(g_pd3dDevice,"camry.x")))
        return E_FAIL;
MyCar->InitCar(0, 0, 0);
```

3.2.5 การกำหนดขอบเขตการชนของฉาก

เนื่องจากฉากในเกมนี้ถูกสร้างขึ้นจากโปรแกรม 3ds max ซึ่งเป็นโปรแกรมที่สามารถสร้างภาพ 3 มิติได้ง่ายกว่าและสะดวกกว่าการสร้างภาพโดยการเขียนโปรแกรมเองมาก แต่การทำภาพจากโปรแกรม 3ds max ก็มีข้อเสียตรงที่ว่าเราไม่สามารถทราบตำแหน่งที่แท้จริงของฉากได้ว่าตัวตึกหรือ ขอบถนนต่าง ๆ ที่จะทำการตรวจสอบนั้นอยู่ที่จุดใดเมื่อนำฉากมารวมกับตัวโปรแกรมจึงต้องแก้ปัญหานี้โดยใช้วัตถุขนาดเล็กเคลื่อนไปยังตำแหน่งที่ต้องการตรวจสอบแล้วอ่านค่าตำแหน่งออกมาแสดงให้เราทราบ ซึ่งผลของการตรวจสอบจะแสดงได้ดังรูป



รูปที่3.18 แสดงจุดคู่ลำดับของตำแหน่งที่ต้องการตรวจสอบ

```
static   FLOAT  x1 = 1.00f ,      z1 = -120.95f,
                x2 = -1.31f,      z2 = -120.07f,
                x3 = -82.66f,     z3 = -216.18f,
        x4 = -80.78f,     z4 = -209.84f,
```

x5 = -97.19f, z5 = -203.41f,

x6 = -90.04f, z6 = -201.86f,

x7 = -91.01f, z7 = -184.91f,

x8 = -86.97f, z8 = -193.50f,

x9 = -70.05f, z9 = -193.72f,

x10 = -72.26f, z10 = -199.66f,

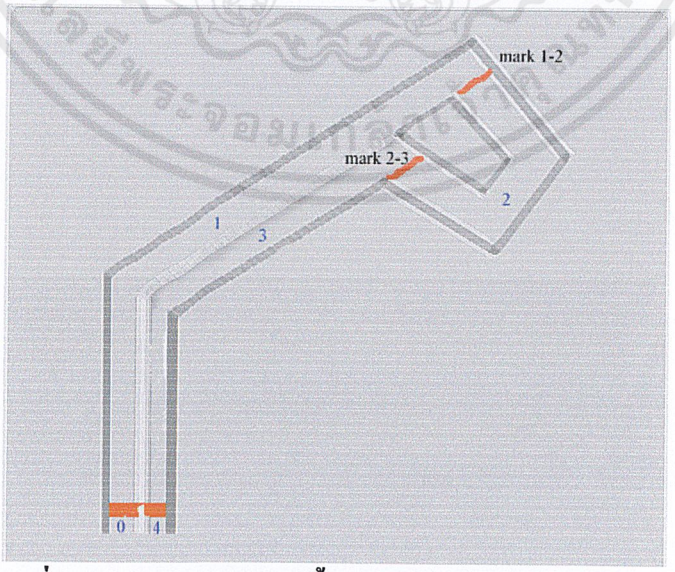
x11 = -4.00f, z11 = -119.06f,

x12 = -1.68f, z12 = -119.93f;

เมื่อเราได้ตำแหน่งของจุดต่าง ๆ ในฉากมาแล้ว ก็จะสามารถสร้างเป็นสมการเส้นตรงของขอบถนนที่เราต้องการได้ และเมื่อเรามีสมการเส้นตรงของขอบถนนแล้วเราก็จะสามารถนำสมการนี้มาใช้ตรวจสอบการชนได้ต่อไป

<u>การแบ่งฉากออกเป็นช่วง ๆ (Mark)</u>

จากฉากที่เราสร้างไว้จะเป็นการเดินรถแบบวิ่งไปตามทางเรื่อย ๆ พอถึงจุดกลับรถก็จะวิ่งกลับทางเดิม แต่ต่างกันที่รถที่วิ่งจากจุดเริ่มต้นกับรถที่วิ่งเข้าเส้นชัยนั้นอยู่คนละฝั่งถนนกัน จึงทำให้เป็นการยากที่จะตรวจสอบว่า ช่วงเวลาในขณะนี้เราจะใช้เงื่อนไขใดในการตรวจสอบว่า เกิดการชนระหว่างรถที่เราบังคับกับขอบถนนหรือไม่ จึงต้องแก้ไขปัญหาในจุดนี้โดยการแบ่งฉากทั้งหมดออกเป็นช่วง ๆ และเป็นการกำหนดขอบเขตของเงื่อนไขให้ตรวจสอบเฉพาะเงื่อนไขที่มีความเกี่ยวข้องเท่านั้น ดังรูป



รูปที่3.19 แสดงการแบ่งฉากทั้งหมดออกเป็นช่วง ๆ

<u>เงื่อนไขในการเปลี่ยนช่วง (เปลี่ยน Mark)</u>

```
if ( ( x == 0.0f ) && ( z == 0.0f ) )
    { mark = 0 ; }
```

เงื่อนไขนี้เป็นการกำหนดค่า mark เริ่มต้นให้กับการเริ่มเล่น โดยที่เมื่อทำการเริ่มเล่นตัวรถจะอยู่ที่ตำแหน่ง x = 0 และ y = 0 ทุกครั้ง การทำเช่นนี้เป็นการทำเพื่อให้การเปลี่ยน mark ในขั้นตอนต่อไปถูกต้อง และเป็นมาตรฐานเดียวกันทุกครั้งเมื่อเริ่มเล่นใหม่

```
else if ( ( mark == 0 ) && ( cx != 0.0f ) && ( cz != 0.0f ) )
    { mark = 1 ; }
```

เงื่อนไขนี้เป็นการกำหนดค่า mark ในขณะที่รถได้มีการเคลื่อนที่ครั้งแรกของการเล่นในรอบนั้น และเป็นการเริ่มจับเวลาในการแข่งขันด้วย

```
if ( ( mark == 1 ) && ( cz > ( m2*(cx-x4)+z4 ) ) && ( cx < x4 ) )
