โปรแกรมจำลองสภาพจราจร

ROAD TRAFFIC SIMULATOR

โดย

นายจำนงค์ ระวัง

นายสุรัตน์ ผลประเสริฐ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมสารสนเทศ

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2547

ROAD TRAFFIC SIMULATOR

BY

Mr. CHAMNONG RAWANG

Mr. SURAT PHONPRASERT

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENT FOR THE DEGREE OF

BACHELOR IN DEPAARTMENT OF INFORMATION ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2004

ใบเสนอปริญญานิพนธ์

หัวข้อปริญญานิพนธ์	โปรแกรมจำลองสภาพการจราจร		
นักศึกษา	นายจำนงค์ ระวัง	รหัสประจำตัว	45015788
	นายสุรัตน์ ผลประเสริฐ	รหัสประจำตัว	45015828
อาจารย์ผู้ควบคุมปริญญานิพนธ์	ผศ. มยุรี เลิศเวชกุล		
ระดับการศึกษา	ปริญญาวิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมสารสนเทศ		
ภาควิชา	วิศวกรรมสารสนเทศ		
ปีการศึกษา	2547		

ปริญญานิพนธ์ฉบับนี้ได้รับความเห็นชอบจากอาจารย์ผู้ควบคุมปริญญานิพนธ์เป็นที่เรียบร้อยแล้ว

.....

(ผศ. มยุรี เลิศเวชกุล)

อาจารย์ผู้ควบคุมปริญญานิพนธ์

หัวข้อปริญญานิพนธ์	โปรแกรมจำลองสภาพการจราจร
นักศึกษา	นายจำนงค์ ระวัง รหัสประจำตัว 45015788
	นายสุรัตน์ ผลประเสริฐ รหัสประจำตัว 45015828
อาจารย์ผู้ควบคุมปริญญานิพนธ์	ผศ. มยุรี เลิศเวชกุล
ระดับการศึกษา	ปริญญาวิศวกรรมศาสตรบัณฑิต
	สาขาวิศวกรรมสารสนเทศ
ภาควิชา	วิศวกรรมสารสนเทศ
ปีการศึกษา	2547

บทคัดย่อ

โครงงานนี้เป็นการพัฒนาโปรแกรมจำลองสภาพการจราจร โดยโปรแกรมที่สร้างขึ้นจะเป็นการจำลองสภาพการจราจรภายใต้เงื่อนไขต่างๆ ที่สามารถกำหนดได้เช่น สภาพถนน ความกว้างของถนน การตีเส้นจราจรเป็นต้น โดยโปรแกรมสามารถแสดงผลการจำลองในลักษณะของภาพเคลื่อนไหวและแสดงผลของประสิทธิภาพในการจัดการจราจร ได้เช่น ความเร็วเฉลี่ยของถนนแต่ละเส้น เวลารอเฉลี่ยของแต่ละแยกไฟแดง เป็นต้น

โปรแกรมนี้ได้พัฒนาโดยใช้ภาษา JAVA ซึ่งเป็นภาษาเชิงวัตถุอีกภาษาหนึ่งที่มีประสิทธิภาพ และเลือกใช้ชุดโปรแกรมสำหรับพัฒนาโปรแกรม Java Developer Kit (JDK) เป็น Compiler และ Interpreter

Thesis Title	ROAD TRAFFIC SIMULATOR		
Student	Mr. Chamnong Rawang	ID.	45015788
	Mr. Surat Phonprasert	ID.	45015828
Advisor	Asst. Prof. Mayuree Lertwatechakul		
Graduate Level	Bachelor Degree of Information Engineering		
Department	Information Engineering		
Academic Year	2004		

ABSTRACT

This project is Road Traffic Simulate Programming Development subject to many condition, such state of road wide of road and the traffic line. It able show graphic user interface and efficiency of traffic for example Speed average and waiting time.

This Program develop by JAVA language. Once of Object language which good efficiency and we chose Java Developer Kit (JDK) to Compiler and Interpreter.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้คงไม่สามารถสำเร็จได้ด้วยดีหากไม่ได้รับความกรุณาจากหลาย ๆ ท่านขอขอบพระคุณ ผศ. มยุรี เลิศเวชกุล อาจารย์ที่ปรึกษาปริญญานิพนธ์ ที่ได้ให้ความดูแลเอาใจใส่ให้คำปรึกษาที่ดีตลอดมา ทั้งด้านการดำเนินงาน และการจัดทำปริญญานิพนธ์

ขอขอบพระคุณคณะครูอาจารย์ทุกท่านที่ช่วยประสิทธิประสาทวิชาความรู้ จนสามารถนำความรู้ที่ได้มาใช้จนสามารถสร้างสรรค์ผลงานได้สำเร็จ

ขอขอบคุณเพื่อน ๆ ที่พยามยามช่วยเหลือด้วยคำแนะนำและแนวคิดใหม่ ๆ เพิ่มเติม

สุดท้ายนี้ขอขอบพระคุณ คุณพ่อ คุณแม่อันเป็นที่เคารพรักยิ่ง ซึ่งท่านได้เลี้ยงดูอบรมสั่งสอนและส่งเสียให้ศึกษา จนสามารถสำเร็จการศึกษาได้ตามความปรารถนา

นายจำนงค์ ระวัง

นายสุรัตน์ ผลประเสริฐ

สารบัญ

สารบัญ (ต่อ)

สารบัญภาพ

รูป หน้า

สารบัญตาราง

ตาราง **หน้า**

บทที่ 1
บทนำ

1.1 แนวคิดและที่มาของปัญหา

ปัจจุบันปัญหาการจราจรเป็นปัญหาสำคัญของเมืองใหญ่ๆในประเทศไทยโดยเฉพาะกรุงเทพมหานคร ซึ่งเป็นปัญหาที่เรื้อรังมานาน ถึงแม้ว่ารัฐบาลจะทุ่มเทงบประมาณจำนวนมหาศาลเพื่อใช้ในการศึกษาแก้ปัญหา แต่การศึกษาแก้ปัญหาก็ที่ผ่านมาก็ยังไม่สามารถแก้ปัญหาได้อย่างเป็นรูปธรรม ผู้ดำเนินโครงงานจึงมีแนวคิดในการสร้างโปรแกรมจำลองสภาพการจราจร เพื่อใช้เป็นเครื่องมือช่วยอีกทางหนึ่งในการจำลองสภาพการจราจรขึ้นมาภายใต้เงื่อนไขต่าง ๆ ที่สามารถกำหนดได้ ทำให้เจ้าหน้าที่ที่รับผิดชอบมีความสะดวกมากขึ้นอันจะทำให้การแก้ปัญหาสภาพการจราจรเป็นรูปธรรมมากขึ้น

1.2 จุดประสงค์

1.2.1 เพื่อจำลองสภาพการจราจรภายใต้เงื่อนไขต่างๆ ที่สามารถกำหนดได้

1.2.2 เพื่อเป็นโปแกรมต้นแบบช่วยในการศึกษาหาแนวทางป้องกันและแก้ไขปัญหาสภาพการจราจร

1.2.3 เพื่อศึกษาระบบการจำลองปัญหา

1.2.4 เพื่อศึกษาการเขียนโปแกรมด้วยภาษาจาวา

1.3 ขอบเขตของโครงงาน

1.3.1 สามารถกำหนดแผนที่ถนนได้

1.3.2 ถนนอาจมีได้หลายเลนและมีข้อกำหนดในการเดินรถตามชนิดของเส้นทาง

1.3.3 มีสัญญาณไฟจราจรที่สามารถกำหนดระยะเวลาในการเปลี่ยนสัญญาณไฟได้

1.3.4 มีการกำหนดจุดออกรถและจุดหมายปลายทาง รถจะสามารถเคลื่อนย้ายจากจุดออกรถไปถึงจุดหมายปลายทางได้ โดยรถจะมีการกำหนดความเร็วสูงสุดที่สามารถวิ่งได้ รถจะวิ่งซ้อนตำแหน่งกันไม่ได้

1.3.5 มีสถานที่ที่สามารถกำหนดให้เป็นจุดออกรถและจุดหมายปลายทางได้ โดยมีขนาดของลานจอดรถที่จำกัดและไม่จำกัด เช่น 50 คัน 200 คัน เป็นต้น

1.4 ผลที่คาดว่าจะได้รับ

1.4.1 ได้โปรแกรมต้นแบบที่สามารถจำลองสภาพการจราจรภายใต้เงื่อนไขต่างๆที่กำหนดได้

1.4.2 ได้โปรแกรมต้นแบบที่สามารถช่วยในการศึกษาหาแนวทางป้องกันและแก้ไขปัญหาสภาพการจราจร

1.4.3 ได้ความรู้จากการศึกษาระบบการจำลองและสามารถนำไปประยุกต์ใช้กับปัญหาอื่นๆได้

1.4.4 ได้ทักษะจากการศึกษาการเขียนโปรแกรมด้วยภาษาจาวา

1.4.5 ได้โปรแกรมต้นแบบที่สามารถนำไปพัฒนาต่อให้มีประสิทธิภาพมากยิ่งขึ้น

1.5 ขั้นตอนการดำเนินงาน

มีขั้นตอนดังนี้

1.5.1 ศึกษาและค้นคว้าข้อมูล เป็นการศึกษาค้นคว้าเรื่องรูปแบบระบบการจำลองปัญหา เป็นการศึกษาหลักการที่ใช้ในการจำลองโดยการค้นคว้าจากหนังสือต่างๆ

1.5.2 วิเคราะห์และออกแบบรูปแบบการจำลองสภาพการจราจร

1.5.3 ศึกษาการเขียนโปรแกรม ส่วนใหญ่เป็นการศึกษาและค้นคว้าการเขียนโปรแกรมภาษา JAVA จากหนังสือและเว็บไซท์ต่างๆ

1.5.4 การออกแบบตัวโปรแกรม

1.5.5 ทดสอบและแก้ไขโปรแกรม

1.5.6 ทดสอบและแก้ไขระบบ

1.5.7 จัดทำเอกสารโครงงาน

ลำดับ	ขั้นตอน	2547							2548		
		มิ.ย.	ก.ค.	ส.ค.	ก.ย.	ต.ค.	พ.ย.	ธ.ค.	ม.ค.	ก.พ.	มี.ค.
1	ศึกษาและค้นคว้าข้อมูล										
2	วิเคราะห์และออกแบบ										
3	ศึกษาการเขียนโปรแกรม										
4	ออกแบบโปรแกรม										
5	ทดสอบและแก้ไขโปรแกรม										
6	ทดสอบและแก้ไขระบบ										
7	จัดทำเอกสารโครงงาน										

ตารางที่ 1.1 แสดงตารางลำดับการทำงาน

1.6 อุปกรณ์ที่ต้องใช้

1.6.1. ฮาร์ดแวร์

- เครื่องคอมพิวเตอร์สำหรับการพัฒนาโปแกรมจำนวน 1 เครื่อง
- เครื่องคอมพิวเตอร์สำหรับการพัฒนาโปรแกรมที่มีการเชื่อมต่อกับระบบเครือข่าย จำนวน 1 เครื่อง

1.6.2. ซอฟต์แวร์

- Java Developer Kit (JDK)

บทที่ 2
หลักการและทฤษฎีเบื้องต้น

2.1 หลักการจำลองเบื้องต้น

การจำลองแบบปัญหา (Simulation) คือกระบวนการออกแบบจำลอง (Model) ของระบบงานจริง (Real System) แล้วดำเนินการทดลองใช้แบบจำลองนั้น เพื่อการเรียนรู้พฤติกรรมของระบบงาน หรือเพื่อประเมินการใช้กุศโลบาย (Strategies) ต่าง ๆ ในการดำเนินงานของระบบภายใต้ข้อกำหนดที่วางไว้

จากคำจำกัดความดังกล่าวจะเห็นได้ว่า กระบวนการของการจำลองแบบปัญหานั้นแบ่งเป็นสองส่วน คือ การสร้างแบบจำลองส่วนหนึ่งและการนำแบบจำลองนั้นไปใช้งานเชิงวิเคราะห์อีกส่วนหนึ่ง ดังนั้นจะเห็นได้ว่ากลไกของวิธีการของการจำลองแบบปัญหาขึ้นอยู่กับแบบจำลองและการใช้ แบบจำลองที่ใช้ในการจำลองแบบปัญหานี้อาจเป็นหุ่นเป็นระบบหรือเป็นแนวความคิดลักษณะหนึ่งลักษณะใด โดยไม่จำเป็นต้องเหมือน (Identical) กับระบบงานจริง แต่ต้องสามารถช่วยให้เข้าใจระบบงานจริงเพื่อประโยชน์ในการอธิบายพฤติกรรมและเพื่อการปรับปรุงการดำเนินงานของระบบงานจริง

2.1.1 ระบบงาน

กลไกสำคัญอันหนึ่งในการจำลองแบบปัญหาอยู่ที่แบบจำลองการที่จะสามารถสร้างแบบจำลองที่ใช้ในการจำลองแบบปัญหาได้ ผู้สร้างต้องมีความเข้าใจในระบบงานจริงเป็นอย่างดี ความรู้ความเข้าใจในระบบงานจริงเป็นหัวใจสำคัญของการสร้างและใช้งานแบบจำลอง

ระบบงาน หมายถึง กลุ่มขององค์ประกอบ (Elements) ที่มีความสัมพันธ์กัน โดยที่ความหมายของระบบงานบอก เฉพาะลักษณะว่าระบบงานจะมีลักษณะอย่างไรโดยไม่ได้บอกรูปร่างหน้าตาที่แน่ชัด ดังนั้นเมื่อเวลาที่ต้องการจะศึกษาระบบงานใดระบบงานหนึ่ง จึงจำเป็นที่จะต้องบอกรูปร่างหน้าตาที่ชัดเจนของระบบงานที่กำลังศึกษา การบอกรูปร่างหน้าตาที่ชัดเจนของระบบงานมักบอกโดยการกำหนดขอบเขตของระบบงาน (System Boundaries) ซึ่งก็คือ การกำหนดองค์ประกอบของระบบ การแสดงความสัมพันธ์ระหว่างองค์ประกอบ และการกำหนดองค์ประกอบอื่น ๆ ที่อยู่นอกระบบแต่มีผลกระทบต่อการทำงานของระบบ องค์ประกอบอื่น ๆ ที่อยู่นอกระบบนี้ เรียกโดยรวมว่า สิ่งแวดล้อมของระบบ (System Environment) องค์ประกอบต่าง ๆ ทั้งภายในและภายนอกระบบงานจะมีลักษณะเฉพาะตัว

(Attributes) ที่ทำให้เกิดกิจกรรม (Activities) และกิจกรรมเหล่านั้นภายใต้เงื่อนไขบางประการจะก่อให้เกิดการเปลี่ยนแปลงสถานภาพของระบบงาน (System Status) ดังนั้นนอกจากการกำหนดขอบเขตของระบบงานแล้วยังต้องกำหนดลักษณะเฉพาะตัวขององค์ประกอบ กิจกรรมที่เกิดขึ้นจากองค์ประกอบเหล่านั้น และการเปลี่ยนแปลงสถานภาพของระบบอันเนื่องมาจากกิจกรรมขององค์ประกอบ

ประเภทของระบบงาน

การจำลองระบบงานอาจจำแนกได้หลายแบบแล้วแต่การนำไปใช้งานในการจำลองแบบปัญหา การจำแนกระบบงานเพื่อความสะดวกในการใช้งานนั้นมักจะจำแนกโดยอาศัยลักษณะการเปลี่ยนแปลงของระบบเป็น 4 ประเภทดังนี้

ระบบต่อเนื่องหรือระบบเป็นช่วง (Continuous versus Discrete Systems)

โดยพิจารณาจากพฤติกรรมในการเปลี่ยนแปลงสภาวะของระบบเทียบกับเวลา ถ้าการเปลี่ยนสภาวะของระบบเป็นการเปลี่ยนไปตามเวลาอย่างต่อเนื่อง ระบบงานนั้นก็จะเป็นระบบต่อเนื่อง แต่ถ้าการเปลี่ยนสภาวะของระบบเกิดขึ้นที่ช่วงเวลาใดเวลาหนึ่ง ไม่ต่อเนื่อง ระบบงานนั้นก็จะเป็นระบบเป็นช่วง หรือระบบไม่ต่อเนื่อง

ระบบตายตัวหรือระบบไม่แน่นอน (Deterministic versus Stochastic System)

ระบบตายตัว หมายถึง ระบบซึ่งการเปลี่ยนแปลงสภาวะที่ระดับใหม่สามารถบอกได้จากสภาวะและกิจกรรมของระบบที่ระดับก่อน ส่วนระบบไม่แน่นอน หมายถึง ระบบซึ่งการเปลี่ยนสภาวะเป็นแบบสุ่ม และในบางกรณีก็สามารถหาค่าความน่าจะเป็น (Probability) ของการเปลี่ยนสภาวะ