    { mark = 2 ; }
```

เงื่อนไขนี้จะใช้เมื่อรถวิ่งไปถึงจุดแบ่งช่วงระหว่าง mark = 1 และ mark = 2 ซึ่งก็คือจุดกลับรถนั่นเอง โดยการที่จะสามารถเปลี่ยนค่า mark ไปยัง mark = 2 และใช้เงื่อนไขการตรวจสอบการชนของ mark =2 ได้นั้น ค่าของ mark เดิมต้องมีค่าเท่ากับ 1 และต้องผ่านจุดเปลี่ยน mark(1-2) มาแล้ว

```
if ( ( mark == 2 ) && ( cz < ( m2*(cx-x4)+z4 ) ) && ( cx < x2 ) )
    { mark = 1 ; }
```

เงื่อนไขนี้จะเป็นการย้อนกลับมาจากเงื่อนไขเดิม กล่าวคือรถอยู่ที่ mark = 2 แต่ต้องการย้อนกลับมายัง mark = 1

```
if ( ( mark == 2 ) && ( cz < ( m9*(cx-x9)+z9 ) ) && ( cx > x4 ) && ( cx < x11 ) )
    { mark = 3 ; }
```

เงื่อนไขนี้จะใช้เมื่อรถวิ่งไปถึงจุดแบ่งช่วงระหว่าง mark = 2 และ mark = 3 ซึ่งก็คือจุดกลับรถจะเข้าสู่ทางตรงเพื่อย้อนกลับไปยังเส้นชัย โดยการที่จะสามารถเปลี่ยนค่า mark ไปยัง mark = 3 และใช้เงื่อนไขการตรวจสอบการชนของ mark =3 ได้นั้น ค่าของ mark เดิมต้องมีค่าเท่ากับ 2 และต้องผ่านจุดเปลี่ยน mark(2-3) มาแล้ว

```
if ( ( mark == 3 ) && ( cz > ( m9*(cx-x9)+z9 ) ) && ( cx > x4 ) && ( cx < x9 ) )
    { mark = 2 ; }
```

เงื่อนไขนี้จะเป็นการย้อนกลับมาจากเงื่อนไขเดิม กล่าวคือรถอยู่ที่ mark = 3 แต่ต้องการย้อนกลับมายัง mark = 2

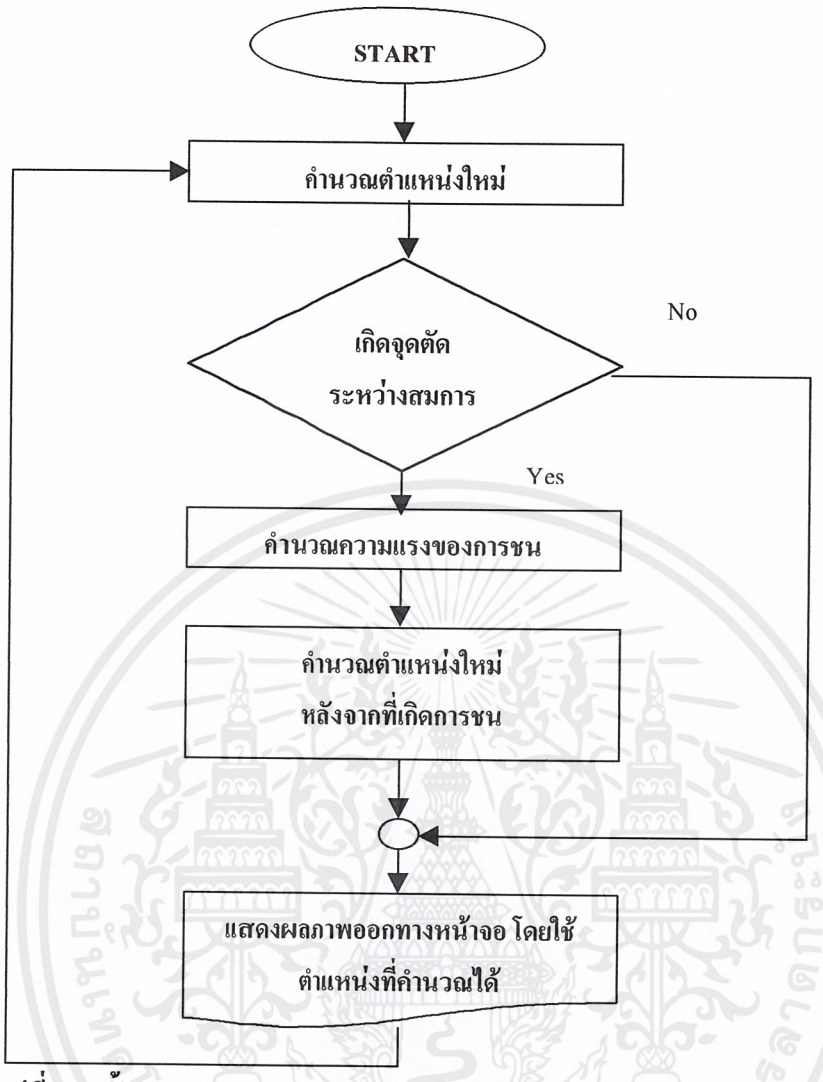
```
if ( ( mark == 3 ) && ( z > -1.0f) )
    { mark = 4 ; }
```

เงื่อนไขนี้เป็นการเข้าสู่เส้นชัยเพื่อทำการหยุดเวลา โดยการจะเข้าสู่ mark = 4 ได้นั้น ค่าของ mark เดิมจะต้องเป็น 3 เท่านั้น

จะเห็นได้ว่าการที่เราแบ่งฉากทั้งหมดออกเป็นช่วง ๆ จะสามารถช่วยให้การตรวจสอบเงื่อนไขทำได้ง่ายขึ้นและเป็นขั้นตอนมากขึ้นด้วย ซึ่งถ้าฉากของเรามีขนาดใหญ่กว่านี้ก็สามารถแบ่งฉากออกเป็นช่วงได้อีกตามความเหมาะสม

3.2.6 หลักการตรวจสอบการชน

ในการตรวจสอบว่าเมื่อใดที่จะเกิดการชนขึ้น ทำได้โดยใช้สมการเส้นตรงที่ได้จากตำแหน่งรถเดิมกับตำแหน่งรถใหม่ ซึ่งจะได้เป็นจุดคู่อันดับ 2 จุด แล้วจึงนำจุด 2 จุดที่ได้มาสร้างเป็นสมการเส้นตรงของรถขึ้น จากนั้นจึงเอาสมการเส้นตรงของรถมาคำนวณกับสมการเส้นตรงของขอบทางตามเงื่อนไขในแต่ละช่วงที่ได้ทำการแบ่งไว้ก่อนหน้านี้ (แบ่งช่วงของฉาก) ถ้าหากว่าสมการเส้นตรงของรถกับสมการเส้นตรงของขอบทางไม่เกิดการตัดกันระหว่าง 2 สมการก็จะแสดงว่าไม่เกิดการชนกัน แต่ถ้าเกิดการตัดกันของสมการเส้นตรงทั้งสอง ก็จะแสดงว่าตัวรถเกิดการชนขึ้นกับขอบทาง ก็จึงค่อยทำตามขั้นตอนหลังเกิดการชนกันต่อไป และใช้จุดที่มีการตัดกันของสมการเป็นจุดที่เกิดการชนขึ้นนั่นเอง



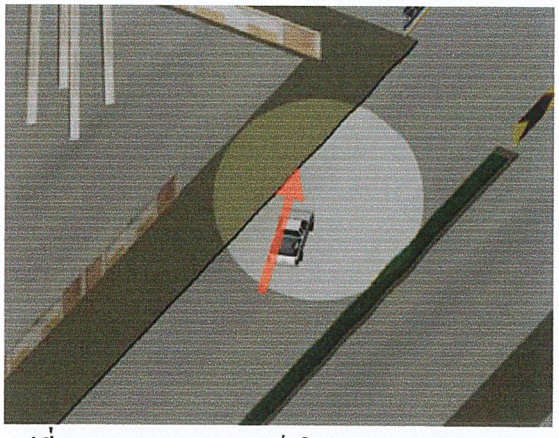
รูปที่3.20 ขั้นตอนการคำนวณการชน

<u>การคำนวณหลังจากเกิดการชน</u>

หลังจากที่พบจุดตัดของสมการเส้นตรงของรถกับสมการเส้นตรงของเส้นขอบทางก็จะแสดงว่าเกิดการชนกันขึ้น หลักการทำงานหลังจากที่มีการชนกันนั้นคือ

พิจารณาความเร็วของรถว่ามีความเร็วเท่าใด เพื่อนำมาใช้คำนวณความเร็วของรถหลังจากที่เกิดการชนขึ้น และคำนวณหาแรงสะท้อนว่าจะให้สะท้อนออกไปแรงเท่าใด

พิจารณาทิศทางของตัวรถเมื่อทำการปะทะ เพื่อหาทิศทางต่อไปของตัวรถหลังจากที่ชน



ก่อนเกิดการชน

ข้อมูลเข้า

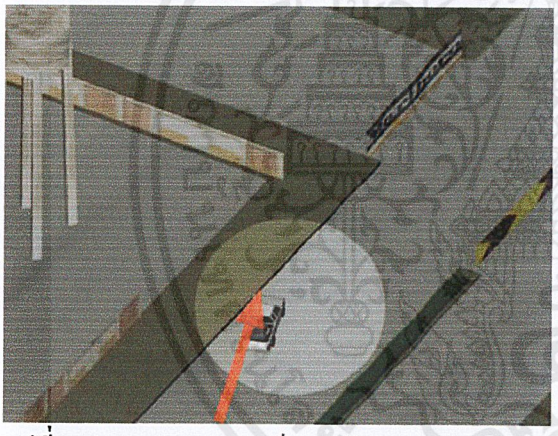
- ความเร็วของรถ
- ทิศทางของรถ

ผลลัพธ์

- ความเร็วรถเท่าเดิม
- รถวิ่งไปทิศทางเดิม

รูปที่3.21 แสดงเหตุการณ์ ก่อนเกิดการชน

ในกรณีที่สมการเส้นตรงของรถกับสมการเส้นตรงของขอบทางไม่เกิดการตัดกันขึ้น ก็แสดงว่าทิศทางการเคลื่อนที่ของรถไม่ได้วิ่งไปโดนขอบทาง ดังนั้น ความเร็วของรถและทิศทางของรถก็จะไม่มีการเปลี่ยนแปลง แล้วจึงแสดงผลออกหน้าจอ



เกิดการชนขึ้น

ข้อมูลเข้า

- ความเร็วของรถ
- ทิศทางของรถ

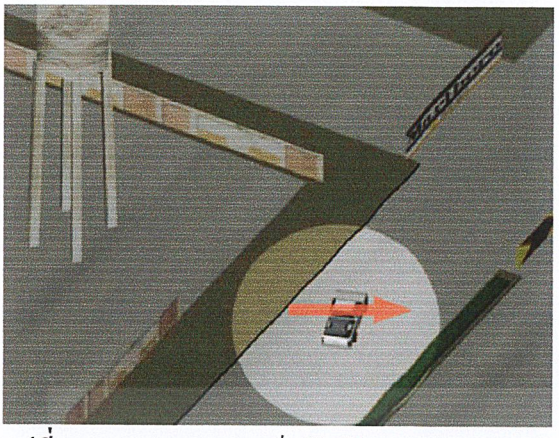
รูปที่3.22 แสดงเหตุการณ์ขณะเกิดการชน

แต่ในกรณีที่ตำแหน่งรถใหม่ที่คำนวณได้ เมื่อนำมาสร้างสมการเส้นตรงของรถขึ้นนั้น เกิดการตัดกันของสมการเส้นตรงของรถกับสมการเส้นตรงของขอบทาง ก็จะแสดงว่าเกิดการชนขึ้น ดังนั้นจะแสดงภาพออกทางหน้าจอเลยแบบกรณีที่ไม่เกิดการชนไม่ได้ คือจะต้องมีการคิดคำนวณในกรณีของเงื่อนไขเมื่อเกิดการชนดังนี้

ลดค่าความเร็วของรถหลังจากที่ทำการชน

คำนวณทิศทางของการสะท้อนเมื่อเกิดการชน

ระยะของการสะท้อนจะแปรผันตามความเร็วก่อนการชน



รูปที่3.23 แสดงเหตุการณ์หลังเกิดการชน

ผลลัพธ์

- ความเร็วของรถลดลง ตามระดับความเร็วของรถก่อนที่จะเกิดการชน
- เกิดการสะท้อนออกจากจุดที่เกิดการชนขึ้น โดยคำนวณจากทิศทางการชน
- ตำแหน่งรถใหม่จะคำนวณจากความเร็วของรถก่อนเกิดการชน ยิ่งความเร็วมากก็จะยิ่งสะท้อนแรง

เมื่อคำนวณตำแหน่งรถใหม่หลังเกิดการชนได้แล้ว ก็จะนำตำแหน่งรถนั้นแสดงผลออกทางหน้าจอ

บทที่ 4
ผลการทดลองและการวิเคราะห์ปัญหา

การทดลองและการวิเคราะห์ปัญหาที่กล่าวถึงในบทนี้ จะเป็นขั้นตอนการทดสอบและผลการทดสอบที่ได้แต่ละขั้น โดยผลการทดสอบนี้ จะถูกนำไปวิเคราะห์ถึงปัญหาและแนวทางในการพัฒนาต่อไปในอนาคต โดยหวังว่าจะเป็นประโยชน์แก่ผู้จะนำไปศึกษาเพื่อพัฒนาต่อ และเพื่อให้เห็นถึงปัญหา และข้อดีข้อเสียของปัญหานี้โดยที่ผู้ศึกษาไม่จำเป็นต้องทำการทดสอบเพื่อหาปัญหาและนำไปพัฒนาต่ออีกครั้ง

คุณสมบัติของระบบที่นำมาทดสอบ

- ระบบปฏิบัติการ Microsoft Window 98
- ติดตั้ง Microsoft DirectX เวอร์ชั่น 8.0 runtime
- เครื่องคอมพิวเตอร์ที่มีซีพียูความเร็ว 600 MHz
- หน่วยความจำหลักขนาด 128 Mbyte

ขั้นตอนการดำเนินการทดสอบ

- ทำการติดตั้งตัวโปรแกรมและซอฟต์แวร์ที่จำเป็นต้องใช้
- ทำการรันและประมวลผลภายใต้ระบบที่กำหนด
- ตรวจสอบการตอบสนองของรถขณะทำการควบคุมโดยใช้แป้นพิมพ์
- ตรวจสอบการตอบสนองการชนกันของตัวรถกับฉาก
- ตรวจสอบการหา Error

การประเมินผล

- หลังจากติดตั้งตัวโปรแกรมแล้วสามารถใช้โปรแกรมได้
- การประมวลผลของโปรแกรมจะต้องใช้ความเร็วในระดับที่ยอมรับได้
- การเคลื่อนที่ของรถเป็นไปตามการควบคุม
- ฉากเปลี่ยนแปลงได้ตามต้องการ
- เมื่อเกิดการชนกันของรถกับฉากต้องมีการตอบสนอง