2.1.2 แบบจำลอง

แบบจำลองหมายถึง ตัวแทนวัตถุ ระบบ หรือแนวคิดลักษณะใดลักษณะหนึ่ง แบบจำลองอาจนำไปใช้งานหลายลักษณะดังนี้

1. เป็นเครื่องช่วยคิด (An aid to thought) เช่น แบบจำลองโครงข่าย (Network Model) ช่วยทำให้ผู้สร้างแบบจำลองได้มองเห็นว่าจะมีกิจกรรมที่ต้องทำอะไรบ้างและทำอะไรก่อนอะไรหลัง
2. เป็นเครื่องสื่อความหมาย (An aid to communication) แบบจะลองจะช่วยให้เข้าใจพฤติกรรมของระบบงานและช่วยให้สามารถอธิบายพฤติกรรม ปัญหาและการแก้ปัญหาของระบบ

3. เป็นเครื่องช่วยสอบและฝึกอบรม (Purposes of training and instruction) เช่น แบบจำลองเครื่องควบคุมการบินจะช่วยให้นักบินทำความเข้าใจและความคุ้นเคย กับระบบการควบคุมเครื่องบินจริงก่อนขึ้นฝึกบินจริง
4. เป็นเครื่องมือสำหรับการทำนาย (A tool of prediction) จากการที่แบบจำลองจะช่วยให้เข้าใจพฤติกรรมของระบบงาน ก็จะช่วยให้ผู้สร้างแบบจำลองสามารถคาดคะเนหรือทำนายได้ว่า เมื่อมีเหตุการณ์ที่มีผลกระทบต่อองค์ประกอบของระบบเกิดขึ้น จะมีผลอะไรเกิดขึ้นกับระบบ
5. เป็นเครื่องมือสำหรับการทดลอง (An aid to experimentation) โดยที่แบบจำลองเป็นสิ่งที่สร้างขึ้นแทนระบบงานจริง ในกรณีที่ต้องการทดลองเงื่อนไขต่าง ๆ กับระบบงานจริงแต่ทำไม่ได้ ก็จำนำเอาเงื่อนไขนั้น ๆ มาทดลอง กับ แบบจำลองเพื่อจะดูว่าจะเป็นผลอย่างไร เพื่อประโยชน์ในการตัดสินใจว่าควรจะนำเงื่อนไขนั้น ๆ ไปใช้กับระบบงานจริงหรือไม่

ประเภทของแบบจำลองในการจำลองแบบปัญหา (Classification of Simulation Models)

ประเภทของแบบจำลองในการจำลองแบบปัญหา นอกจากจะสามารถจำแนกได้ตามประเภทของระบบงานที่มันเป็นตัวแทนอยู่แล้ว ยังมีลักษณะพิเศษเฉพาะตัวของแบบจำลองซึ่งทำให้มันสามารถจำแนกประเภทออกไปตามคุณลักษณะพิเศษดังนี้

1. แบบจำลองทางกายภาพ (Physical of Iconic Models) เป็นแบบจำลองที่มีรูปร่างหน้าตาเหมือนระบบงานจริง อาจมีขนาดเท่ากับระบบงานจริงหรือมีขนาดเล็กกว่าหรือใหญ่กว่า (Scaled Models) อาจเป็นแบบจำลองของระบบงานจริงในมิติใดมิติหนึ่ง (Dimension) หรือทั้งสามมิติ ตัวอย่างของแบบจำลองประเภทนี้ได้แก่ เครื่องยนต์ต้นแบบ (Prototype) ซึ่งสร้างขึ้นเพื่อทดสอบสมรรถนะก่อนการสร้างจริง แบบจำลองของส่วนควบคุมการบินของเครื่องบิน

2. แบบจำลองอนาล็อก (Analog Models) เป็นแบบจำลองที่มีพฤติกรรมเหมือนระบบงานจริง ตัวอย่างของแบบจำลองประเภทนี้ได้แก่ อนาล็อกคอมพิวเตอร์ที่ใช้ควบคุมการผลิตในอุตสาหกรรมอาหารสัตว์ และอุตสาหกรรมเคมี ซึ่งใช้การเคลื่อนที่ของกระแสไฟฟ้าซึ่งแสดงบนแผงควบคุมบอกให้รู้ถึงการเคลื่อนที่ ของวัตถุในระบบงานจริง การใช้กราฟแทนความสัมพันธ์ของสิ่งต่าง ๆ ที่วัดค่าได้ เช่น ๆ ความสัมพันธ์ระหว่างค่าใช้จ่ายใจการผลิต กับจำนวนสินค้าที่ผลิต ซึ่งเป็นแบบจำลองที่ใช้ขนาดความยาวของเส้นกราฟแสดงค่าของเงินหรือจำนวนสินค้า

3. เกมการบริหาร (Management Games) เป็นแบบจำลองการตัดสินใจ (Decision Models) ในกิจการต่าง ๆ เช่น ธุรกิจ สงคราม การลงทุน ฯลฯ เป็นแบบจำลองที่ใช้แสดงผลถ้ามีการตัดสินใจแบบต่าง ๆ เพื่อใช้เป็นข้อมูลสำหรับการตัดสินใจ

4. แบบจำลองทางคอมพิวเตอร์ (Computer Simulation Models) เป็นแบบจำลองที่อยู่ในรูปแบบของโปรแกรม ซึ่งก่อนที่จะมาเป็นคอมพิวเตอร์โปรแกรม แบบจำลองอาจอยู่ในรูปของแบบจำลองประเภทหนึ่งประเภทใดที่กล่าวมาแล้วทั้งหมด

5. แบบจำลองทางคณิตศาสตร์ (Mathematical Models) เป็นแบบจำลองที่ใช้สัญลักษณ์และฟังก์ชันทางคณิตศาสตร์แทนองค์ประกอบในระบบงานจริง เช่น ใช้ X แทนค่าใช้จ่ายในการผลิต Y แทนจำนวนสินค้าที่ผลิต

ในระบบงานจริงที่มีความยุ่งยากซับซ้อนแบบจำลองระบบงานอาจใช้แบบจำลองหลายประเภทร่วมกัน

โครงสร้างของแบบจำลอง (Structure of Simulation Model)

โครงสร้างของแบบจำลองอาจะเขียนเป็นรูปแบบแสดงความสัมพันธ์ทางคณิตศาสตร์ได้เป็น

$$E = f(x_i, y_i)$$

โดยที่ E คือ ผลของการปฏิบัติงานของระบบ

x_i คือ ตัวแปรและพารามิเตอร์ที่เราสามารถควบคุมได้

y_i คือ ตัวแปร และพารามิเตอร์ที่เราไม่สามารถควบคุมได้

f คือ ความสัมพันธ์ระหว่าง x_i และ y_i ที่ทำให้เกิด E

รูปแบบของความสัมพันธ์ดังกล่าวแสดงให้เราเห็นว่าสมรรถนะของระบบนั้น เป็นผลกระทบเนื่องมาจากตัวแปรต่าง ๆ ทั้งที่อยู่และไม่อยู่ในความควบคุมของเรา และโดยที่ระบุทุกระบบที่ทำการศึกษาจะต้องมีขอบเขตจำกัดอีกทั้งต้องมีวัตถุประสงค์ของการศึกษา เมื่อรวมเข้ากับรูปแบบของความสัมพันธ์ข้างต้น จะเห็นได้ว่าโครงสร้างของแบบจำลองนั้นควรประกอบไปด้วย

1. องค์ประกอบ (Components) ในทุกระบบงานจะประกอบไปด้วยองค์ประกอบต่าง ๆ ในแบบจำลองที่ใช้แทนระบบงานก็จะต้องประกอบไปด้วยองค์ประกอบที่จำเป็นสำหรับการทำงานของระบบงาน

2. ตัวแปรและพารามิเตอร์ (Variables and Parameters) พารามิเตอร์ คือค่าคงที่ซึ่งผู้ใช้แบบจำลองเป็นผู้กำหนดให้ อาจเป็นค่าที่กำหนดขึ้นเองเพื่อศึกษาผลที่เกิดขึ้นจากค่าของพารามิเตอร์นั้น หรือเป็นค่าที่วัดหรือประเมินได้จากข้อมูล ส่วนตัวแปรนั้นเป็นค่าที่แปรผัน มีค่าได้หลายค่าตามสภาวะจริงของการใช้งาน จำแนกได้เป็นสองประเภท คือ ตัวแปรจากภายนอก (Exogenous Variable) หรือตัวแปรนำเข้า (Input Variables) หมายถึงตัวแปรจากภายนอกระบบซึ่งเข้ามามีผลกระทบต่อสมรรถนะของระบบหรือเป็นตัวแปรที่เป็นผลเนื่องมาจากปัจจัยภายนอกระบบ และตัวแปรภายใน (Endogenous Variables) หมายถึงตัวแปรที่เกิดขึ้นภายในระบบ ตัวแปรภายในอาจอยู่ในลักษณะตัวแปรสถานภาพ (Status Variables) ซึ่งเป็นตัวแปรที่ใช้บอกสถานภาพหรือเงื่อนไขของระบบ หรืออยู่ในลักษณะของตัวแปรนำออก (Output Variables) ซึ่งก็คือผลที่ได้จากการใช้งานระบบ ในทางสถิติ ตัวแปรจากภายนอกคือตัวแปรอิสระ (Independent Variables) และตัวแปรภายในคือตัวแปรตาม (Dependent Variables)

3. ฟังก์ชันความสัมพันธ์ (Functional Relationships) คือฟังก์ชันที่ใช้อธิบายความสัมพันธ์ระหว่างตัวแปรกับพารามิเตอร์ ฟังก์ชันความสัมพันธ์นี้อาจอยู่ในลักษณะที่แน่นอนตายตัว (Deterministic) ซึ่งเป็นลักษณะที่เมื่อใส่ข้อมูลนำเข้าจะสามารถหาได้ว่าผลลัพธ์จะเป็นเท่าไหร่แน่นอน และอาจอยู่ในลักษณะไม่แน่นอน (Stochastic) ซึ่งเมื่อใส่ข้อมูลนำเข้าให้ฟังก์ชันไม่แน่ว่าจะได้ผลลัพธ์ออกมาเท่าไหร่ ลักษณะของฟังก์ชันความสัมพันธ์จะอยู่ในรูปของสมการคณิตศาสตร์ เช่น $Y = 4+0.7X$ ซึ่งฟังก์ชันความสัมพันธ์เหล่านี้อาจหามาได้จากสมมติฐานหรือประเมินข้อมูลร่วมกับวิธีทางสถิติหรือคณิตศาสตร์

4. ขอบข่ายจำกัด (Constraints) คือข้อจำกัดของค่าตัวแปรต่าง ๆ ซึ่งอาจเป็นข้อจำกัดที่ผู้ใช้แบบจำลองเป็นผู้กำหนด เช่น ข้อจำกัดของทรัพยากรต่าง ๆ ที่มีอยู่ในระบบ

5. ฟังก์ชันเป้าหมาย (Criterion Function) หมายถึง ข้อความ (Statement) ที่บอกเป้าหมาย (Goals) หรือวัตถุประสงค์ (Objectives) ของระบบงาน และวิธีประเมินผลตามเป้าหมาย วัตถุประสงค์ของระบบงานอาจแบ่งได้เป็นสองประเภท คือ การคงสภาพของระบบงาน (Retentive) ซึ่งเป็นวัตถุประสงค์ที่จะทำให้ระบบสามารถคงสภาพการใช้ทรัพยากร เช่น เวลา พลังงาน ความชำนาญ ฯลฯ หรือคงสถานภาพของระบบ

2.1.3 กระบวนการจำลองปัญหา

ขั้นตอนต่าง ๆ ต่อไปนี้เป็นขั้นตอนการจำลองแบบปัญหาที่ใช้คอมพิวเตอร์คำนวณ

1. การตั้งปัญหาและการให้คำจำกัดความของระบบงาน (Problem Formulation and System Definition) ขั้นตอนนี้เป็นขั้นตอนที่สำคัญที่สุดของการจำลองแบบปัญหาขั้นตอนนี้เป็นการกำหนดวัตถุประสงค์ของการศึกษาระบบ การกำหนดขอบเขต ข้อจำกัดต่าง ๆ และวิธีการวัดผลของระบบ

2. การสร้างแบบจำลอง (Model Formulation) จากลักษณะของระบบงานที่ต้องทำการศึกษา เขียนแบบจำลองที่สามารถอธิบายพฤติกรรมของระบบงานตามวัตถุประสงค์ของการศึกษา

3. การจัดเตรียมข้อมูล (Data Prepation) วิเคราะห์หาข้อมูลต่าง ๆ ที่จำเป็นสำหรับแบบจำลอง และจัดเตรียมให้อยู่ในรูปแบบที่จะนำไปใช้งานกับแบบจำลองได้

4. การแปรรูปแบบจำลอง (Model Translation) แปลงแบบจำลองไปอยู่ในรูปแบบของโปรแกรมคอมพิวเตอร์

5. การทดสอบความถูกต้อง (Validation) เป็นการวิเคราะห์เพื่อช่วยให้ผู้เขียนและผู้ใช้แบบจำลองมั่นใจว่าแบบจำลองที่ได้นั้นสามารถใช้แทนระบบงานจริงตามวัตถุประสงค์ของการศึกษาได้

6. การออกแบบการทดลอง (Strategic Planning) เป็นการออกแบบการทดลองที่ทำให้แบบจำลองสามารถให้ข้อมูลที่ใช้ในการวิเคราะห์หาผลลัพธ์ตามที่ต้องการ

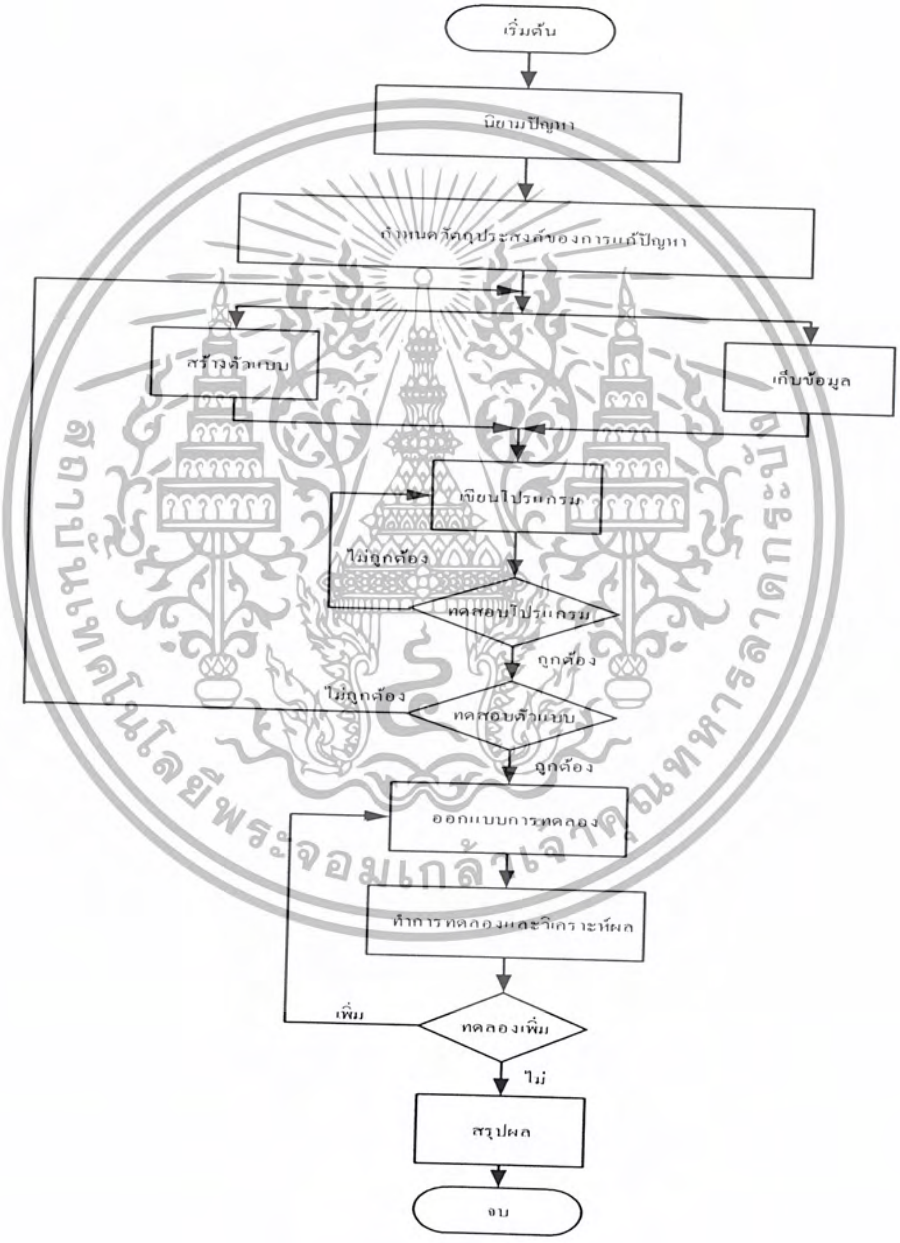
7. การวางแผนการใช้งานแบบจำลอง (Tactical Planning) เป็นการวางแผนว่าจะใช้งานแบบจำลองในการทดลองทำอะไร จึงจะได้ข้อมูลสำหรับวิเคราะห์ผลเพียงพอ (ด้วยระดับความเชื่อมั่นในผลการวิเคราะห์ที่เหมาะสม) ความแตกต่างระหว่างขั้นตอนนี้กับขั้นตอนการออกแบบการทดลองมีอยู่ว่าในการออกแบบการทดลองเป็นเพียงแต่การบอกเงื่อนไขของการทดลอง ส่วนขั้นตอนนี้เป็นการบอกว่าต้องดำเนินการทดลองตามเงื่อนไขดังกล่าวกี่ครั้งจึงจะได้จำนวนข้อมูลตามที่เหมาะสมกล่าวคือได้ความมีนัยสำคัญทางสถิติที่ยอมรับได้ในราคาที่เหมาะสม