- จำนวน Error ที่เกิดขึ้นต้องไม่มี หรือน้อยที่สุด

ต่อไปจะกล่าวถึงรายละเอียดของขั้นตอนการทดสอบ และผลของการทดสอบที่ได้

4.1 ขั้นตอนการทดสอบการติดตั้งโปรแกรมและซอฟต์แวร์ที่จำเป็น

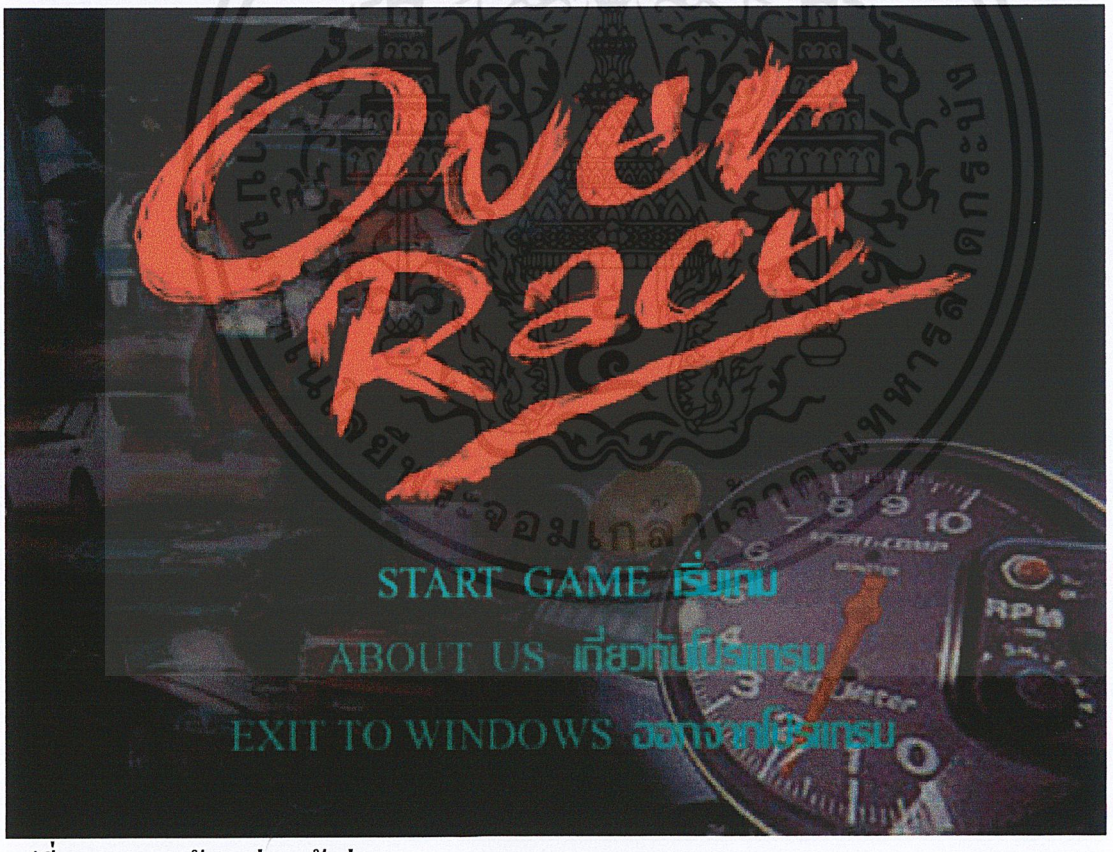
- ติดตั้งตัวโปรแกรม โดยคลิกไฟล์ " SETUP.EXE "
- ตัวโปรแกรมและซอฟต์แวร์จะถูกทำการติดตั้ง
- ผู้เล่นทำการ Run โปรแกรม โดยคลิกที่ไฟล์ "OVERRACE.EXE"

ผลการทดสอบ

- ติดตั้งโปรแกรมและซอฟต์แวร์ที่จำเป็นลงบนเครื่องคอมพิวเตอร์ได้
- ผู้เล่นสามารถรันโปรแกรมได้
- เนื้อที่ฮาร์ดดิสก์ถูกใช้ไปประมาณ 45 Mbyte

4.2 ขั้นตอนการทดสอบการรันโปรแกรมและการประมวลผลภายใต้ระบบที่กำหนด

รันโปรแกรมภายใต้เครื่องคอมพิวเตอร์ที่กำหนด โดยคลิกที่ไฟล์ "OVERRACE.EXE"จะได้ดังรูปที่ 4.1



รูปที่4.1 แสดงหน้าจอก่อนเข้าสู่เกม

จากรูปจะเห็นได้ว่ามีทางเลือก 3 ทาง

1. *FREE RUN ขับอิสระ* เริ่มเล่นเกม

2. *ABOUT US เกี่ยวกับโปรแกรม* บทความที่เกี่ยวกับผู้ที่ทำการพัฒนา
3. *EXIT TO WINDOWS ออกจากโปรแกรม* ออกจากเกม

ผลการทดสอบ

- สามารถรันโปรแกรมได้บนระบบที่กำหนด
- การแสดงหน้าจอแรกทำได้สมบูรณ์ดี
- สามารถเลือกรายการที่แสดงได้ตามต้องการของผู้เล่น

4.2.1 ขั้นตอนทดสอบเริ่มแรกโดยการเลือกหัวข้อ "ABOUT US เกี่ยวกับโปรแกรม" จะได้หน้าจอดังรูป

รูปที่4.2 หน้าจอ "ABOUT US เกี่ยวกับโปรแกรม"

ผลทดสอบ

- แสดงหน้าจอ "ABOUT US เกี่ยวกับโปรแกรม" ได้ถูกต้อง
- สามารถย้อนกลับมาสู่หน้าจอเมนูหลักได้

4.2.2 ขั้นตอนการเลือกหัวข้อ “EXIT TO WINDOWS ออกจากโปรแกรม”

ผลทดสอบ

- ออกจากโปรแกรมได้
- โปรแกรมสามารถคืนทรัพยากรให้สู่หน่วยความจำของระบบ
- ความเร็วของวินโดว์หลังออกจากโปรแกรมอยู่ในระดับปกติ

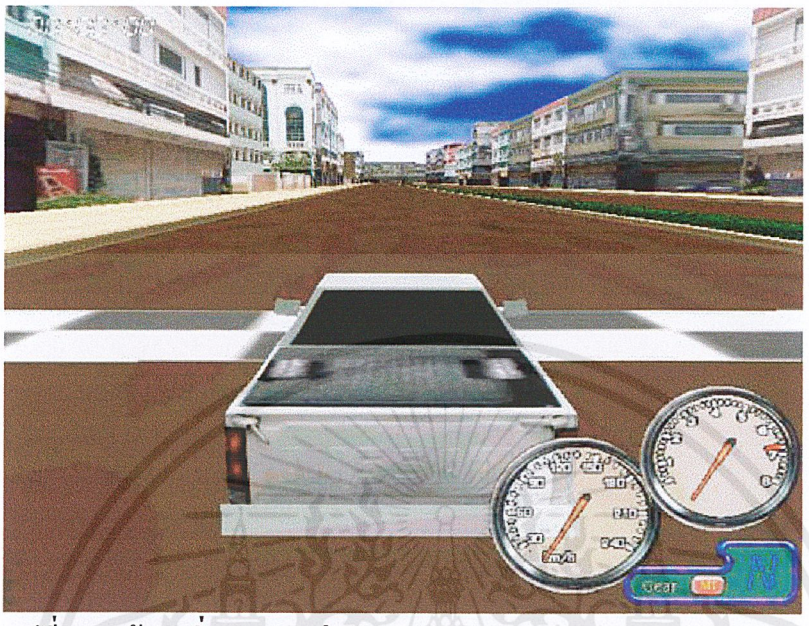
4.2.3 ขั้นตอนการเลือกหัวข้อ “FREE RUN ขับอิสระ” จะได้หน้าจอต่อไปดังรูป

รูปที่4.3 หน้าจอเลือกรถที่จะขับ

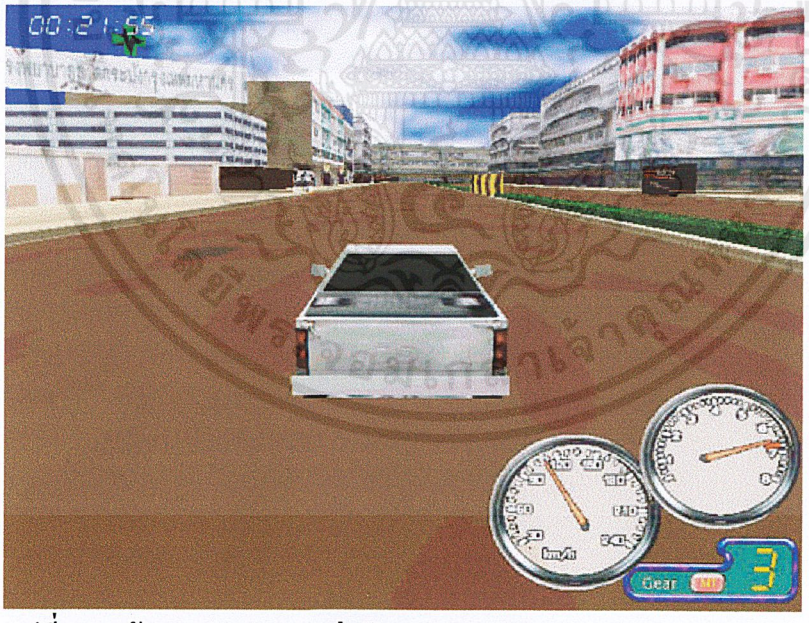
ผลทดสอบ

- โปรแกรมสามารถแสดงหน้าจอเลือกรถได้
- รูปของรถหมุนรอบตัวเองอยู่กลางหน้าจอ
- สามารถเลือกรถคันอื่นโดยการกดปุ่มลูกศร ซ้าย หรือ ขวา และรูปรถที่แสดงกับรายละเอียดของรถเปลี่ยนไปตามที่เลือก
- สามารถแสดงรถได้ทุกคันที่มีอยู่
- ความเร็วในการแสดงภาพบนหน้าจออยู่ในระดับที่ยอมรับได้
- เมื่อเลือกรถแล้วสามารถเปลี่ยนหน้าจอเป็นหน้าจอทำการแข่งขันได้

4.2.4 หน้าจอเริ่มการแข่งขันหลังจากทำการเลือกรถแล้ว



รูปที่4.4 หน้าจอเริ่มทำการแข่งขัน



รูปที่4.5 หน้าจอขณะทำการแข่งขัน

ผลการทดสอบ

- โปรแกรมแสดงภาพหน้าจอขณะทำการแข่งขันได้ ดังรูปที่ 4.4
- สามารถเร่งความเร็วของรถได้เมื่อกดแป้นพิมพ์ Z

- สามารถลดความเร็วของรถได้เมื่อกดแป้นพิมพ์ X
- สามารถควบคุมการเลี้ยวของรถได้โดยการกดปุ่มลูกศรซ้ายหรือขวา
- สามารถปรับระดับเกียร์ให้เพิ่มขึ้นหรือลดลงได้โดยการกดปุ่มลูกศรขึ้นหรือลง
- สามารถออกจากหน้าจอการแข่งขันได้เมื่อสิ้นสุดการแข่งหรือกดปุ่ม Esc

4.3 ขั้นตอนการทดสอบการทำงานของตัวรถเมื่อทำการควบคุม

การทดสอบการควบคุมของรถ และผลการทำงานจะแสดงไว้ในตารางที่ 4.1

การทดสอบ	วิธีทดสอบ	ผลการทดสอบ
การทดสอบการเร่งความเร็ว	กดแป้นพิมพ์ Z ค้างไว้	ตัวรถเคลื่อนที่ไปข้างหน้าโดยมีความเร็วเพิ่มขึ้นเรื่อยๆตามเวลาที่กดแป้นพิมพ์ Z ค้างไว้
การทดสอบการลดลงของความเร็วเมื่อทำการปล่อยปุ่มแป้นพิมพ์ Z	เร่งความเร็วรถถึงระดับที่ต้องการ ,ปล่อยปุ่มแป้นพิมพ์ Z	ตัวรถค่อยๆลดความเร็วลงอย่างช้าๆจนรถหยุดนิ่ง
การทดสอบการกดปุ่มลดความเร็ว	เร่งความเร็วรถถึงระดับที่ต้องการ ,กดแป้นพิมพ์ X ค้างไว้	ตัวรถลดความเร็วลงอย่างรวดเร็วจนรถหยุดนิ่ง
การทดสอบการเพิ่มเกียร์	เร่งความเร็วจนถึงระดับสูงสุดในแต่ละเกียร์ , กดแป้นลูกศรขึ้น	ความเร็วของตัวรถสามารถเพิ่มขึ้นได้ต่อไปจนถึงความเร็วสูงสุดในระดับนั้น
การทดสอบการลดเกียร์	เร่งความเร็วในระดับสูงสุดของเกียร์ 2 , กดแป้นลูกศรลง เพื่อลดเกียร์ให้เป็นเกียร์ 1	ความเร็วลดลงมาอยู่ที่ระดับความเร็วสูงสุดของเกียร์ 1
การทดสอบการเลี้ยวซ้าย (เดินหน้า)	กดแป้นพิมพ์ Z ค้างไว้ , กดปุ่มลูกศรซ้ายค้างไว้	ตัวกล้องเบี่ยงไปทางซ้าย แล้วจึงทำการเลี้ยวซ้าย
การทดสอบการเลี้ยวขวา (เดิน	กดแป้นพิมพ์ Z ค้างไว้ , กดปุ่ม	ตัวกล้องเบี่ยงไปทางขวา แล้ว

หน้า)	ลูกศรขวาค้างไว้	จึงทำการเลี้ยวขวา
การทดสอบการถอยหลัง	เปลี่ยนเกียร์เป็นเกียร์ R , กดแป้นพิมพ์ Z ค้างไว้	รถเคลื่อนที่ถอยหลัง
การทดสอบถอยไปทางซ้าย	เปลี่ยนเกียร์เป็นเกียร์ R , กดแป้นพิมพ์ Z ค้างไว้ , กดปุ่มลูกศรซ้ายค้างไว้	รถเคลื่อนที่ถอยหลังไปทางซ้ายมือ
การทดสอบถอยไปทางขวา	เปลี่ยนเกียร์เป็นเกียร์ R , กดแป้นพิมพ์ Z ค้างไว้ , กดปุ่มลูกศรซ้ายค้างไว้	รถเคลื่อนที่ถอยหลังไปทางขวามือ
การทดสอบเปลี่ยนมุมมอง	กดแป้นพิมพ์ V	มุมมองของกล้องเปลี่ยนไปยังอีกมุมมองหนึ่ง
การทดสอบการชนของตัวรถกับตัวฉาก	เคลื่อนตัวรถให้วิ่งเข้าหาขอบทางด้วยความเร็วระดับต่างๆ	รถไม่สามารถวิ่งผ่านขอบทางไปสู่ตัวอาคารได้
การทดสอบผลที่ได้หลังจากที่เกิดการชน	เคลื่อนตัวรถเข้าหาขอบทางด้วยความเร็วระดับต่างๆ และในมุมต่างๆ	ตัวรถที่ทำการชนจะมีความเร็วลดลงอย่างเห็นได้ชัด ไม่เกิดการทะลุผ่านสิ่งกีดขวางหรือขอบทาง และมีการสะท้อนออกจากจุดที่เกิดการชน
การทดสอบการเข้าเส้นชัย	ขับรถเข้าเส้นชัย	เมื่อรถเข้าสู่เส้นชัยเวลาจะหยุดเดิน ความเร็วของรถจะค่อยๆ ลดลงจนถึง 0 จากนั้นกล้องจะทำการเคลื่อนที่ไปรอบตัวรถ

ตารางที่ 4.1 การทดสอบการควบคุมของรถ และผลการทำงาน

4.4 ขั้นตอนการทดสอบฉากในการแข่งขัน

- เคลื่อนรถไปทุกที่ของฉากที่รถสามารถไปได้
- ตรวจสอบความครบถ้วนของฉากที่ทำไว้กับที่มีอยู่ในเกม
- ทำการวิ่งเข้าชนกับสถานที่ต่างๆในฉาก

ผลการทดสอบ

- รถสามารถเคลื่อนที่ไปยังสถานที่ต่างๆภายในฉากได้
- มีสถานที่ครบตามที่ได้ทำการออกแบบไว้
- มีการเปลี่ยนแปลงของฉากตามตำแหน่งต่างๆที่รถวิ่งไปได้ดี
- ความเร็วของตัวเกมอยู่ในระดับที่ยอมรับได้
- ฉากที่เราไม่ต้องการให้ผ่านจะเกิดการชนขึ้นทุกจุดที่ไม่ให้ผ่าน

4.5 ขั้นตอนการทดสอบการหาข้อผิดพลาดของโปรแกรม