8. การดำเนินการทดลอง (Experimentation) เป็นการคำนวณหาข้อมูลต่าง ๆ ที่ต้องการและความไวของการเปลี่ยนแปลงข้อมูลจากแบบจำลอง

9. การตีความผลการทดลอง (Interpretation) จากการทดลอง เลือกวิธีการที่จะแก้ปัญหาได้ดีที่สุดไปใช้กับระบบงานจริง

10 . การนำไปใช้งาน (Implementation) จากผลการทดลอง เลือกวิธีการที่จะแก้ปัญหาได้ดีที่สุดไปใช้กับระบบงานจริง

11. การจัดทำเป็นเอกสารการใช้งาน (Documentation) เป็นการบันทึกข้อความในการจัดทำแบบจำลอง โครงสร้างของแบบจำลอง วิธีการใช้งานและผลที่ได้จากการใช้งาน เพื่อประโยชน์สำหรับผู้ที่จะนำไปใช้งาน และเพื่อประโยชน์ในการปรับปรุงดัดแปลงแบบจำลองเมื่อเกิดการเปลี่ยนแปลงระบบ ฯลฯ ดังแสดงในรูปที่ 1



รูปที่ 2.1 แสดงขั้นตอนการจำลอง

2.2 หลักการโปรแกรมเชิงวัตถุเบื้องต้น

วัตถุทั่วไปในธรรมชาติจะเก็บสถานะ (State) เป็นข้อมูลของตัวเองได้ และต้องทำกิจกรรมบางอย่างได้ เช่นการคำนวณและเปลี่ยนข้อมูลของตัวเองหรือทำการติดต่อโต้ตอบกับวัตถุอื่น หากเราสมมุติว่านักเรียนคนหนึ่งเป็นวัตถุ เขาก็ควรจะมีชื่อ มีที่อยู่ มีผลการเรียนเป็นข้อมูลของตัวเอง และเขาน่าจะสามารถตอบคำถามอย่างเช่น ชื่ออะไร ผลการเรียนอยู่ระดับใด หรือเมื่อมีการประกาศผลการสอบ เขาก็น่าจะสามารถจดจำผลสอบได้ เป็นต้น ตัวแปรในภาษา Imperative ไม่สามารถจำลองพฤติกรรมและการทำงานของวัตถุอย่างนี้ ภาษาเชิงวัตถุจึงเสนอคลาส (Class) ขึ้นเป็นกลไกสำหรับกำหนดโครงสร้างและพฤติกรรมของวัตถุ เมื่อต้องการจำลองการทำงานของวัตถุนั้น เราจะต้องสร้างสิ่งที่เรียกว่า Instance ของคลาสนั้น ซึ่งจะมีพื้นที่หน่วยความจำสำหรับเก็บข้อมูลของวัตถุ และมี Methods สำหรับกระทำต่อข้อมูลของมันเองหรือโต้ตอบแลกเปลี่ยนข้อมูลกับ Instances อื่น อาจกล่าวได้ว่า Instances เป็นเหมือนกับตัวแปรที่พื้นที่เก็บข้อมูลได้ พร้อมกับมี Operations ที่เกี่ยวข้องกับหน้าที่ของมันด้วย แต่ Instances มีช่วงชีวิตต่างจากตัวแปรในภาษา Imperative

คลาสคือโครงสร้างและพฤติกรรมของวัตถุประเภทหนึ่ง หากเปรียบ Instances ในภาษาเชิงวัตถุก็เป็นเหมือนกับตัวแปรในภาษา Imperative แล้วคลาสก็จะเป็นเหมือนชนิดของข้อมูล (Type) นั้นเอง คลาสถูกสร้างขึ้นเพื่อกำหนดโครงสร้างของ Instances ของวัตถุประเภทหนึ่ง เราจึงต้องมีคลาสก่อนที่จะทำการสร้าง Instances ของคลาสนั้น และเมื่อมีคลาสหนึ่งแล้วเราจะสร้าง Instances ของคลาสนั้นขึ้นกี่ Instances ก็ได้ โดยที่ Instances ของคลาสเดียวกันจะมีโครงสร้างเหมือนกันแต่อาจมีข้อมูลที่ต่างกันไป ในการกำหนดคลาส เราต้องระบุว่าคลาสนั้นมีสมาชิก (Member) ใดบ้าง สมาชิกของคลาสมีสองประเภทคือ

1. สมาชิกที่เป็นข้อมูลเรียกว่า Data members หรือ Fields อาจเป็นค่าคงที่ ตัวแปรของข้อมูลชนิดพื้นฐาน Array หรือแม้แต่ Instances ของคลาส
2. สมาชิกที่เป็นฟังก์ชันเรียกว่า Method members หรือเรียกสั้นๆว่า Methods ซึ่งอาจเป็นได้ทั้ง Procedures (ฟังก์ชันที่ไม่ส่งค่าออกมา) หรือ Functions (ฟังก์ชันที่ส่งค่าออกมา)

ภาษา JAVA ใช้คลาสในการทำ Encapsulation ส่วนการทำ Information hiding นั้นมี Modifiers สำหรับกำหนดให้สมาชิกของคลาสถูกอ้างถึงจากภายนอกคลาสได้หรือไม่ Modifiers มีดังต่อไปนี้

- Private สมาชิกจะถูกอ้างถึงได้จากภายในคลาสเท่านั้น

- Public สมาชิกจะถูกอ้างถึงได้จากคลาสใดๆ ที่มองเห็นคลาสนั้นได้
- Protected สมาชิกจะถูกอ้างถึงได้จากคลาสใดๆ ใน Package เดียวกัน หรือคลาสที่ขยายออกจากคลาสนั้น
- Modifier เรียก (Default) สมาชิกจะถูกอ้างถึงได้จากคลาสใดๆ ใน Package เดียวกัน

2.3 ภาษา JAVA เบื้องต้น

ภาษา JAVA เริ่มเป็นที่สนใจแพร่หลายเมื่อบริษัท Sun Microsystems ประกาศให้ JAVA เป็นภาษาสำหรับการสร้างโปรแกรมที่ใช้งานบน Internet โดยเน้นข้อได้เปรียบของภาษา JAVA จุดที่โปรแกรมที่สร้างขึ้นบนคอมพิวเตอร์เครื่องหนึ่งสามารถนำไปทำงานได้บนเครื่องคอมพิวเตอร์เครื่องอื่นที่อาจมีหน่วยประมวลผลต่างชนิดและโดยไม่ต้องคอมไพล์โปรแกรมใหม่

ภาษา JAVA นำไวยากรณ์ภาษาส่วนใหญ่มาจากภาษา C++ เพื่อให้ผู้ที่คุ้นเคยกับภาษา C หรือ C++ ไม่ต้องเสียเวลาศึกษาไวยากรณ์ส่วนใหญ่ของภาษา JAVA แม้ว่าภาษา JAVA จะนำไวยากรณ์มาจากภาษา C++ แต่ก็ตัดกลไกของภาษา C++ ออกไปหลายอย่าง โดยเพิ่มความสามารถให้คอมไพเลอร์จัดการแทน เช่นภาษา JAVA ไม่มี Preprocessor commands, Function prototype และ Header files รวมทั้งกลไกที่จะทำให้ภาษายุ่งยากเกินไปเช่น Default arguments และ Operator overloading ก็ถูกตัดทิ้งไปด้วย และเนื่องจากภาษา JAVA เป็นภาษาเชิงวัตถุ โครงสร้างอย่างเช่น Structures, Unions, Bits fields และ Enumerated types รวมทั้งการทำ Typedef จึงถูกตัดออกไป ภาษา JAVA ถูกออกแบบให้เป็นภาษาเชิงวัตถุดีกว่า ภาษา C++ ดังจะเห็นได้ว่ากลไกที่อาจทำให้เกิดความกำกวมอย่างเช่น Multiple inheritance และกลไกที่อาจทำลายแบบแผนการเขียนโปแกรมเชิงวัตถุที่ดี อย่างเช่น Friend method และ Goto statement ก็ถูกตัดออกไปด้วย

ภาษา JAVA นำความคิดในการสร้างเครื่องจักรสมมุติมาใช้ เพื่อให้ภาษามีคุณสมบัติ Platform independent แต่แยกการแปลภาษาออกไปโดยใช้คอมไพเลอร์ แล้วนำโปแกรมของเครื่องจักรสมมุติที่ได้มาทำงานโดย Interpreter วิธีการนี้ทำให้โปรแกรมทำงานได้เร็วขึ้นกว่าการใช้ Interpreter เพียงอย่างเดียว ในการสร้างโปรแกรมภาษา JAVA ต้องมี Compiler และ Interpreter ของภาษา JAVA ซึ่งปัจจุบันมีหลายบริษัทผลิตชุดโปแกรมสำหรับพัฒนาโปรแกรมภาษา JAVA ออกมา แต่โครงงานนี้เลือกใช้ Java Developer Kit (JDK) ของบริษัท JavaSoft เนื่องจากบริษัท JavaSoft เป็นผู้พัฒนาภาษา JAVA ดังนั้นคอมไพเลอร์ที่มากับ JDK จึงเป็นมาตรฐานของภาษาและมีความทันสมัยมากกว่า

2.3.1 Java Program

Java Program ที่ถูกสร้างโดยภาษา Java มี 3 ชนิดหลักคือ แอปเพล็ต (applet), แอปพลิเคชัน (Application) และ เซิร์ฟเล็ต (servlet)

แอปพลิเคชัน (Application) คือโปรแกรมที่มีลักษณะ standalone ซึ่งสามารถรันโดยใช้ทรัพยากรของคอมพิวเตอร์เครื่องเดียว และจากความจริงที่ว่าโปรแกรมที่เขียนด้วย Java ทั้งหมดจะถูกสร้างจากคลาสต่าง ๆ โดย แอปพลิเคชันจะถูกสร้างจากหลาย ๆ คลาสซึ่งจะต้องมีอยู่ หนึ่งคลาสที่มี main method อยู่ภายในคลาสและชื่อของคลาสนี้จะถูกใช้เป็นชื่อไฟล์ (*.java) โดยโค้ดที่อยู่ใน main method นี้จะถูกจัดการก่อนส่วนอื่นเมื่อโปรแกรมเริ่มทำงาน ซึ่งเป็นจุดเริ่มต้น initial thread ที่จะไปยัง thread อื่น ๆ โดยแอปพลิเคชันจะมีลักษณะดังนี้

```
import java.....
import java.....
public class ชื่อClass1
{
  //statement
}
public static void main(String[] args)
{
  // initial statement
}
```

แอปเพล็ต (Applet) คือโปรแกรมที่ถูกรวมเข้าเป็นส่วนหนึ่งของเว็บเพจโดยใช้แท็ก <APPLET>ใน HTML โดย แอปเพล็ตจะถูกรันด้วยเบราเซอร์ที่มี Java Virtual Machine (JVM) ติดตั้ง เช่น Netcape, Internet Explorer, Hot Java ฯลฯ และเหมือนกับแอปพลิเคชัน คือแอปเพล็ตถูกสร้างจากคลาสต่าง ๆ แต่ไม่ต้องมี main method เป็นจุดเริ่มต้นการทำงานของโปรแกรม แต่จะต้องสร้างซับคลาสจาก Applet class ขึ้นก่อนเซิร์ฟเล็ต (servlet) ใช้สำหรับเพิ่มความสามารถให้กับ server คล้ายกับ Applet ที่ใช้เพิ่มความสามารถให้กับเบราเซอร์ แต่แตกต่างกันตรงที่ servlet ไม่ต้องมี GUI โดยเซิร์ฟเล็ตมีจุดเด่นหลายด้านคือ เร็วกว่า CGI และใช้ API มาตรฐานที่สนับสนุนโดยหลาย ๆ เว็บเซิร์ฟเวอร์

2.3.2 การสร้าง GUI ด้วยสวิง (Swing)

GUI (Graphical User Interface) คือส่วนของโปรแกรมที่ใช้ติดต่อกับผู้ใช้โปรแกรม โดยมีจุดประสงค์ของการออกแบบเพื่อให้ใช้งานง่ายโดยอาศัยกราฟิกเข้ามาเป็นส่วนประกอบ นอกจากนี้ GUI ยังเป็นตัวบอกรูปร่างหน้าตาของโปรแกรมด้วยว่ามีลักษณะอย่างไร ใช้งานง่ายเพียงใด สำหรับใน Java ใช้ Abstract Window Toolkit (AWT) ซึ่งให้คลาสสำหรับสร้าง GUI ให้กับแอปพลิเคชันและแอปเพล็ตมาตั้งแต่ JDK1.0 โดยโปรแกรมที่ใช้ AWT จะมีรูปร่างลักษณะตาม GUI ของระบบปฏิบัติการนั้น ๆ เนื่องจากบางส่วนของ AWT ใช้ เนทีฟโค้ดด้วย เช่น ถ้าใช้ Microsoft Windows ลักษณะของปุ่มกดจะวาดโดยใช้ปุ่มกดจากไลบรารีของ Microsoft Windows เอง โดย Component ของ AWT จะมีลักษณะเป็น Heavyweight Components คือแต่ละ Component จะถูกสนับสนุนโดยตรงจากระบบ Windows

Swing Components คือ ส่วนประกอบหลักที่อยู่ใน JFC ซึ่งใช้สำหรับสร้าง GUI โดยชื่อจะต้องขึ้นต้นด้วยตัว J และอยู่ใน javax.swing package (AWT Components จะอยู่ใน java.awt package) ดังนั้นจะต้องมี import javax.swing.* ใน โปรแกรมด้วย นอกจากนี้อาจต้องมี import java.awt.* กล่าวได้ว่า Swing Components เป็น lightweight เนื่องจากแต่ละ Component เช่น ปุ่มกด จะถูกทำโดย container เช่น เฟรม (Frame), ไดอะล็อกบ็อกซ์ (Dialog box) แทน ซึ่งเป็น heavyweight components โดยปุ่มกดจะถูกวาดโดยใช้คำสั่งพื้นฐานจาก Java เช่น line, rectangle จะไม่ใช้ ไลบรารี่ของระบบปฏิบัติการ ทำให้ Swing Components เหมือนกัน ทุก ๆ แพลตฟอร์ม

ทุก ๆ โปรแกรมที่ใช้ Swing Components จะต้องมีคอนเทนเนอร์ที่เรียกว่า top-level container ได้แก่ JFrame, JApplet และ JDialog อย่างน้อยหนึ่งอย่างในฐานะราก (root) เพื่อจะสนับสนุน Swing Components ในการ painting ซึ่งคือการวาดภาพต่าง ๆ ลงบนจอ และจัดการเหตุการณ์ต่าง ๆ (event handling) โดยแต่ละ top-level container จะต้องมี "content pane" (ซึ่งเรียกใช้โดย getContentPane method) เป็นที่สำหรับวาง visible component ต่าง ๆ และเป็นที่กำหนด Layout manager เราจะไม่ใส่ component ต่าง ๆ โดยตรงลงไปที่ top-level container แต่จะใส่ลงไปใน content pane แทน โดยทั่วไปนิยมใช้ JPanel เป็น Content pane เพราะง่ายและมีลักษณะเป็น opaque โดย default อยู่แล้ว