ในขั้นตอนการทดสอบเพื่อหาข้อผิดพลาดจะไม่มีรูปแบบที่ชัดเจน แต่จะทำการทดสอบโดยการเล่นเกมหลายๆรอบ เพื่อหาข้อผิดพลาด และจากการทดสอบพบว่าข้อผิดพลาดที่พบจะได้แก่

- การโต้ตอบในกรณีที่เกิดการชนยังมีข้อผิดพลาด
- การเคลื่อนที่ของรถยังทำได้ไม่ดีเท่าที่ควร
- ยังไม่มีการใส่รถคันอื่นภายในฉาก

บทที่5
สรุปและข้อเสนอแนะ

5.1 บทสรุป

จากการทดลองโปรแกรมการจำลองการขับขี่ยานพาหนะแบบเสมือนจริงได้ผลสรุปดังนี้

5.1.1 โปรแกรมการจำลองการขับขี่ยานพาหนะแบบเสมือนจริงนี้ มีหน้าจอให้เลือกรถยนต์ได้ 2 แบบ คือ รถกระบะนิสสันสีขาว และรถเก๋งโตโยต้าคัมรี่ โดยผู้เล่นสามารถเลือกรถได้ทีละ 1 คัน

5.1.2 เมื่อผู้เล่นเลือกรถเสร็จแล้ว โปรแกรมสามารถให้ผู้เล่นบังคับรถได้ดังนี้

- เลี้ยวซ้าย/ขวา
- เดินหน้า/ถอยหลัง
- เร่งความเร็ว/เบรกรถ
- เปลี่ยนเกียร์/เปลี่ยนมุมมองรถ

5.1.3 เมื่อรถเริ่มเคลื่อนที่ผู้เล่นสามารถบังคับทิศทางของรถให้เลี้ยวซ้าย/ขวาได้ อีกทั้งยังสามารถเร่งความเร็วของรถหรือเบรกได้อย่างอิสระ

5.1.4 เมื่อเกิดการชนตัวรถจะไม่ทะลุผ่านไป แต่จะสะท้อนออกมาในทิศตรงข้าม และมีการแสดงเสียงให้รู้เมื่อเกิดการชนขึ้น

5.1.5 สามารถเปลี่ยนมุมมองได้ 2 แบบ คือ มุมมองภายในรถ และมุมมองภายนอกรถ โดยผู้เล่นสามารถเปลี่ยนมุมมองได้อย่างอิสระในขณะเล่น

5.1.6 เมื่อวิ่งไปจนถึงเส้นชัยแล้วเวลาของการแข่งขันจะหยุดเดิน ความเร็วรถจะลดลงถึง 0 และจะมีตัวหนังสือแสดงให้รู้ว่าเข้าเส้นชัยแล้ว

5.1.7 โปรแกรมมีส่วนให้เข้าไปดูรายละเอียดของผู้พัฒนาโปรแกรม

5.2 ข้อจำกัดในการใช้งานโปรแกรม

โปรแกรมการจำลองการขับขี่ยานพาหนะแบบเสมือนจริง มีข้อจำกัดของโปรแกรมดังนี้

5.2.1 เป็นโปรแกรมที่ผู้เล่นทำการเล่นเพียงคนเดียว ไม่มีคู่แข่งขันที่เป็นคอมพิวเตอร์ หรือที่เป็นผู้เล่นคนอื่นเลย

5.2.2 โปรแกรมใช้คีย์บอร์ดเป็นตัวควบคุมเพียงอย่างเดียว

5.2.3 โปรแกรมมีฉากให้เลือกในการแข่งขันเพียงแค่ฉากเดียว

5.2.4 รถที่ให้เลือกในการแข่งขันมีแค่ 2 แบบ ซึ่งน้อยเกินไปสำหรับเกมแข่งรถ

5.2.5 เสียงดนตรีประกอบในโปรแกรมยังไม่สมจริงมากนัก

5.3 บทการวิจารณ์และแนวทางการพัฒนา

5.3.1 เนื่องจากโปรแกรมนี้ยังไม่มีคู่แข่งขันกับผู้เล่น ดังนั้นควรมีการพัฒนาโปรแกรมให้สามารถเพิ่มผู้เล่นเข้าในเกมไม่ว่าจะเป็นการแข่งกับเพื่อนหรือแข่งกับคอมพิวเตอร์ก็ตาม ซึ่งจะทำให้เกิดความตื่นเต้น และจะทำให้สนุกสนานมากขึ้น

5.3.2 เนื่องจากใช้คีย์บอร์ดเป็นตัวควบคุมจึงทำให้การบังคับไม่ถนัดนัก ดังนั้นควรจะมีการพัฒนาโปรแกรมให้สามารถใช้ได้ทั้งคีย์บอร์ด เมาส์ จอยสติก ตามความถนัดของผู้เล่นแต่ละคน

5.3.3 ควรจะมีการเพิ่มฉากและรถให้มากขึ้นเพื่อความหลากหลาย ทำให้ผู้เล่นไม่เกิดความเบื่อหน่ายเพราะมีรถให้เลือกหลายแบบ และมีฉากให้เลือกหลายฉาก

5.3.4 เสียงประกอบในโปรแกรมควรจะให้เสมือนจริง เช่น เสียงเร่งเครื่องยนต์ เสียงเบรก เป็นต้น ทำให้ผู้เล่นรู้สึกเสมือนขับรถจริงๆ

เพื่อประโยชน์ในการพัฒนาต่อไป จึงแนะนำให้หาหลักการแสดงภาพใหม่ก่อนเพื่อให้แสดงผลภาพได้เร็วขึ้น เมื่อมีการเพิ่มคู่แข่งลงไปจะได้ไม่มีผลกับความเร็วในการแสดงผลมากนัก

บรรณานุกรม

นิรุธ อำนวยศิลป์. 2543. หนังสือ Microsoft Visual C++ 6. ครั้งที่1.กรุงเทพฯ.: สำนักพิมพ์ Success Media

ชัยวัฒน์ คำรัตน์. 2542. หนังสือ เขียนเกม 3 มิติ ด้วย DirectX เล่ม1,2. ครั้งที่1.กรุงเทพฯ.: สำนักพิมพ์ Infopress

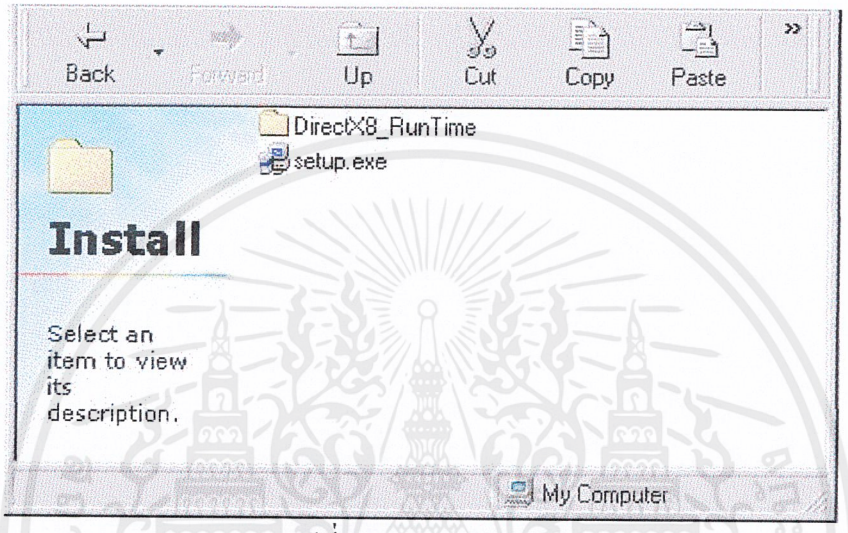
ธาริน สิทธิธรรมชารี. 2542. หนังสือ Inside Direct3D. ครั้งที่5.กรุงเทพฯ.: สำนักพิมพ์ Success Media

สุรสิทธิ์ คิวประสพศักดิ์. 2543. Microsoft DirectX 8 SDK Documentation (for VC++). ครั้งที่1. กรุงเทพฯ.: สำนักพิมพ์ซีเอ็ดยูเคชั่น

ภาคผนวก ก.
การติดตั้งโปรแกรม

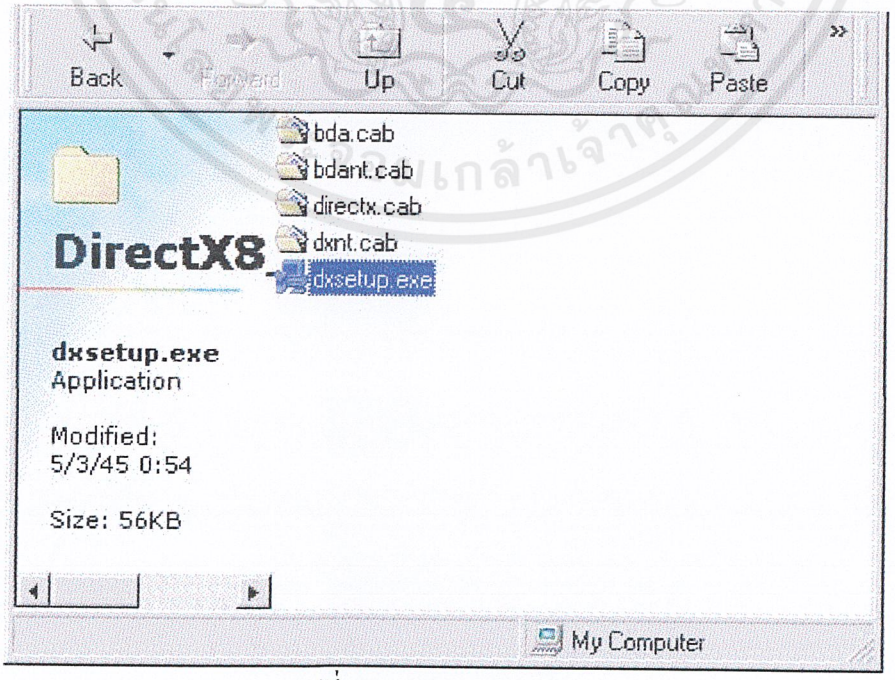
วิธีติดตั้ง DirectX Runtime

1. จาก folder Install เข้าไปใน folder Direct8_RunTime



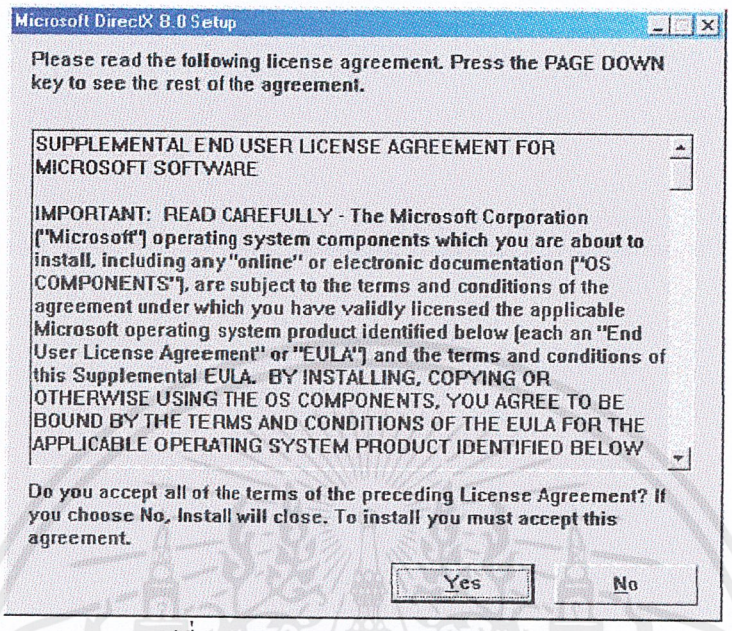
รูปที่ ก.1 folder Install

2. Double click ที่ icon dxsetup.exe



รูปที่ ก.2 icon dxsetup.exe

3. click ปุ่ม Yes

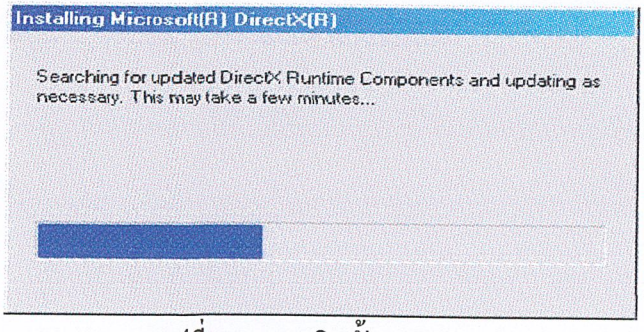


รูปที่ ก.3 DirectX License Agreement

4. click ปุ่ม Install DirectX

รูปที่ ก.4 ปุ่ม Install DirectX

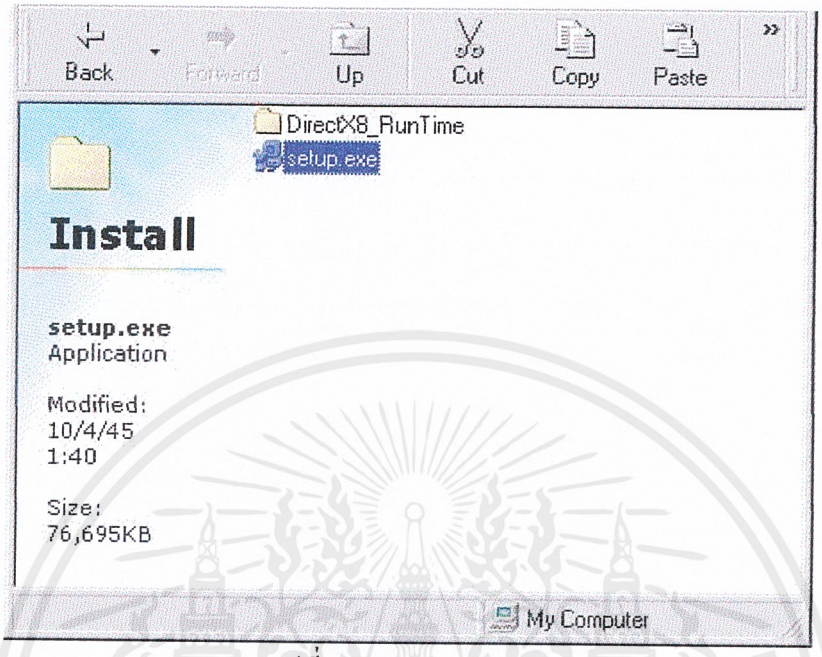
5. รอ ขณะกำลังติดตั้ง DirectX จากนั้นเครื่องจะ restart



รูปที่ ก.5 ขณะติดตั้ง DirectX

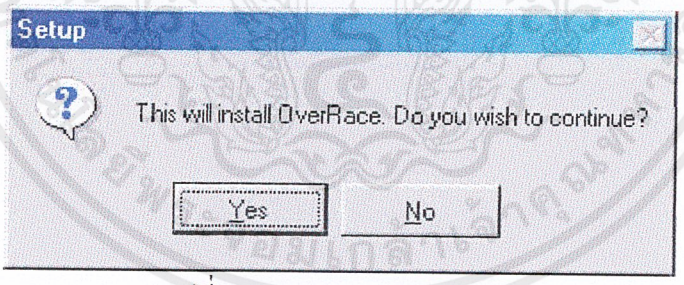
วิธีติดตั้ง Program

1. จาก folder Install double click ที่ icon setup.exe



รูปที่ ก.7 icon setup.exe

2. click ปุ่ม Yes

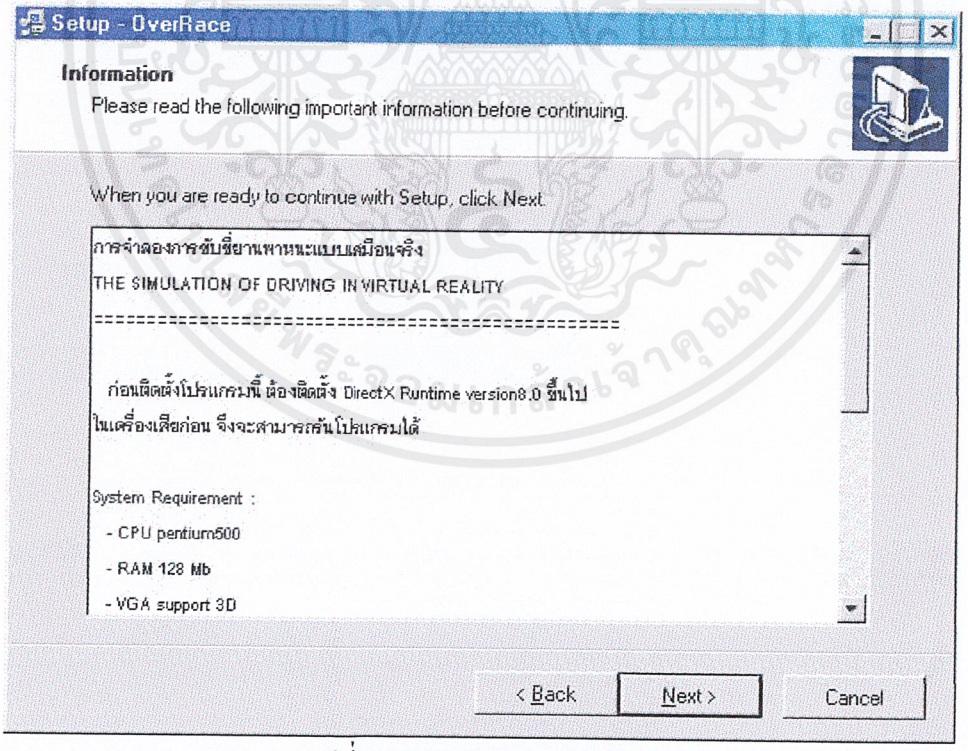


รูปที่ ก.8 setup confirm dialog

3. click ปุ่ม Next >

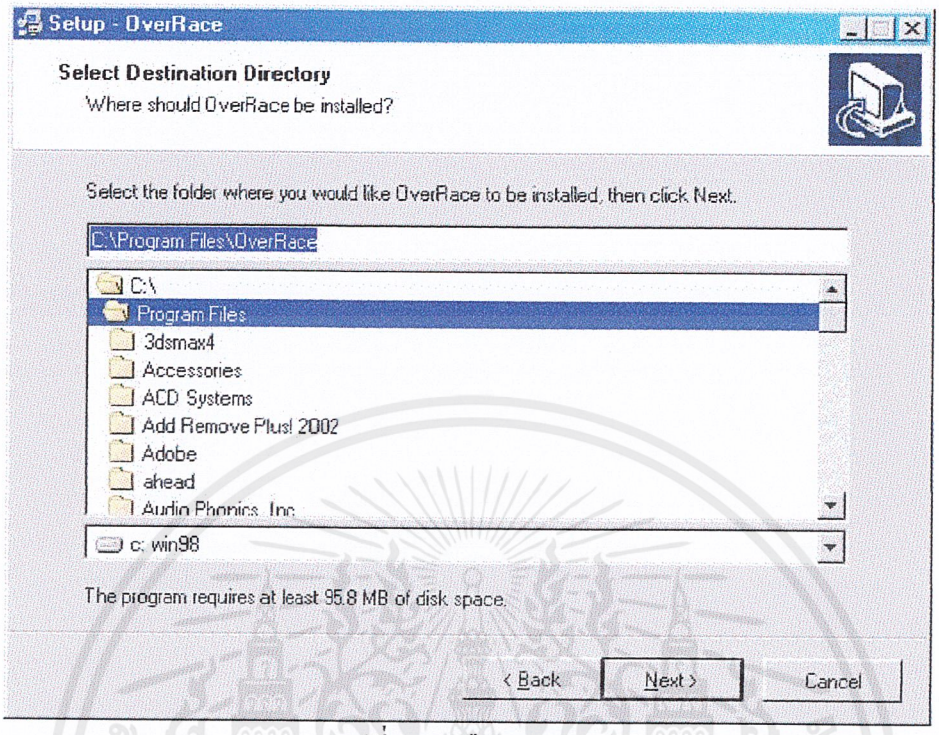
รูปที่ ก.9 Setup Wizard

4. click ปุ่ม Next >



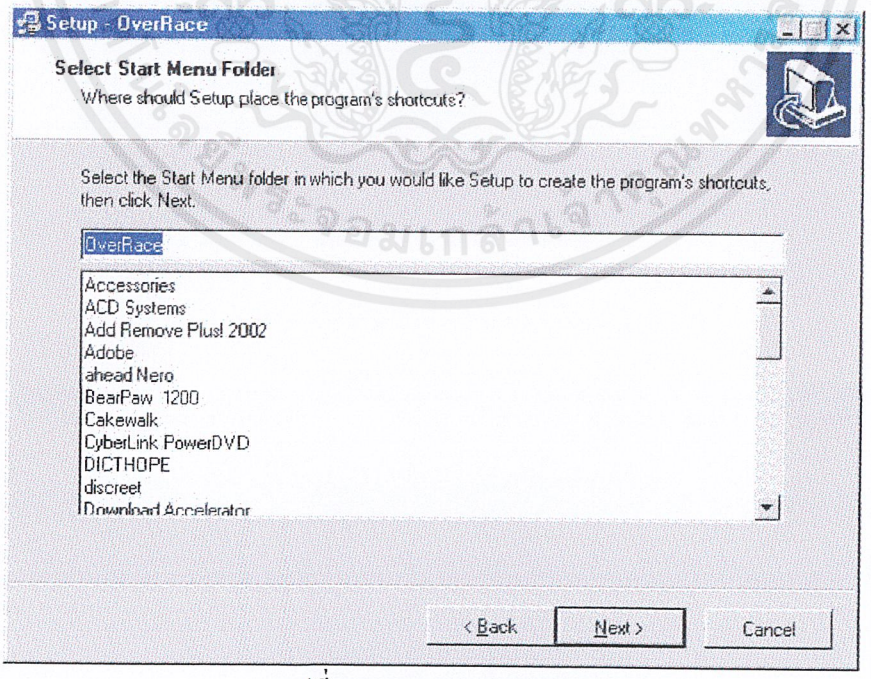
รูปที่ ก.10 Information dialog

5. เลือก folder ที่จะติดตั้ง หรือ คงเดิมไว้ แล้ว click ปุ่ม Next >



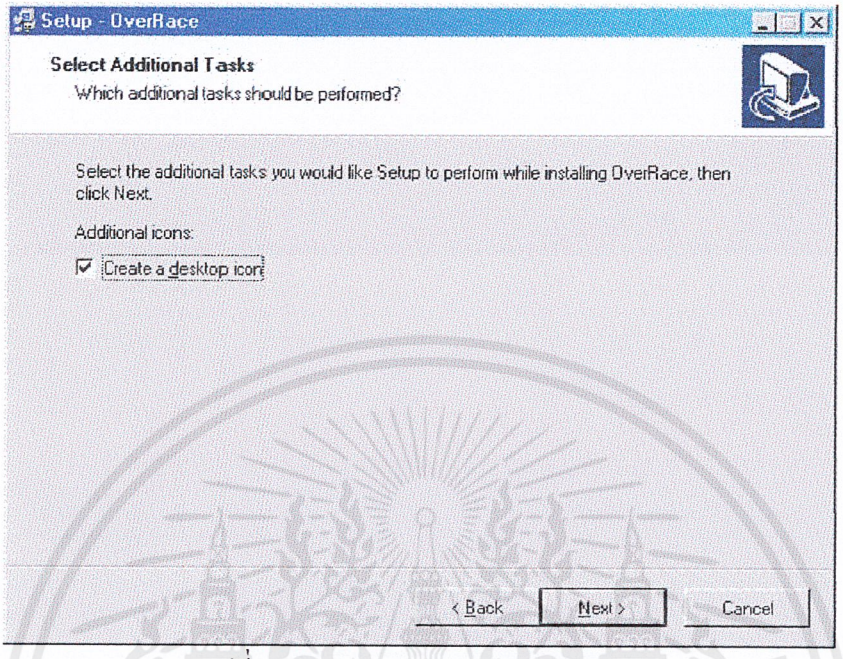
รูปที่ ก.11 เลือก folder

6. กำหนดชื่อและตำแหน่งของ Shortcut ใน Start Menu หรือคงเดิมไว้ แล้ว click ปุ่ม Next >



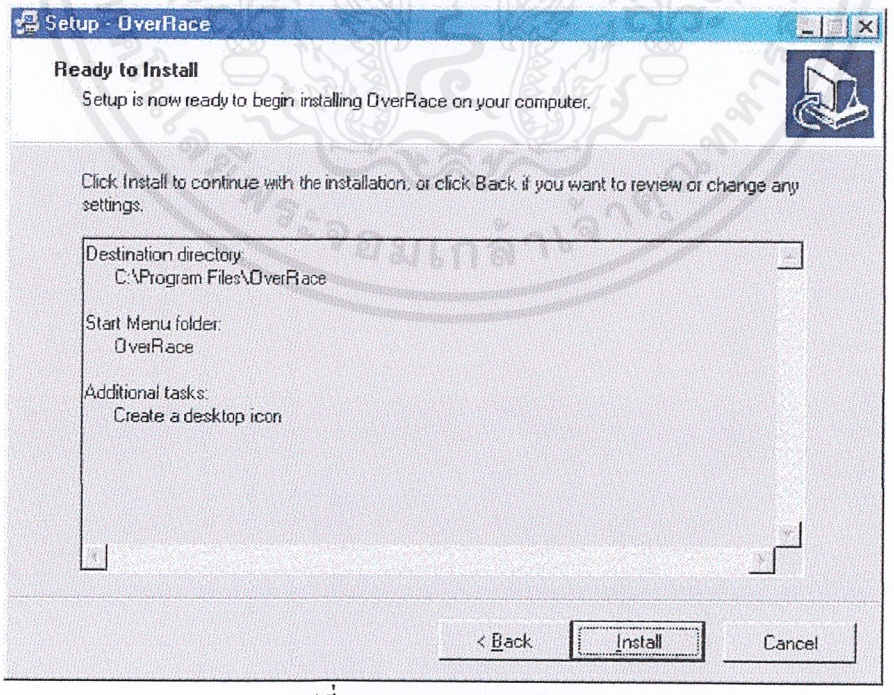
รูปที่ ก.12 shortcut dialog

7. checked ที่ช่อง Create a desktop icon ถ้าต้องการสร้าง Shortcut ไว้ที่ Desktop ถ้าไม่ต้องการให้ปล่อยว่างไว้ จากนั้น click ปุ่ม Next >



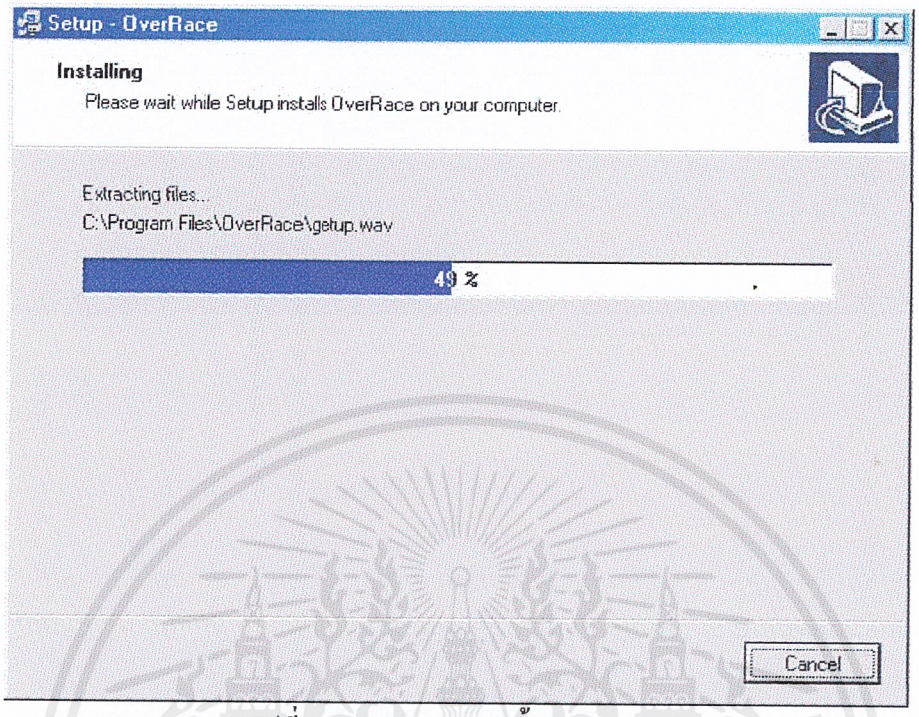
รูปที่ ก.13 Create desktop icon dialog

8. click ปุ่ม Install



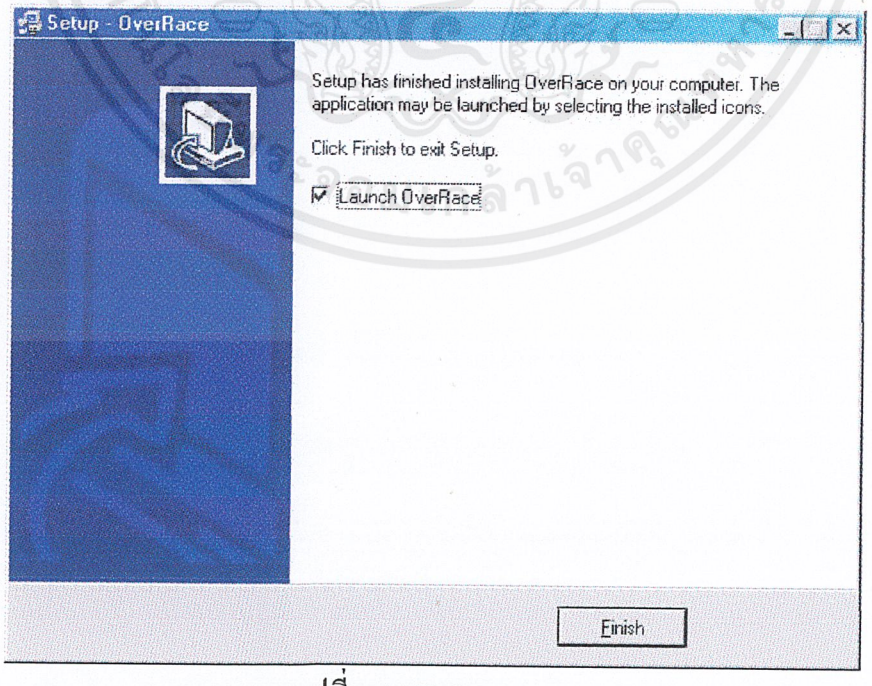
รูปที่ ก.14 Ready to Install

9. รอ ขณะ Program กำลังติดตั้ง หรือ click ปุ่ม Cancel เมื่อต้องการยกเลิกการติดตั้ง Program



รูปที่ ก.15 ขณะกำลังติดตั้ง Program

10. checked ที่ช่อง Launch OverRace ถ้าต้องการ run Program ทันทีที่ click ปุ่ม Finish ถ้ายังไม่ต้องการ run Program ให้ปล่อยว่างไว้ จากนั้น click ปุ่ม Finish



รูปที่ ก.16 Finish setup

ภาคผนวก ข.