2.3.3 เทรดในภาษาจาวา

ภาษาจาวาสนับสนุนการโปรแกรมมัลติเทรดดิ้ง ด้วยกลไกที่กำหนดขึ้นในภาษา ต่างกับภาษา C และ C++ ที่สนับสนุนเทรดด้วยไลบรารี (Library) ที่เขียนขึ้นต่างหากสำหรับแต่ละคอมไพเลอร์และระบบปฏิบัติการ โปรแกรมภาษาจาวาทุกๆ โปรแกรมจะต้องมีเทรดอย่างน้อยหนึ่งเทรดเสมอ

คอมไพเลอร์จะเป็นผู้สร้างเทรดให้แก่ main() ของทุก ๆ โปรแกรมเรียกว่าเทรดหลัก (main tread) และจาวา อินเตอร์พรีเตอร์ จะเป็นผู้ควบคุมเทรดหลักที่หลังจากทำงานแล้วอาจจะมีการสร้างเทรดอื่นๆได้อีก

รายละเอียดการและการทำงานของคลาสเทรด

คลาสเทรดถูกกำหนดใน package java.lang ใช้สำหรับสร้างอินสแตนซึ่งเรียกสั้นๆว่า เทรด คลาสเทรดมีโปรแกรมสำหรับการทำงานของเทรดตัวหนึ่ง ที่มีการดำเนินของโปรแกรมแยกออกจากโปรแกรมที่สร้างมัน เทรดที่สร้างขึ้นจะยังไม่เริ่มต้นทำงานจนกว่า start() ของมันจะถูกเรียกเมื่อ start() ถูกเรียกแล้วมันจะไปเข้าคิว จนมันถูกเลือกไปทำงาน โปแกรมใน run ก็จะถูกทำงาน

คลาสเทรดมีเมธตอดสำหรับควบคุมให้เทรดเข้าสู่สถานะต่างๆ เช่น start(), yield(), sleep(), suspend(), resume(), และ stop() โดยปกติเราจะไม่โอเวอร์ไรด์ (override) เมทธอดเหล่านี้นอกจากนี้ยังมี

```
public void run()
```

ซึ่งเป็นเมทธอดที่ว่างเปล่า ที่โดยปกติเราจะต้องโอเวอร์ไรด์เพื่อระบุว่าเมื่อเทรดนั้นอยู่ในสถาน running จะทำอะไรบ้าง หากไม่ทำการโอเวอร์ไรด์ run() ก็จะได้เทรดที่ไม่ทำอะไรเลย

รายละเอียดการทำงานของอินเตอร์เฟสรันเอเบิล (Interface runable)

วิธีที่สองในการสร้างเทรดคือสร้างคลาสที่อิมพลีเมนรันเอเบิลอินเตอร์เฟส (implement runable interface) ในอินเตอร์ฟสนี้มีเพียง run() เป็น abstract method คลาสที่อิมพลีเมนรันเอเบิลอินเตอร์เฟสจะต้องมี run เพื่อระบุการทำงานของเทรดในสถานะ running

อินเตอร์เฟสรันเอเบิลถูกกำหนดไว้ใน java.lang ดังนี้

```
Public abstract interface Runable
{       Public abstract void run();
}
```

การใช้รันเอเบิลสร้างเทรดอาจทำได้สองวิธี

1) สร้างคลาสที่อิมพลีเมนรันเอเบิลโดยมีเทรดตัวหนึ่งเป็นสมาชิก
2) สร้างอินสแตนของคลาสที่มีอิมพลีเมนรันเอเบิลแล้วส่งให้เป็นพารามิเตอร์ของคอนสตรักเตอร์ของคลาสเทรด

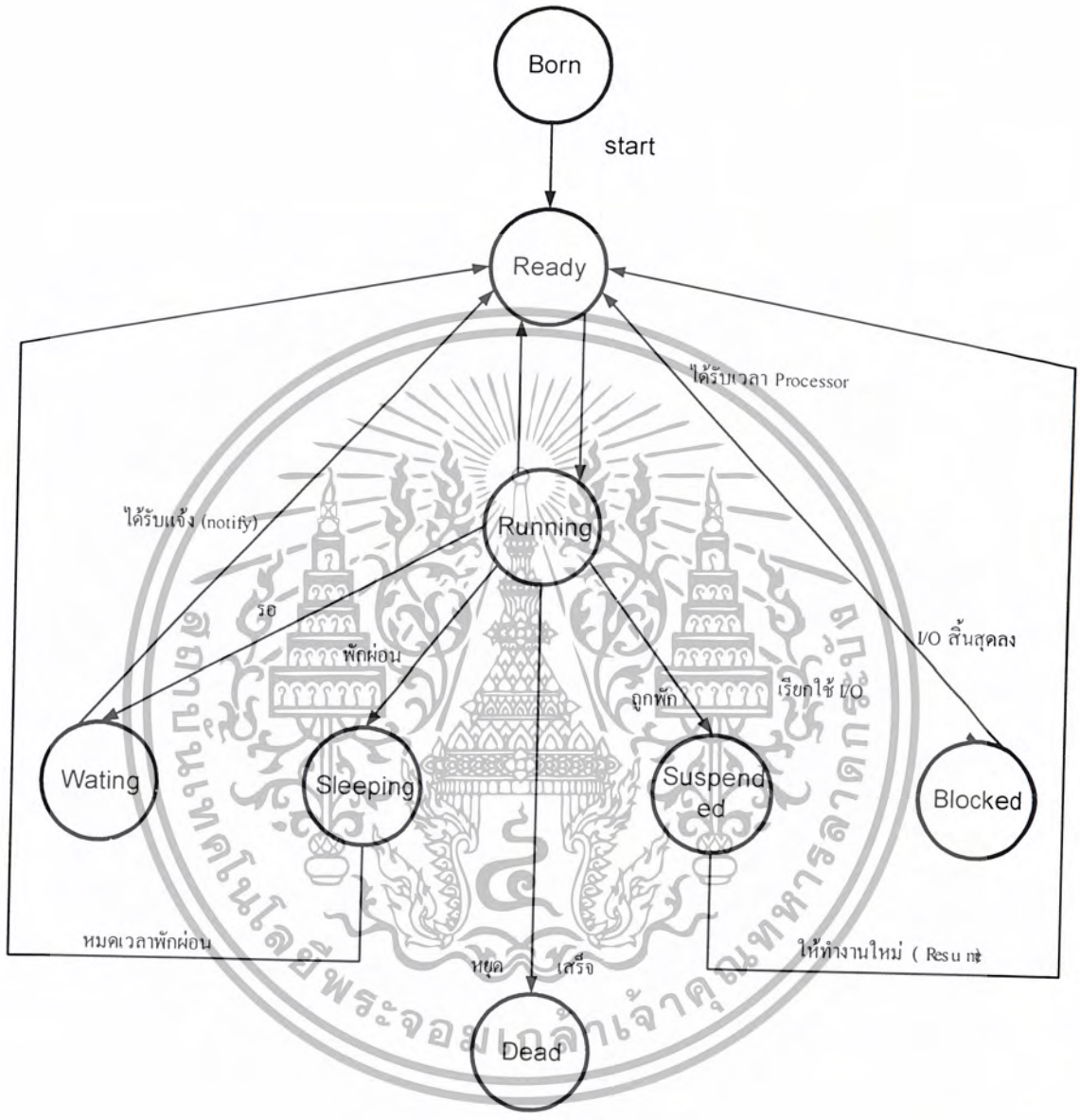
เมธตอดของเทรดที่สำคัญในภาษาจาวา

เมธตอดของ java.lang.tread	
currentThread()	คืนค่ารีเฟอร์เรนซ์ของเทรดที่กำลัง execute
destroy()	ทำลายเทรดโดยไม่ clean up
getName()	คืนชื่อของเทรด
getPiority()	คืนค่าความสำคัญของเทรด
getThreadGroup()	คืนค่า thread group
isAlive()	เทรดอยู่หรือไม่
isdaemon()	เทรดเป็นเดมอนเทรดหรือไม่
join(long mills)	คอยให้เทรดตาย
resume()	เรียกเทรดที่ถูก suspend() ให้กลับมาทำงาน
run()	จะถูกเรียกทำงานเมื่อเทรดเข้าสู่สถานะ ready
setdaemon()	กำหนดให้เทรดเป็นเดมอนเทรด
setName(String name)	กำหนดชื่อเทรด
setPiority()	กำหนดความสำคัญเทรด
sleep(long mills)	สั่งให้เทรดหลับ
start()	สั่งให้เทรดเริ่มต้นทำงาน JVM จะเรียก run() มาทำงานให้เอง
stop()	สั่งให้เทรดหยุดทำงาน อาจก่อให้เกิดเอกเซฟชัน
suspend()	สั่งให้เทรดหยุดการทำงานชั่วคราว
yield()	สั่งให้เทรดเปลี่ยนสถานะจาก running เป็น ready

Methods inherited from class java.lang.Object	
Wait()	สั่งให้เทรดเข้าสู่สถานะบล็อก (blocled state)
Notify()	ใช้เรียกเทรดอื่นที่ถูก wait() กลับสู่สถานะ ready
notifyAll()	สั่งให้ทุกเทรดที่ถูก wait มีโอกาสถูกเลือกเข้าทำงานเท่าๆกัน

วัฏจักรของเทรด (Thread Lifetime)

ในเวลาต่างๆเทรดจะอยู่ในสภาวะต่างๆ (thread states) เริ่มจากสภาวะเริ่มต้น (Born state) เทรดจะอยู่ในสภาวะดังกล่าวจนกว่าเมธตอด start ของเทรดดังกล่าวจะถูกเรียก เมื่อเรียกเมทธอด start แล้วเทรดจะอยู่ในสภาวะพร้อม (Ready State) เทรดซึ่งมีความสำคัญสูงสุดจะเข้าไปอยู่ในสภาวะทำงาน (Running State) เมื่อได้รับเวลาจากหน่วยประมวลผล (Processor) เทรดจะหมดสภาพหรือตาย (Dead State) เมื่อมันทำงานเสร็จสมบูรณ์ หรือถูกเรียกด้วยเมธตอด stop เทรดที่ตายแล้วจะถูกกำจัดโดยระบบ บางครั้งเทรดอาจจะอยู่ในสภาวะบล็อกเมื่อเทรดมีการเรียกใช้ Input/Output Block เทรดจะกลายเป็นสภาวะพร้อมเมื่อการทำงานของ I/O ที่รออยู่สิ้นสุดลงเทรดเทรดที่อยู่ในสภาวะบล็อกจะไม่สามรถใช้หน่วยประมวลผล ถึงแม้ว่าหน่วยประมวลผลจะว่างอยู่ก็ตาม เทรดจะอยู่ในสภาวะพักผ่อนเมื่อเทรดถูกเรียกด้วยเมธตอด sleep เทรดที่พักผ่อนจะกลับมาอยู่ในสภาวะพร้อมอีกครั้งเมื่อหมดเวลาพักผ่อน เทรดที่อยู่ในสภาวะพักผ่อนจะไม่สามารถใช้เวลาของหน่วยประมวลผล ถึงแม้ว่าหน่วยประมวลผลจะว่างอยู่ก็ตาม เมื่อเทรดที่ทำงานอยู่ถูกเรียกด้วยเมทธอด suspend จะกลายสภาพเป็นถูกพัก (Suspend State) เทรดที่ถูกพักจะกลายสภาพมาเป็นสภาวะพร้อมอีกครั้ง เมื่อถูกเรียกด้วยเมธตอด notify จากเทรดอื่นที่ทำงานกับวัตถุนั้น เทรดทุกอันจะเปลี่ยนสภาพเป็นพร้อมเมื่อถูกดรียกด้วยเมทธอด notifyAll ดังรูปที่ 2.1แสดงวัฏจักรของเทรด



รูปที่2.2 แสดงวัฏจักรของเทรด

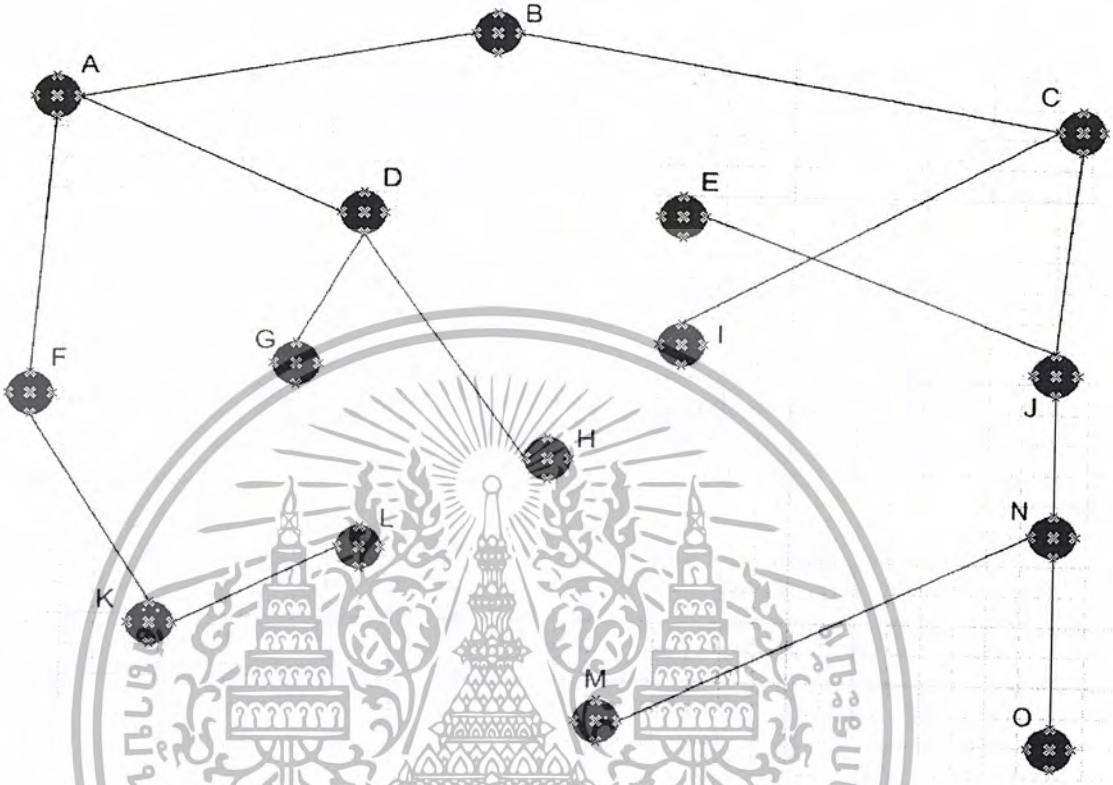
2.4 หลักการหาเส้นทางที่เหมาะสมที่สุด

หลักการพื้นฐานของการหาเส้นทางที่เหมาะสมที่สุด (optimality principle) ที่ไม่ขึ้นอยู่กับโครงสร้างเส้นทางการจราจรแบบใดกล่าวว่าถ้าทางแยกจุดหนึ่ง (จุด B) อยู่บนเส้นทางที่เหมาะสมที่สุดระหว่างต้นทาง (จุด A) และปลายทาง (จุด C) แล้ว เส้นทางนั้นจะเป็นเส้นทางที่เหมาะสมที่สุด

ระหว่างทางแยกนั้น (จุด B) กับปลายทาง (จุด C) ด้วยข้อความนี้สามารถพิสูจน์ได้โดยใช้หลักการตรรกศาสตร์ธรรมดาคือ สมมุติให้ r1 r2 คือเส้นทางที่ดีที่สุดระหว่างจุด A กับจุด C ถ้า r1 คือเส้นทางที่เหมาะสมที่สุดระหว่างจุด A กับจุด B แล้วให้ r2 เป็นเส้นทางระหว่างจุด B กับ C ถ้ามีเส้นทางอื่นที่เหมาะสมกว่าเส้นทาง r2 ก็ควรนำมาใช้เป็นเส้นทางที่ดีที่สุดระหว่างจุด A กับจุด C ซึ่งขัดกับข้อสมมุติฐานที่ว่า r1 r2 คือเส้นทางที่ดีที่สุด ดังนั้นเส้นทาง r2 จึงต้องเป็นเส้นทางที่ดีที่สุดระหว่างจุด B และจุด C ด้วย