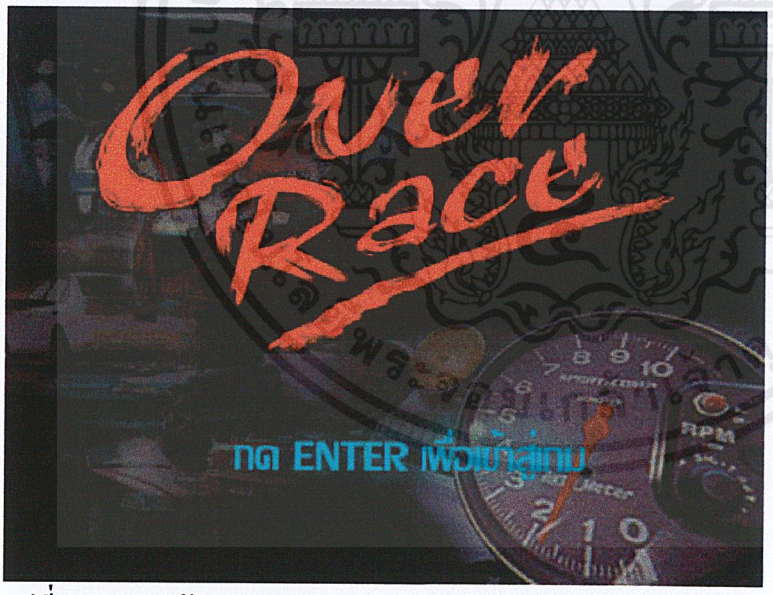
คู่มือการเล่นเกม

ประสิทธิภาพของระบบที่แนะนำ

1. ระบบปฏิบัติการ Microsoft Window 98
2. ติดตั้ง Microsoft DirectX เวอร์ชั่น 8.0 runtime
3. เครื่องคอมพิวเตอร์ที่มีซีพียูความเร็ว 600 MHz
4. หน่วยความจำหลักขนาด 128 Mbyte
5. อุปกรณ์อินพุทประเภทคีย์บอร์ด
6. การเร่งการแสดงผลภาพ 3 มิติ

ขั้นตอนการเข้าสู่เกม

1. เริ่มต้นเกมจะพบกับเมนูหน้าแรกของเกม ให้กดแป้น ENTER เพื่อที่จะไปสู่หน้าจอหลักของเกมดังรูปที่ ข.1



รูปที่ ข.1 เมนูหน้าแรกของเกม

2. เมื่อผ่านหน้าจอเมนูแรกของเกมแล้วจะเข้ามาสู่หน้าจอเมนูหลัก ในหน้าจอเมนูหลักนี้จะมีเมนูให้ผู้เล่นได้เลือก 3 เมนูดังนี้
   - เริ่มเกม (START GAME)
   - เกี่ยวกับโปรแกรม (ABOUT US)

- ออกจากโปรแกรม (EXIT TO WINDOWS)

หน้าจอเมนูหลักแสดงได้ดังรูปที่ ข.2

รูปที่ ข.2 หน้าจอเมนูหลักของเกม

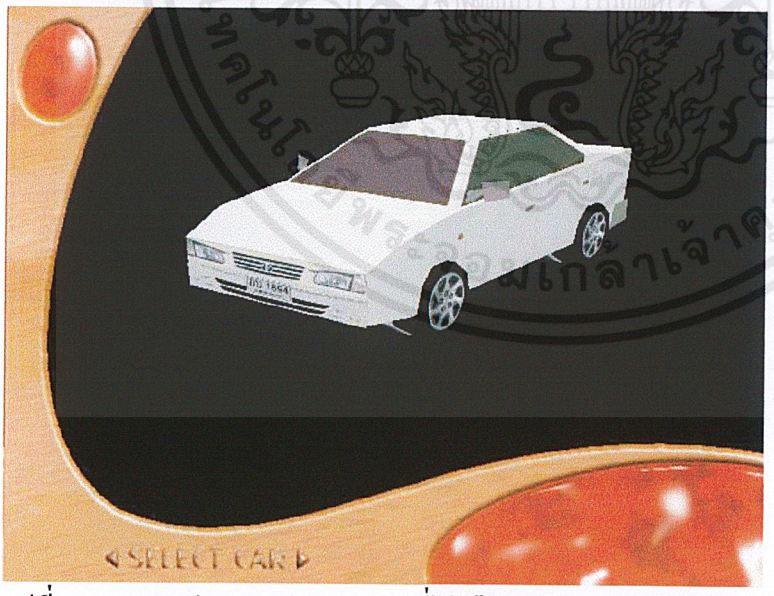
3. เมนูเกี่ยวกับโปรแกรม (ABOUT US) เป็นเมนูที่จะแสดงรายละเอียดเบื้องหลังการพัฒนาโปรแกรม โดยที่จะมีรายละเอียดดังรูปที่ ข.3

รูปที่ ข.3 เมนูเกี่ยวกับโปรแกรม (ABOUT US)

ผู้เล่นสามารถที่จะกลับไปหน้าจอเมนูหลักได้โดยกดที่แป้น ENTER หรือ ESC

4. เมนูออกจากโปรแกรม (EXIT TO WINDOWS) จะเป็นการออกจากเกม
5. เมื่อผู้เล่นต้องการจะเข้าสู่โหมดการเล่นต้องมาเลือกที่เมนูเริ่มเกม (START GAME) โดยเลือกที่เมนูเริ่มเกมแล้วกดแป้น ENTER จะเข้าสู่หน้าจอเมนูถัดไป
6. หลังจากเลือกเมนูเริ่มเกมแล้วก็จะเข้าสู่เมนูเลือกรถ คือผู้เล่นสามารถเลือกรถที่ใช้แข่งในเกมได้ตามความพอใจ ซึ่งภายในเมนูเลือกรถก็จะรถให้ผู้เล่นได้เลือก 2 รุ่นดังรูปที่ ข.4 และ รูปที่ ข.5

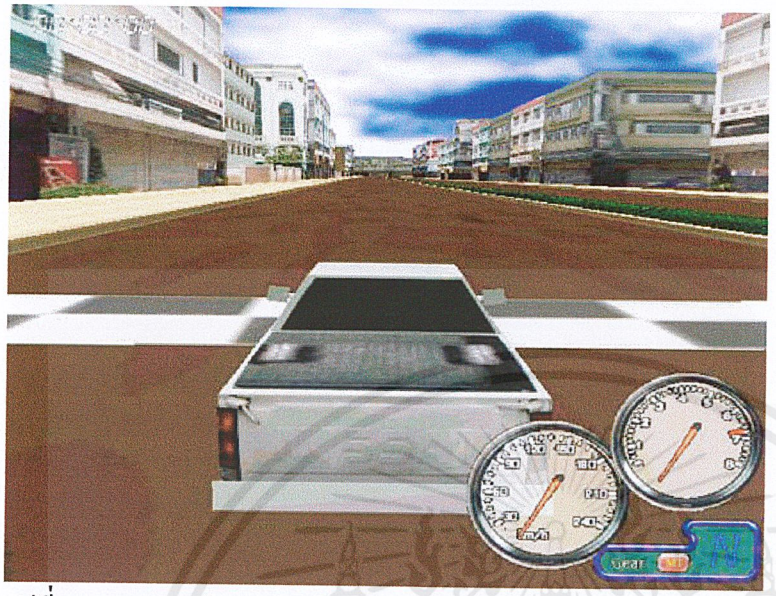
รูปที่ ข.4 แสดงรูปแบบของรถแบบ 1 ที่ให้เลือก



รูปที่ ข.5 แสดงรูปแบบของรถแบบ 2 ที่ให้เลือก

เมื่อผู้เล่นเลือกรถที่จะใช้แข่งได้แล้วก็ทำการกดแป้น ENTER หรือถ้าผู้เล่นต้องการที่จะกลับไปหน้าจอเมนูหลักก็สามารถทำได้โดยกดแป้น ESC

7. ถัดจากหน้าจอการเลือกรถก็จะเข้าสู่หน้าจอการแข่งขัน ส่วนของหน้าจอการแข่งขันจะมีลักษณะดังรูปที่ ข.6



รูปที่ ข.6 แสดงหน้าจอการแข่งขัน

ภายในหน้าจอการแข่งขันจะประกอบไปด้วย

- รถที่ผู้เล่นทำการเลือก
- ที่วัดความเร็วของรถ ซึ่งจะแสดงไว้ที่ด้านล่างขวาของจอภาพ
- ที่วัดรอบของรถ ซึ่งแสดงไว้คู่กับที่วัดความเร็วของรถ
- ส่วนของการแสดงสถานะของเกียร์รถที่ใช้ในปัจจุบัน
- ส่วนของการแสดงเวลาที่ใช้ในการแข่งขัน ซึ่งอยู่ด้านซ้ายบนของจอภาพ
- ฉากที่ใช้แข่งขัน

8. ขณะที่ผู้เล่นทำการเล่น สามารถที่จะหยุดเกมได้ชั่วคราว , กลับไปหน้าจอเมนูหลัก หรือออกจากเกมได้ โดยกดที่แป้น ESC จากนั้นเลือกเมนูย่อยที่ต้องการแล้วกดแป้น ENTER ดังรูปที่ ข.7

รูปที่ ข.7 แสดงเมนูย่อยขณะหยุดเกมชั่วคราว

9. เมื่อผู้เล่นบังคับรถจนถึงเส้นชัยก็เป็นอันว่าจบเกมโดยหน้าจอของการจบเกมจะเป็นดังรูปที่ ข.8

รูปที่ ข.8 หน้าจอแสดงการจบเกม

วิธีควบคุมรถ

ในการควบคุมรถจะใช้คีย์บอร์ดเป็นตัวควบคุมทั้งหมด โดยปุ่มที่ใช้ควบคุมมีดังนี้

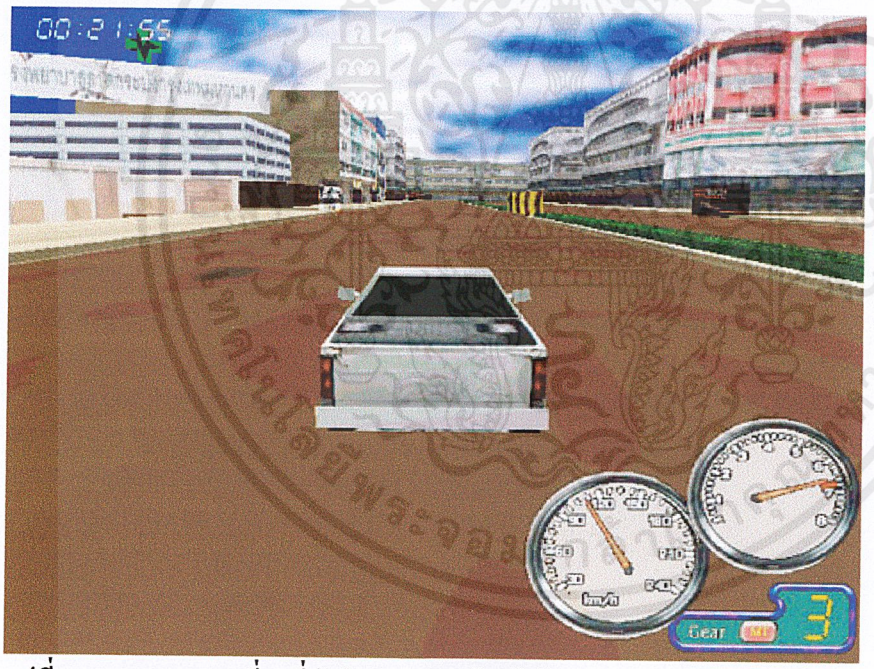
1. การควบคุมรถเลี้ยวซ้ายและขวา
   - แป้น ' → ' เป็นการบังคับรถให้เลี้ยวขวา
   - แป้น ' ← ' เป็นการบังคับรถให้เลี้ยวซ้าย

2. การเร่งความเร็วและการเบรก
   - แป้น ‘ Z ’ จะเป็นการเร่งความเร็วของรถ
   - แป้น ‘ X ’ จะเป็นการเบรกรถ
3. การเปลี่ยนเกียร์

   เมื่อเริ่มเกมนั้นเกียร์ของรถจะอยู่ที่ ‘ N ’ ซึ่งเป็นเกียร์ว่าง ลำดับของการเข้าเกียร์จะเป็นดังนี้

   $R \rightarrow N \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5$
   - แป้น ‘ ↑ ’ จะเป็นการเพิ่มเกียร์ เช่น $N \rightarrow 1, 1 \rightarrow 2$ เป็นต้น
   - แป้น ‘ ↓ ’ จะเป็นการลดเกียร์ เช่น $N \rightarrow R, 5 \rightarrow 4$ เป็นต้น
4. การเปลี่ยนมุมมองของรถ

   ในการเล่นผู้เล่นสามารถที่จะเลือกมุมมองในการขับขี่ได้ 2 แบบ ซึ่งแสดงไว้ดังรูปที่ ข.9 และรูปที่ ข.10



รูปที่ ข.9 มุมมองแบบที่ 1 ที่ผู้เล่นสามารถเลือกได้

รูปที่ ข.10 มุมมองแบบที่ 2 ที่ผู้เล่นสามารถเลือกได้

เงื่อนไขการจบเกม

เมื่อผู้เล่นบังคับรถจากจุดเริ่มต้นไปตามฉากจนถึงเส้นชัย