เมื่อนำหลักการพื้นฐานของการหาเส้นทางที่เหมาะสมที่สุดมาใช้พิจารณาหาเส้นทางที่ดีที่สุดจากทางแยกกลุ่มหนึ่งกับทางแยกสมมุติจุดหนึ่งมาเขียนเป็นรูปต้นไม้ (tree) ที่มีทางแยกสมมุติจุดนั้นเป็นโหนดราก จะเรียกต้นไม้นี้ว่า ซิงค์ทรี (sink tree) ดังรูปที่ 2.3 และ 2.4 ซึ่งสมมุติให้ใช้จำนวนครั้งในการเดินทางผ่านแยกเป็นมาตรฐานที่ใช้ในการเลือกเส้นทาง การสร้างซิงค์ทรีของทางแยกจุดหนึ่งไม่จำเป็นว่าจะมีได้เพียงรูปแบบเดียว วัตถุประสงค์ของอัลกอริทึมสำหรับการเลือกเส้นทางคือการหาซิงค์ทรีให้พบและใช้เป็นเส้นทางในการเดินทางสำหรับทุกทางแยกที่อยู่ในเส้นทางการจราจรหนึ่งๆ

รูปที่ 2.3 เส้นทางการจราจร



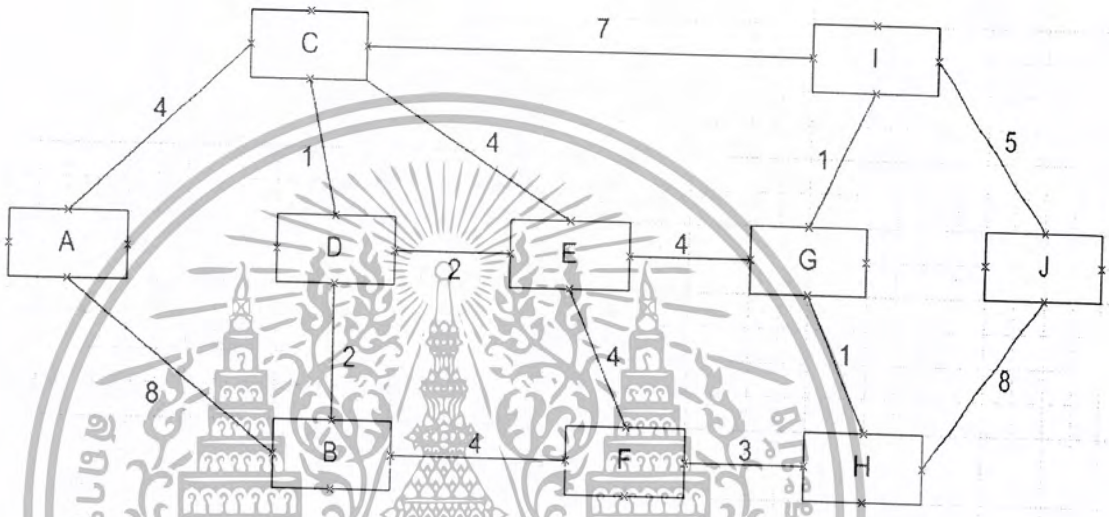
รูปที่ 2.4 ซิงค์ทรีสำหรับทางแยก B

เนื่องจากซิงค์ทรีเป็นรูปต้นไม้ ชนิดหนึ่งจึงไม่มีลักษณะของวังวน (loop) อยู่ภายใน การเดินทางจากทางแยกใดๆ ไปยังทางแยกอื่นจะสามารถทราบจำนวนครั้งได้แน่นอน แต่ในความเป็นจริงนั้นอาจเป็นไปได้ค่อนข้างยากเพราะเส้นทางและทางแยกอาจจะเสียหายหรือถูกยกเลิกการใช้งานชั่วคราวและอาจจะกลับมาใช้งานได้อีกเมื่อไหร่ก็ได้ ดังนั้นทางแยกจะมองเห็นโครงสร้างของเส้นทางการจราจรมีการเปลี่ยนแปลงอยู่ตลอดเวลา การนับจำนวนครั้งที่กล่าวมาจึงไม่สามารถนำไปใช้งานได้โดยตรงอย่างไรก็ตามซิงค์ทรีได้รับความนิยมในการนำมาใช้เป็นมาตรฐานในการเปรียบเทียบประสิทธิภาพของอัลกอริทึมแบบต่างๆ

2.5 หลักการเลือกเส้นทางเดินที่สั้นที่สุด

วิธีการหาเส้นทางเดินที่สั้นที่สุด (Shortest Path First Protocol (SPF)) จะใช้หลักการของทฤษฎีกราฟของเส้นทางการจราจรที่มีทางแยกเป็นโหนดของกราฟและเส้นทางที่เชื่อมระหว่างโหนดเป็น

ด้าน (arc) ด้านจะมีการกำหนดเลเบลด้วยค่าน้ำหนักซึ่งคำนวณจากจากระยะทาง แบนด์วิดธ์ และปริมาณการจราจรเป็นต้น การหาเส้นทางที่สั้นที่สุดก็คือการหาเส้นทางระหว่างโหนดที่มีผลรวมของค่าน้ำหนักน้อยที่สุดเมื่อคำนวณค่าน้ำหนักของแต่ละด้านแล้วจะแสดงในรูปแบบของ link state database ได้ดังนี้

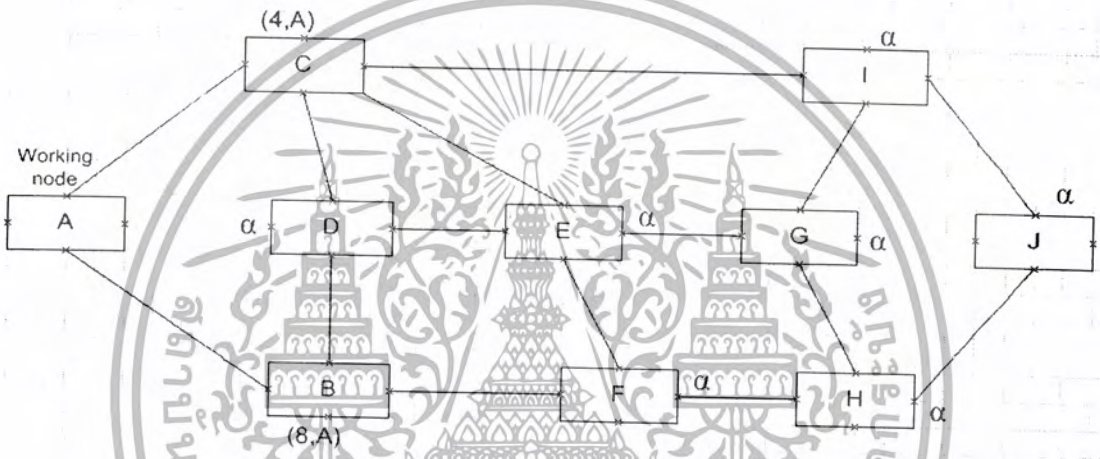


รูปที่ 2.5 แสดงกราฟเส้นทางการจราจร

A	C-4	B-8		
B	A-8	D-2	F-4	
C	A-4	D-1	I-7	E-4
D	B-2	C-1	E-2	
E	C-4	D-2	F-4	G-2
F	B-4	E-4	H-3	
G	E-2	I-1	H-1	
H	F-3	G-1	J-8	
I	C-7	G-1	J-5	
J	H-8	I-5		

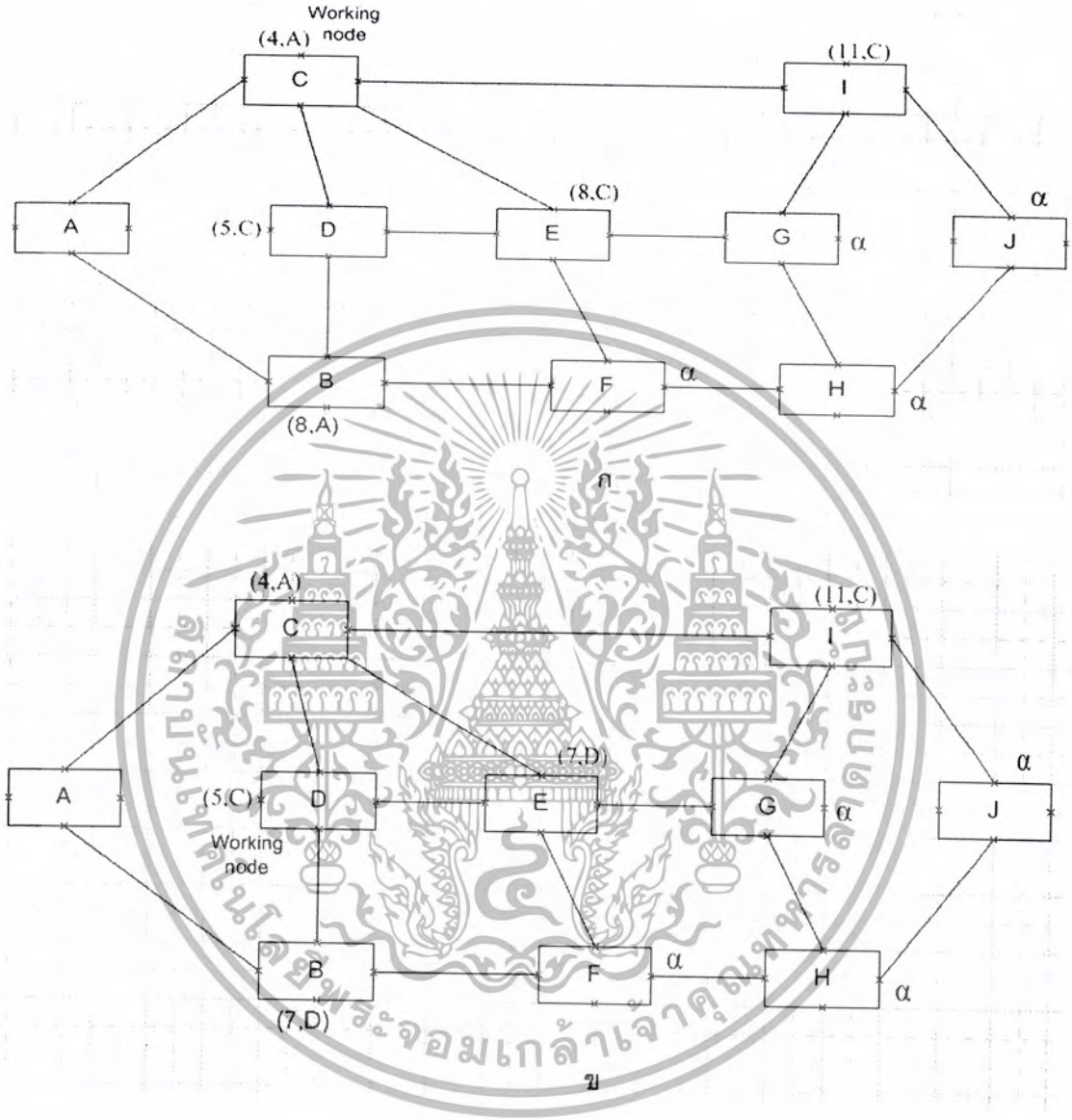
ตารางที่ 2.1 ตาราง link state database

เมื่อต้องการหาเส้นทางที่สั้นที่สุดโดยกำหนดจุดเริ่มต้นและจุดปลายทางโดยในตัวอย่างนี้จะใช้โหนด A เป็นจุดเริ่มต้นขั้นตอนการทำงานแรกจะกำหนดให้ทุกโหนดนอกจากจุดเริ่มต้นมีค่าเป็นอนันต์ จากนั้นจะกำหนดให้ A เป็นโหนดที่ทำงาน (working node) และทำการหาเส้นทางไปยังโหนดที่อยู่ติดกันและกำหนดเลเบลด้วยระยะทางจากโหนดที่ทำงานในรูปแบบของ (น้ำหนัก, โหนด) ดังรูปที่ 2.6



รูปที่ 2.6 แสดงการเริ่มต้นการหาเส้นทาง

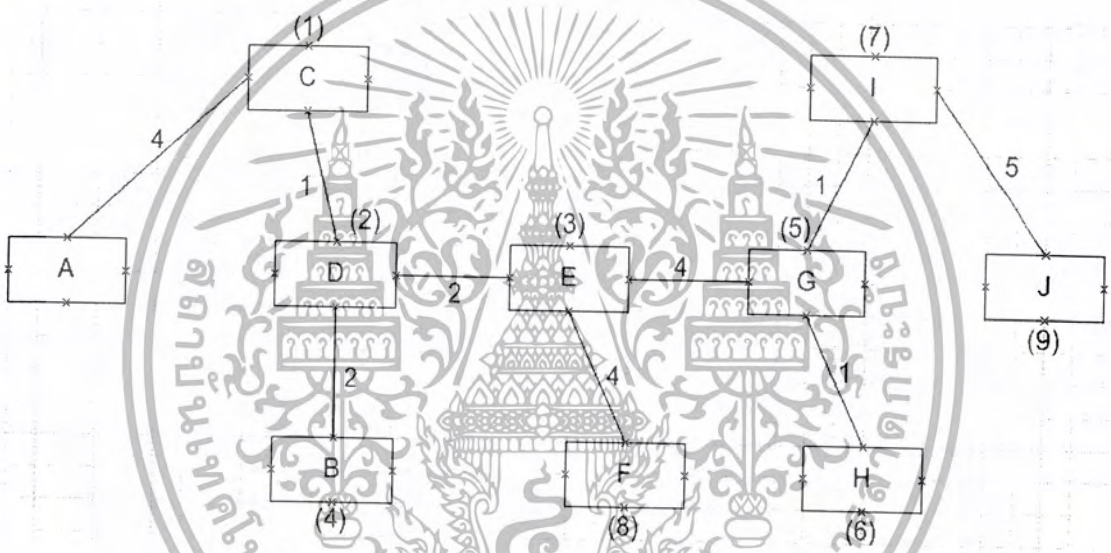
โหนดที่มีค่าน้ำหนักน้อยที่สุดจะกำหนดให้เป็น working node ในรอบการทำงานถัดไปซึ่งจะพบว่าในการหาเส้นทาง working node จะเป็นโหนดที่มีเส้นทางไปยังจุดเริ่มต้นที่สั้นที่สุดเสมอ ดังรูปที่ 2.7



รูปที่ 2.7 (ก) และ (ข) แสดงการกำหนด working node

ในกรณีที่ค่ารวมของเลเบลน้อยกว่าเลเบลของโหนดที่อยู่ติดกันค่าของโหนดเลเบลจะเปลี่ยนไป เนื่องจากพบเส้นทางที่ดีกว่า ดังรูปที่ 2.6 เมื่อโหนด D เป็น working node จะพบว่าเส้นทางจากโหนด A ผ่าน D ไปยังโหนด B เป็นเส้นทางที่มีค่าเลเบลสั้นกว่าหรือเส้นทางจากโหนด A ไปโหนด E ผ่านโหนด D เป็นเส้นทางที่สั้นกว่าเป็นต้น

การทำงานในการหาเส้นทางที่สั้นที่สุดจะทำดังข้างต้นซ้ำไปเรื่อยๆ โดยเปลี่ยน working node จนกว่าเส้นทางที่เป็นไปได้ทั้งหมดจะได้รับการตรวจสอบ ซึ่งได้เป็นการกำหนดเลเบลที่สั้นที่สุดจากโหนดต้นทางไปยังโหนดต่างๆ ซึ่งจากขั้นตอนดังกล่าวจะได้โทโปโลยีเส้นทาง (routing topology)สำหรับโหนด A ในรูปแบบของ spanning tree ดังรูปที่ 2.7 โดยหมายเลขในวงเล็บจะแสดงลำดับของการเลือก เส้นทางที่ไม่ได้อยู่ในทรีจะถูกบล็อกและสามารถนำมาใช้งานได้ในกรณีที่เส้นทางที่ใช้งานอยู่ไม่สามารถใช้งานได้



รูปที่ 2.8 แสดงโทโปโลยีเส้นทางสำหรับโหนด A ในรูปแบบของ spanning tree

บทที่ 3
การออกแบบ

3.1 เอาท์พุทของระบบ

กำหนดเอาท์พุตที่จะศึกษาดังนี้

- ความเร็วเฉลี่ยของรถ ณ เวลาที่สนใจ
- จำนวนรถที่มีในท้องถนน ณ เวลาที่สนใจ

3.2 อินพุทของระบบ

ซึ่งค่าพารามิเตอร์ต่าง ๆ เหล่านี้ เป็นเงื่อนไขเริ่มต้นของการจำลอง

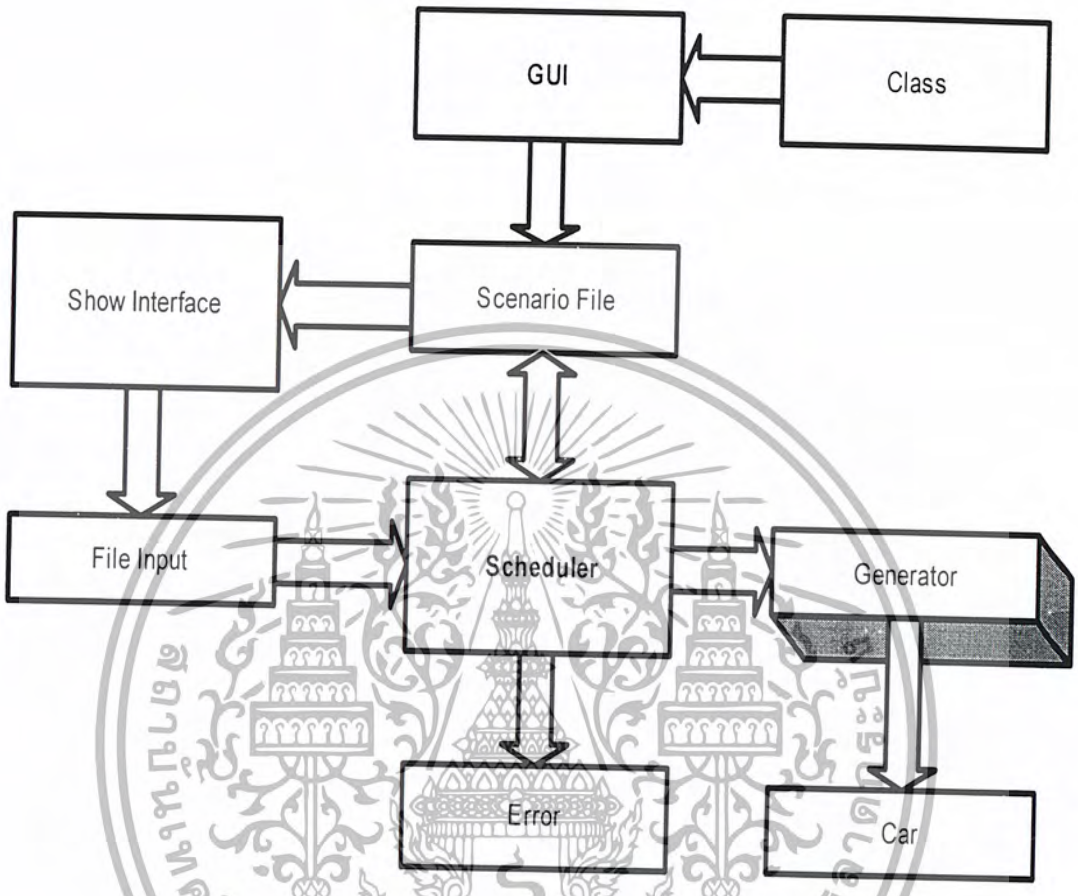
- ช่วงเวลาในการจำลองซึ่งจะอ้างอิงกับเวลาจริงเช่น ต้องการจำลองตั้งแต่ช่วงเลา 8.00 น. ถึง 10.00 น. ก็สามารถใส่ค่าพารามิเตอร์ที่เป็นค่าเวลาที่ใช้ในการจำลองทั้งหมดลงไป
- ระยะเวลาความถี่ในการปล่อยรถจากจุดกำเนิดแต่ละจุด เช่น จุดที่ 1 ต้องการปล่อย 1 คันทุก ๆ 5 นาที ก็จะใส่ค่าเวลา เป็นนาทีในส่วนนี้
- ระยะเวลาในการปล่อยรถของแต่จะจุดกำเนิดรถ ซึ่งจะเป็นค่าเวลาอ้างอิงเทียบกับเวลาจริงเช่น ให้ จุดกำเนิดรถที่เราสนใจ ปล่อยรถ ตั้งแต่ เวลา 9.00 น. ถึง 9.30 น. ก็สามารถใส่ค่าเวลาเริ่มต้นและสิ้นสุดลงไปได้
- ความเร็วเริ่มต้นที่กำหนดให้รถ ในแต่ละ จุดกำเนิด ให้ความเร็วมีหน่วยเป็นกิโลเมตรต่อชั่วโมง(km/hr)
- มีการพิจารณาสัญญาณไฟจราจร ประกอบด้วยหรือไม่ ถ้ามีใส่ค่าเวลาของการเปลี่ยนสัญญาณเป็นวินาที

3.3 สถาปัตยกรรมของระบบ (Architecture)

จากรูปที่ 3.1 สามารถอธิบายการทำงานในส่วนต่าง ๆ ได้ดังนี้

- **GUI** เป็นส่วนที่ใช้ในการสร้างกราฟิกที่ใช้แสดงต่าง ๆ ภายในระบบ
- **Class** เป็นส่วนที่ใช้เป็นเครื่องมือในการสร้างกราฟิกซึ่งมีทั้งคลาสต้นแบบและคลาสที่สร้างขึ้นสำหรับ Object ที่เราใช้ในระบบ

- **Scenario File** ใช้เก็บค่าของระบบทั้งในส่วนที่เป็นค่าที่ต้องนำไปใช้แสดงและค่าที่ต้องนำไปประกอบการทำการจำลองในการจำลองหนึ่งครั้งจะเท่ากับหนึ่งบท (Scenario)
- **Show Interface** เป็นส่วนที่ใช้แสดงค่า GUI ที่สร้างไว้ในรูปไฟล์ต่าง ๆ รวมทั้งแสดงผลจากการ Simulate ด้วย
- **Input file** จะเป็นส่วนที่รับค่าพารามิเตอร์ต่าง ๆ มากจากตัวแสดงอินเตอร์เฟส แล้วนำค่าเหล่านั้นมาประกอบการทำการจำลอง
- **Scheduler** (ตัวจัดลำดับงาน) มีหน้าที่จัดการตารางการทำงานในทั้งหมดในการทำการจำลองแต่ละครั้ง
- **Generator** (ตัวกำเนิดรถ) จะทำหน้าที่กำเนิดรถออกมาโดยจะต้องได้รับการกระตุ้นจากตัวจัดลำดับงานจึงจะ ทำงาน
- **Car** (รถ) จะเกิดขึ้นเมื่อมีการกำเนิดรถจากตัวกำเนิดรถหลังจากนั้นจึงจะทำงานตามหน้าที่
- **Error** เป็นส่วนของการแจ้งผลการผิดพลาดที่เกิดขึ้นจากการทำการจำลอง



รูปที่ 3.1 แสดงสถาปัตยกรรมของระบบ

3.4 ตัวจัดลำดับงาน (Scheduler)

ตัวจัดลำดับงานถือเป็นแกนหลักของระบบที่ใช้ควบคุมการทำงานทั้งหมดในการทำการจำลองก็ว่าได้ เพราะเมื่อเริ่มการทำการจำลองตัวจัดลำดับงานก็จะเริ่มทำงานก่อน หน้าที่หลังจากนั้นจะแบ่งเป็นข้อ ๆ ดังต่อไปนี้

- กระตุ้นให้ องค์ประกอบอื่น ๆ ที่ใช้ในการจำลองเริ่มทำงานนั่นเองซึ่งเป็นการโยนหน้าที่หรือโยนงานในเวลานี้ให้กับองค์ประกอบอื่น ๆ เช่นตัวกำเนิดรถ หลังจากนั้นตัวจัดลำดับงานก็จะว่างงานและรอคอยการกระตุ้นให้กลับมาทำงานอีกครั้งจากองค์ประกอบอื่น ๆ เช่นกันสาเหตุที่ต้องสลับการทำงานก็เนื่องมาจาก ณ เวลาหนึ่งจะมีกระบวนการที่ทำงานได้เพียงกระบวนการเดียว

- จัดลำดับงาน ซึ่งก็มีความหมายตรงกับชื่อของมันก็คือเมื่อมีการโยนงาน หรือการกระตุ้นจากองค์ประกอบอื่นๆ มันจะทำการอ่านค่าแล้วจัดลำดับคิวของการทำงานครั้งต่อไปให้กับองค์ประกอบนั้น ๆ โดยจะเรียงลำดับจากเวลาน้อยสุดไปยังเวลาที่มากสุดแล้วจัดงานที่อยู่ ณ เวลาน้อยสุดไปทำงานก่อน โดยจะโยนงานให้กับองค์ประกอบอื่นเช่นเดิม
- เช็คค่าการเก็บข้อมูล ในส่วนนี้มันจะเช็คพารามิเตอร์จากที่ผู้ใช้ป้อน ซึ่งจะเป็นค่าของระยะเวลาที่ใช้ในการเก็บข้อมูลในแต่ละครั้งของการทำจำลองทั้งหมด
- เช็คเวลาสิ้นสุดของการทำงาน ตรงส่วนนี้ตัวจัดลำดับงานจะทำการเช็คค่าเวลาที่ได้จากเวลาของงานแต่ละงานที่เข้ามาซึ่งเวลาจะเพิ่มขึ้นเรื่อย ๆ จะหยุดเมื่อเวลาถึงจุดที่เท่ากับการกำหนดเวลาทำงานไว้ในตอนต้น จึงหยุดการทำงานและอาจหยุดการทำงานอีกกรณีหนึ่งเนื่องจากหมดงาน ในตารางงานของมัน

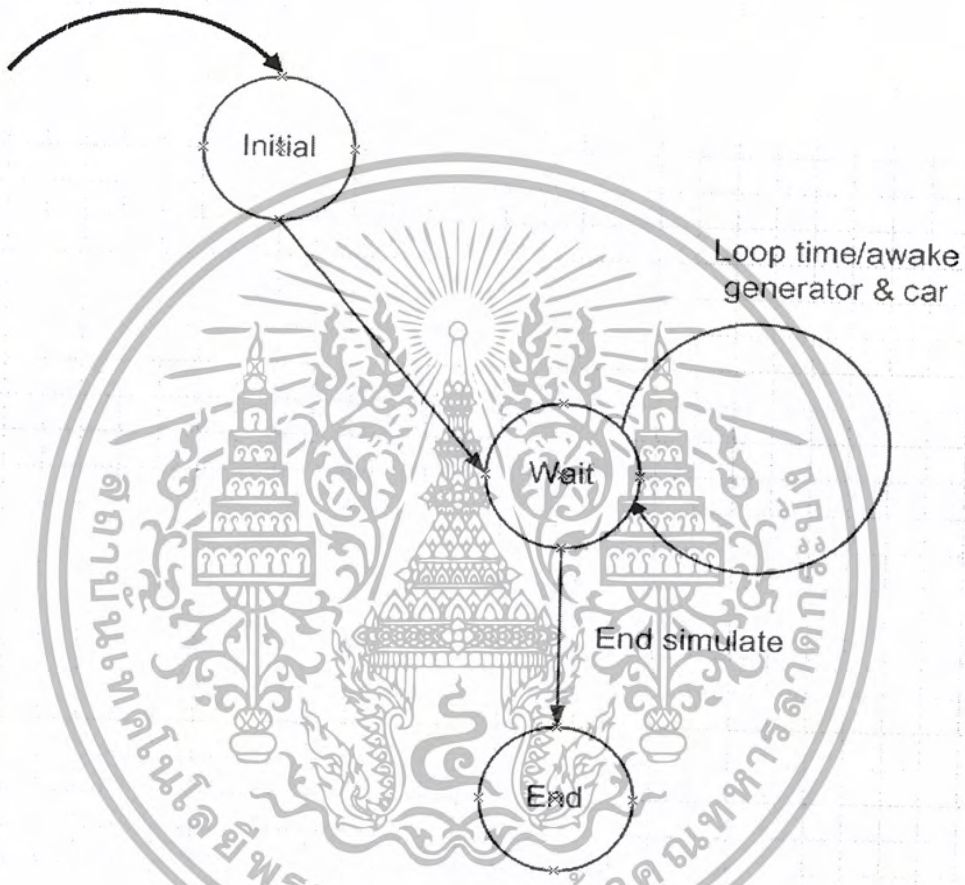
3.4.1 ขั้นตอนการสร้างตัวจัดลำดับงาน

1. วิเคราะห์หน้าที่การทำงานต่าง ๆ ของตัวจัดลำดับงานจากรายละเอียดต่าง ๆ ที่ได้แจกแจงไปแล้วเพื่อนำไปเป็นข้อมูลในการสร้างคลาสของตัวจัดลำดับงาน
2. นำข้อมูลที่ได้ในขั้นตอนแรก นำมาใช้สร้างตัวแปร (Attribute) ต่าง ๆ ที่ใช้เก็บค่าสถานะของการทำงาน ณ เวลาต่าง ๆ ของตัวจัดลำดับงาน
3. สร้างส่วนที่ใช้ในการกระทำ (Method) ต่าง ๆ ตามหน้าที่ของตัวจัดลำดับงาน
4. สร้างตัวอย่าง (Instance) จากคลาสของตัวจัดลำดับงานที่ได้สร้างไปแล้วเพื่อนำมาทดสอบการทำงาน
5. แก้ไขปรับปรุงการทำงานในส่วนที่ยังไม่สมบูรณ์

3.4.2 ขั้นตอนการทำงานของตัวจัดลำดับงาน

1. เมื่อตัวจัดลำดับงานถูกสร้างขึ้นจะทำการอ่านค่าเริ่มต้นต่าง ๆ ที่ใช้ในการทำงาน เช่น ค่าเวลาเริ่มต้นและสิ้นสุดการทำงาน จำนวนตัวกำเนิดรถทั้งหมดในระบบ
2. เมื่อถูกสั่งให้ทำงานก็จะมีวงรอบของการทำงานโดยจะกำหนดเวลาเริ่มต้นให้ตรงกับค่าเวลาเริ่มต้นที่อ่านได้ในตอนแรกและไปกระตุ้นให้วัตถุต่าง ๆ ทำงานเมื่อถึงวงรอบของเวลาทำงานพร้อมทั้งเพิ่มเวลาขึ้นไปทุกครั้งที่มีการทำงาน

3. เมื่อเวลาในระบบเพิ่มขึ้นจนเท่ากับเวลาสิ้นสุดการทำงาน ตัวจัดลำดับงานก็จะออกจากวงรอบการทำงานและหยุดการทำงาน



รูป 3.2 แสดง state diagram ของ Scheduler

3.5 ตัวกำเนิดรถ (Generator)

หน้าที่การทำงานหลักของส่วนนี้คือแจ้งงานให้กับตัวจัดลำดับงานทราบ แล้วจึงรอคอยการแจ้งการทำงานจากตัวจัดลำดับงานอีกที ซึ่งอธิบายแต่ละหน้าที่ดังนี้

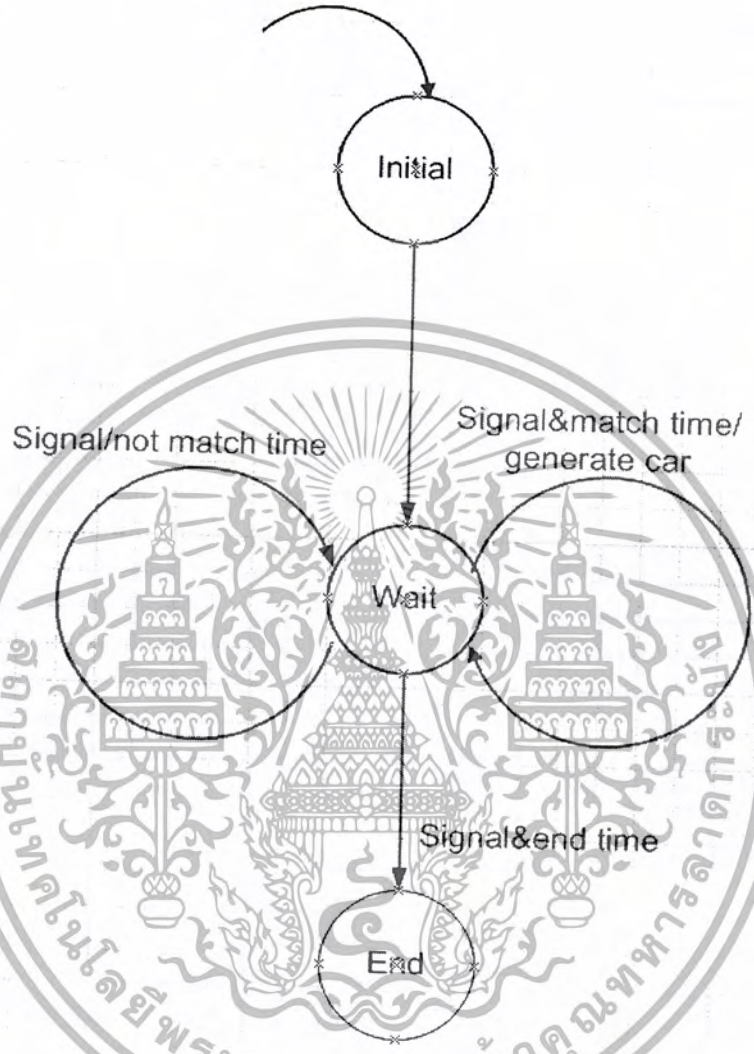
- แจ้งงานให้ตัวจัดลำดับงานทราบเป็นการทำงานขณะที่ตัวจัดลำดับงานโยนงานมาให้ หลังจากนั้น จึงทำในหน้าที่ของตนแล้วแจ้งสถานะกลับเช่น มีงานต่อไปเวลาใดหรือหมดงานที่จะทำแล้ว เพื่อให้งานไปอยู่ที่ตัวจัดลำดับงานอีกครั้ง
- กำเนิดรถหลังจากที่ได้รับการกระตุ้นจากตัวจัดลำดับงานมันจะทำการกำเนิดรถออกมาแล้วก็โยนงานกลับไปให้ตัวจัดลำดับงานอีกครั้ง

3.5.1 ขั้นตอนการสร้างตัวกำเนิดรถ

1. วิเคราะห์หน้าที่การทำงานของตัวกำเนิดรถเพื่อนำมาเป็นข้อมูลที่ใช้ในการสร้างคลาสของตัวกำเนิดรถ
2. นำข้อมูลจากขั้นตอนแรกมาใช้ในการสร้างตัวแปร (Attribute) ต่าง ๆ ที่ใช้ในการเก็บค่าสถานะ การทำงานของตัวกำเนิดรถ
3. สร้างส่วนที่ใช้ทำงาน (Method) ตามหน้าที่ของตัวกำเนิดรถ
4. สร้างตัวอย่างของตัวกำเนิดรถขึ้นมาเพื่อใช้ทดสอบการทำงานก่อนการใช้งานจริง
5. แก้ไขข้อบกพร่อง (ถ้ายังไม่สามารถทำงานได้ตามวัตถุประสงค์)

3.5.2 ขั้นตอนการทำงานของตัวกำเนิดรถ

1. เมื่อตัวกำเนิดรถถูกสร้างขึ้น ก็จะอ่านค่าเริ่มต้นต่าง ๆ ที่ใช้ในการทำงานแล้วรอให้ตัวจัดลำดับงานเรียกให้ทำงาน
2. เมื่อมีการเรียกให้ทำงาน ตัวจัดลำดับงานจะตรวจสอบเวลางานของตัวเองโดยจะเทียบเวลางานถัดไปกับเวลาปัจจุบันที่ตัวจัดลำดับงานส่งมาให้ ถ้าเวลาตรงกับเวลางานถัดไปที่มี ตัวกำเนิดรถก็จะทำการสร้างรถขึ้นมา 1 คัน แล้วเพิ่มเวลางานถัดไปของตัวมันเองเพื่อใช้ตรวจสอบเมื่อถูกเรียกให้ทำงานครั้งต่อไป หลังจากนั้นก็จะมีการตรวจสอบเวลาสิ้นสุดงานด้วย ถ้าถึงเวลาสิ้นสุดงานตัวแปรทำหน้าที่แสดงสถานะสิ้นสุดงานจะถูกกำหนดค่าต่างไปจากเดิมเพื่อให้ตัวจัดลำดับงานรู้ว่าตัวกำเนิดรถนี้ สิ้นสุดงานแล้ว ครั้งต่อไปไม่ต้องเรียกทำงานแล้ว



รูป 3.3 แสดง state diagram ของ Generator

3.6 รถ (Car)

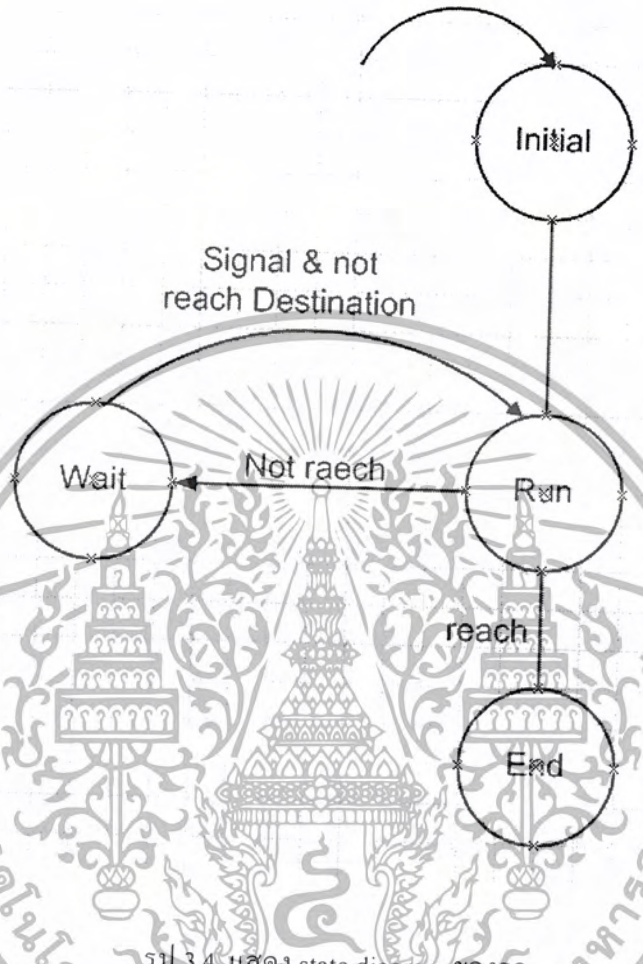
หน้าที่การทำงานของวัตถุประเภทนี้หรือกระบวนการของวัตถุนี้หลังจากที่ได้รับการกระตุ้นต่อจากตัวกำเนิดรถแล้วมันก็จะมีการเช็คสถานะว่าถึงปลายทางแล้วหรือไม่ ถ้ายังก็เคลื่อนที่ต่อไป พร้อมทั้งเช็ค ค่าสถานะว่ามีรถข้างหน้าหรือไม่ ถ้ามันไม่เจอก็ไม่ต้องเบรก แต่ถ้ามีรถมีรถอยู่ข้างหน้าก็จะต้องลดความเร็วลงหลังจากที่มันทำงานในส่วนของตัวเองเสร็จก็จะแจ้งเวลาต่อไปที่จะทำงานให้กับตัวจัดลำดับงานทราบแล้ว รอการเรียกทำงานอีกครั้ง

3.6.1 **ขั้นตอนตอนการสร้างรถ**

1. วิเคราะห์หน้าที่การทำงานของรถ เพื่อนำมาเป็นข้อมูลในการออกแบบส่วนการทำงาน และคุณสมบัติต่าง ๆ ที่ต้องมีในคลาสของรถนี้
2. จากข้อมูลในขั้นตอนแรกนำมาใช้ในการสร้างตัวแปรที่ใช้เก็บค่าสถานะต่างๆของรถพร้อมทั้งคุณสมบัติเฉพาะ เช่น สีรถ ความเร็ว เป็นต้น
3. สร้างส่วนที่ใช้ทำงาน (Method) ของรถ เช่น การวิ่ง เลี้ยว เป็นต้น
4. สร้างตัวอย่างรถ (Instance) ขึ้นมาเพื่อใช้ทดสอบการทำงานต่าง ๆ
5. แก้ไขปรับปรุงในส่วนที่ยังไม่สมบูรณ์

3.6.2 **ขั้นตอนการทำงานของรถ**

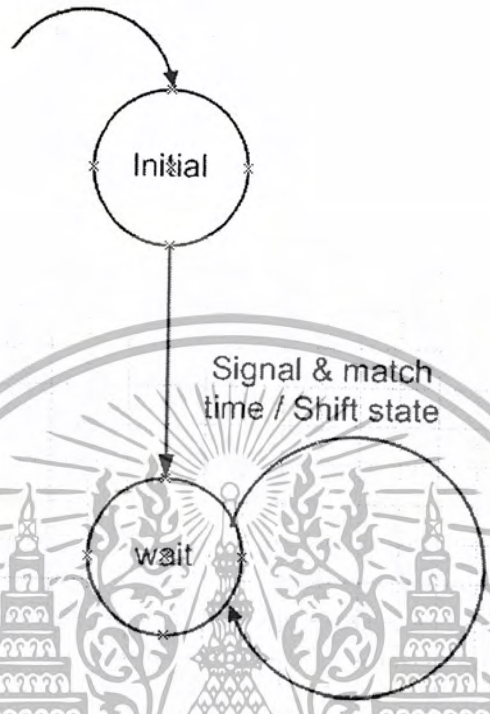
1. เมื่อรถถูกสร้างขึ้นมาในตอนแรกจะมีการกำหนดค่าสถานะเริ่มต้นต่าง ๆ เช่น จุดเริ่มต้นของรถ ความเร็ว
2. เมื่อรถถูกเรียกให้ทำงานก็จะมีการตรวจสอบว่าถึงปลายทางหรือยัง ถ้ายังไม่ถึงปลายทางก็จะต้องมีการเคลื่อนที่ไปข้างหน้า แต่ก่อนที่จะเคลื่อนที่ไปได้จะต้องตรวจสอบพื้นที่ข้างหน้าเสียก่อนว่าบริเวณที่จะเคลื่อนที่ไปนั้นมีรถคันหน้าอยู่หรือไม่และเป็นบริเวณสี่แยกที่มีสัญญาณไฟจราจรหรือไม่ ถ้ามีสิ่งใดสิ่งหนึ่งตามที่กล่าวมาแล้ว ก็จะต้องหยุดรอ ตามเงื่อนไข ต่าง ๆ ก่อน
3. เมื่อรถถึงปลายทางก็จะมีตัวแปรที่ใช้เก็บสถานะสิ้นสุดการทำงานของรถถูกตั้งค่าใหม่ เพื่อให้ระบบรู้ว่ารถคันนี้สิ้นสุดการทำงานแล้วจะได้ลบออกจากระบบ



รูป 3.4 แสดง state diagram ของรถ

3.7 สัญญาณไฟจราจร

ในโปรแกรมนี้เพื่อให้ง่ายต่อการดำเนินงานจึงได้ออกแบบสัญญาณไฟจราจรให้มีเฉพาะสัญญาณไฟเขียวและแดงเท่านั้นโดยได้ตั้งค่าการเปลี่ยนแปลงสัญญาณไว้ที่ 30 วินาที state diagram การทำงานของสัญญาณไฟจราจรแสดงดังรูปที่ 3.5



รูปที่ 3.5 แสดง state diagram ของสัญญาณไฟจราจร

3.8 การออกแบบและสร้างส่วนติดต่อผู้ใช้ (GUI)

ในการออกแบบอินเตอร์เฟสของโปรแกรมกับผู้ใช้จะกำหนดให้มีส่วนที่เป็น หน้าต่างแสดงผลหลัก ซึ่งภายในหน้าต่างแสดงผลหลักประกอบไปด้วย แถบเมนูด้านบน แถบเครื่องมือ (Toolbar) ซึ่งจะใช้เป็นเครื่องมือสร้างองค์ประกอบต่าง ๆ ของการจำลองสภาพการจราจร และพื้นที่ส่วนแสดงผล ซึ่งมีรายละเอียดของส่วนประกอบต่าง ๆ ดังนี้

3.8.1 แถบเมนูด้านบน ประกอบไปด้วย

- เมนู File มีเมนูย่อย New, Open, Save, SavaAs, Close, Exit
- เมนู Edit มีเมนูย่อย Undo, Rotate
- เมนู Setting
- เมนู Simulate เมนูย่อย Run, Stop
- เมนู Result

- เมนู Help

3.8.2 แถบเครื่องมือ (Toolbar)

- ปุ่มสร้างถนนทางตรง
- ปุ่มสร้างถนนสี่แยก
- ปุ่มสร้างจุดปล่อยรถ
- ปุ่มสร้างจุดปลายทาง
- ปุ่มแสดงกริด

3.8.3 พื้นที่การแสดงผล

พื้นที่การแสดงผลจะเป็นส่วนที่ใช้แสดง Graphics ที่ทำการสร้างและสถานการณ์ขณะทำการจำลอง สำหรับการสร้างอินเตอร์เฟสเริ่มจากการสร้างหน้าต่างหลักขึ้นมาเพื่อใช้แสดงส่วนต่าง ๆ ของโปรแกรมทั้งหมดโดยใช้ คลาส JFrame หลังจากนั้นแบ่งพื้นที่แสดงผลภายในโดยใช้ JPanel เป็นContent pane แบ่งส่วนต่าง ๆ แล้วจึงนำแถบเมนูและเครื่องมือต่าง ๆ บรรจุลงใน Content pane ก็จะได้อินเตอร์เฟสตามการออกแบบข้างต้น สำหรับส่วนที่เป็นพื้นที่แสดงผล จะมีการนำ internalframe มาใช้ร่วมด้วย

3.9 การสร้างภาพและการแสดงผลภาพ

สำหรับการสร้างภาพต่าง ๆ ก็จะเป็นการใช้ คำสั่งในการสร้าง Graphics พื้นฐาน แล้วนำการวาดภาพแบบต่าง ๆ มารวมกันให้ได้รูปร่างลักษณะ ของ วัตถุที่ต้องการ โดยวัตถุแต่ละชนิดจะมีการเก็บค่าคุณสมบัติต่าง ๆ เช่น ตำแหน่งของตน ลักษณะการวาดภาพ และคุณสมบัติอื่น ๆ ตามชนิดของวัตถุนั้น ๆ หลังจากที่ได้มีการสร้างรูปแบบการวาดและเก็บลักษณะการวาดแล้ว ในการแสดงผล ก็จะใช้การ Override method repaint(); ของ Panel ที่เป็น super class ซึ่ง repaint() จะไปเรียกใช้ method update(); และ paint(); อีกทีหนึ่ง โดยการนำคำสั่งการวาดภาพต่าง ๆ ไปไว้ใน method repaint(); แต่การ repaint() แต่ละครั้ง จะวาดภาพใหม่ทั้งหมดของ Panel ดังนั้นภาพเก่าที่อยู่ก่อนหน้าถ้าไม่มีการเก็บค่าแล้วนำมาวาดใหม่ พร้อมภาพใหม่ ก็จะถูกลบไป ในการวาดภาพแต่ละครั้งจึงอาศัยการนำลักษณะการวาดต่าง ๆ เก็บไว้ในตัวแปรเวคเตอร์ ซึ่งมีลักษณะคล้าย ๆ อาร์เรย์ เวลาแสดงผลจะเรียกค่าต่าง ๆ ออกมาจากเวคเตอร์จนหมด แล้วจึงสิ้นสุดการ paint();

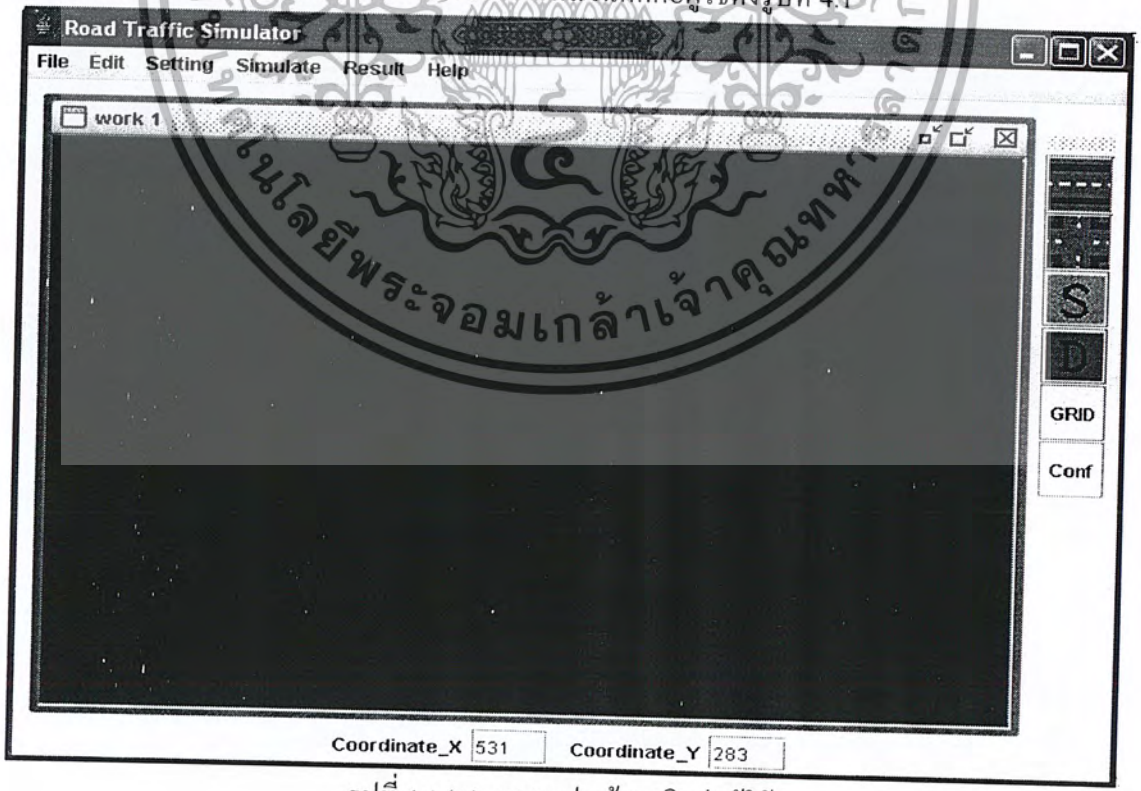
บทที่ 4
ตัวอย่างผลการดำเนินงานและชิ้นงาน

4.1 รูปแบบของโปรแกรม

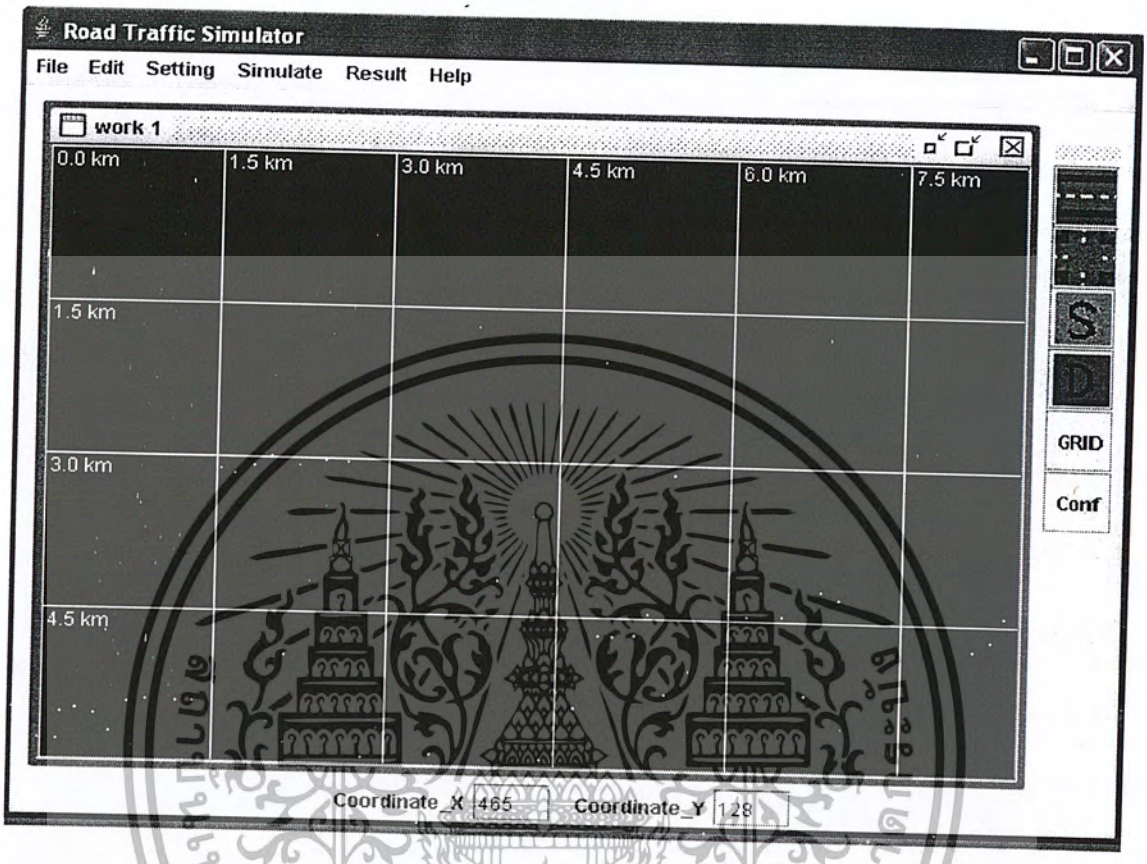
โปรแกรมนี้จะมีการทำงานโดยเริ่มต้นผู้ใช้จะต้องทำการสร้างเส้นทางการจราจรที่จะทำการจำลองพร้อมทั้งจุดที่ใช้อ้างอิงเป็นจุดกำเนิดรถและจุดหมายปลายทาง โดยผู้ใช้สามารถที่จะสร้างเส้นทางและจุดอ้างอิงดังกล่าวโดยใช้แถบเครื่องมือ หลังจากที่ได้สร้างเส้นทางต่าง ๆ ที่จะต้องใช้ในโปรแกรมแล้ว ก่อนที่ผู้ใช้จะทำการเริ่มการจำลองสภาพการจราจร จะมีส่วนที่ให้ผู้ใช้ระบุเงื่อนไขเริ่มต้นที่จะใช้ประกอบการจำลองซึ่งจะอยู่ในส่วนของเมนูต่างๆ จากนั้นก็จะเข้าสู่การจำลองและแสดงผลที่ได้จากการจำลอง

4.2 หน้าแรกของโปรแกรม

เมื่อเริ่มเปิดโปรแกรมจำลองสภาวะการจราจรก็จะมีส่วนติดต่อผู้ใช้ดังรูปที่ 4.1



รูปที่ 4.1 (ก) แสดงรูปหน้าจอติดต่อผู้ใช้

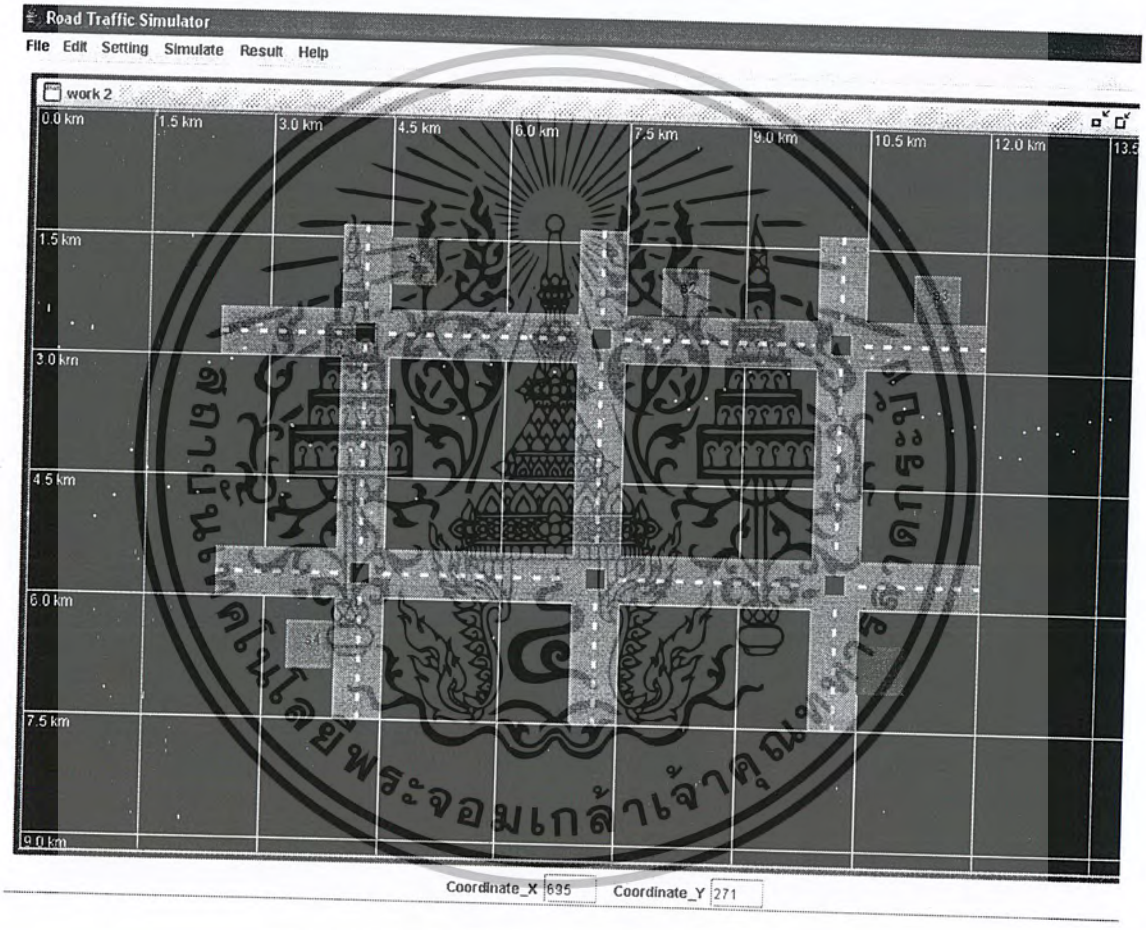


รูปที่ 4.1 (ข) แสดงรูปหน้าจอติดต่อผู้ใช้เมื่อใช้กริด

จากรูปที่ 4.1 แสดงรูปหน้าจอติดต่อผู้ใช้ ด้านบนซ้ายของหน้าจอคือแถบเครื่องมือที่ใช้ในการสร้างเส้นทางการจราจรซึ่งประกอบด้วยส่วนประกอบที่ใช้ในการสร้างถนน ส่วนที่ใช้ในการกำหนดจุดเริ่มต้นและจุดหมายปลายทางในการเดินทาง ด้านขวาของหน้าจอคือแถบเมนูที่ใช้ในการทำคำสั่งและกำหนดค่าต่างๆที่ใช้ในการจำลอง

4.3 การสร้างเส้นทางการจราจร

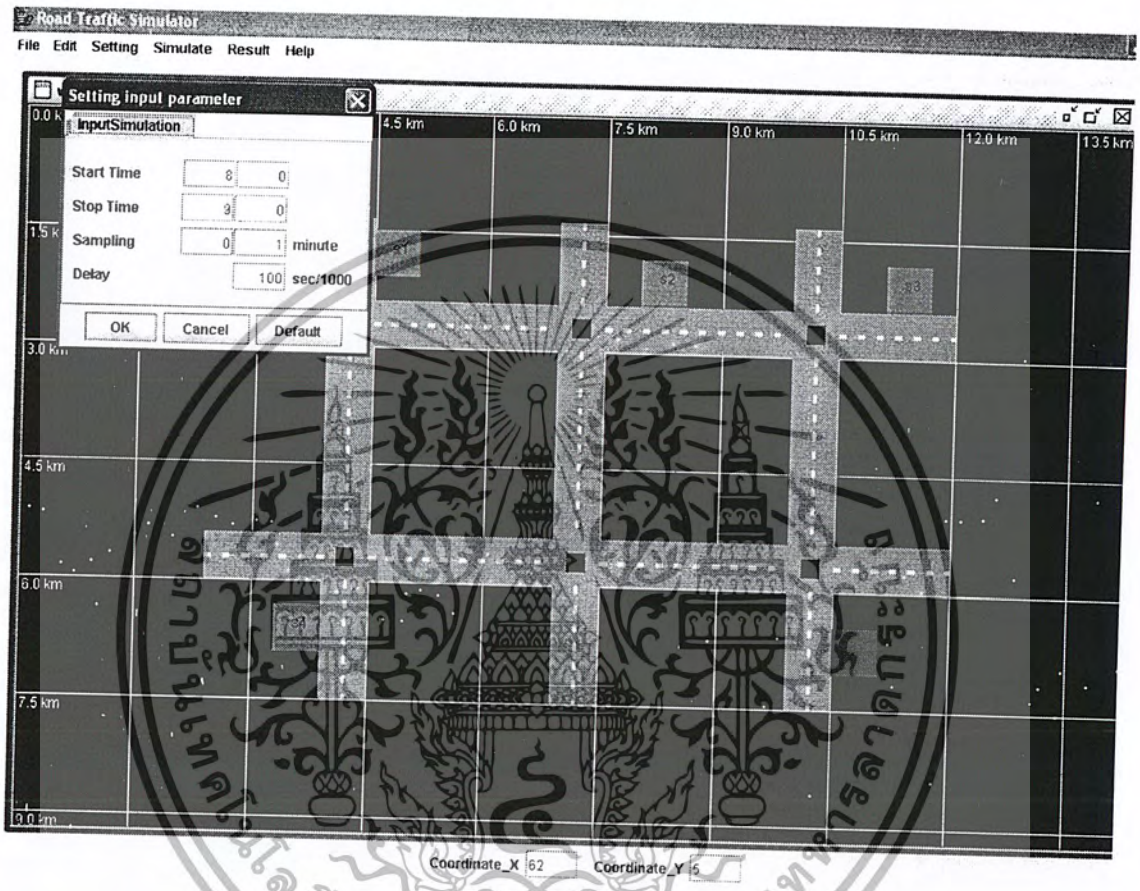
การสร้างเส้นทางการจราจรสามารถสร้างได้จากแถบเครื่องมือ โดยสามารถสร้างเส้นทางการจราจรที่ต้องการทำการจำลองได้ตามต้องการ แล้วทำการกำหนดจุดปล่อยรถและจุดปลายทางได้ตามต้องการ แต่ในโปรแกรมต้นแบบนี้สามารถกำหนดจุดปล่อยรถได้สูงสุด 4 จุด และปลายทางหนึ่งจุด



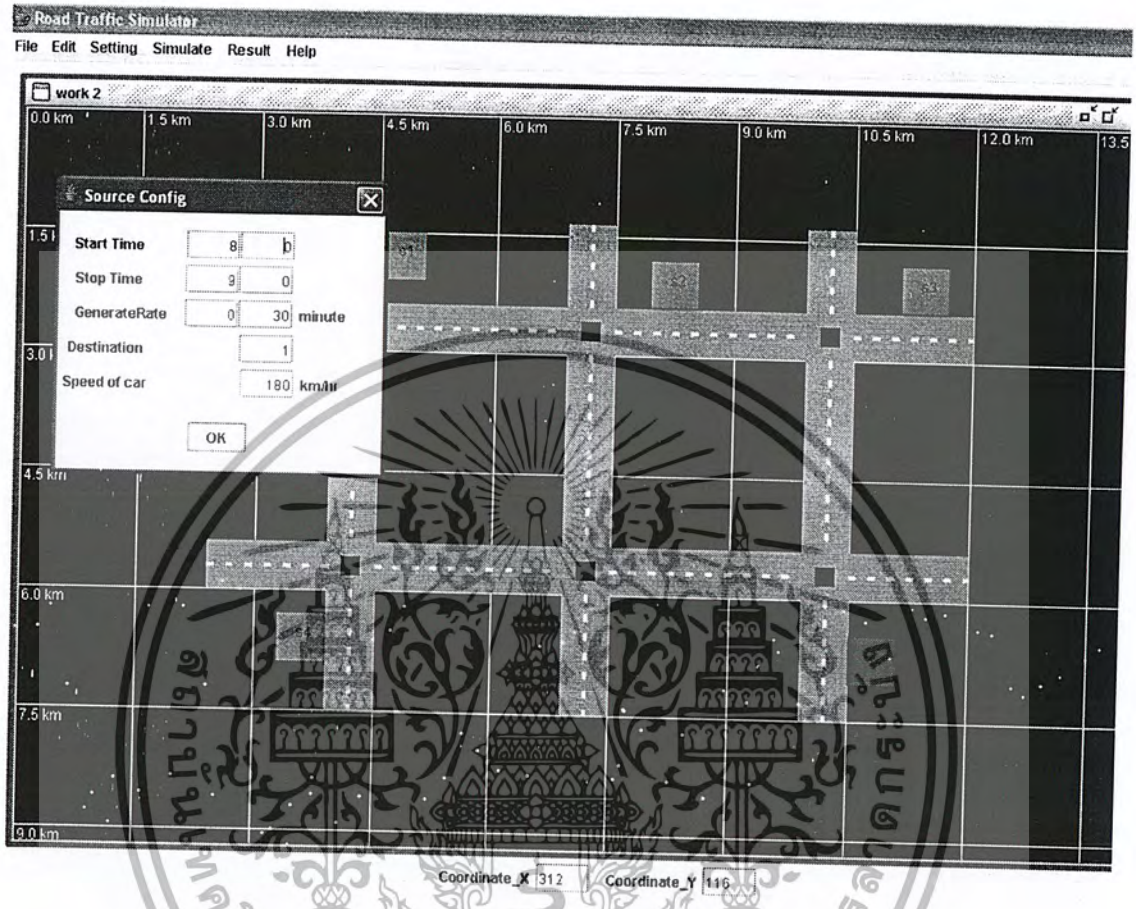
รูปที่ 4.2 แสดงตัวอย่างการสร้างเส้นทางการจราจร

จากรูปที่ 4.2 แสดงตัวอย่างการสร้างเส้นทางเบื้องต้นเป็นตัวอย่างคร่าว ๆ โดยผู้ใช้สามารถสร้างได้โดยใช้เครื่องมือในการสร้างที่มีในแถบเครื่องมือ

4.4 การกำหนดเงื่อนไขในการจำลอง



รูปที่ 4.3 (ก) แสดงการกำหนดเงื่อนไขที่ใช้ในการจำลอง

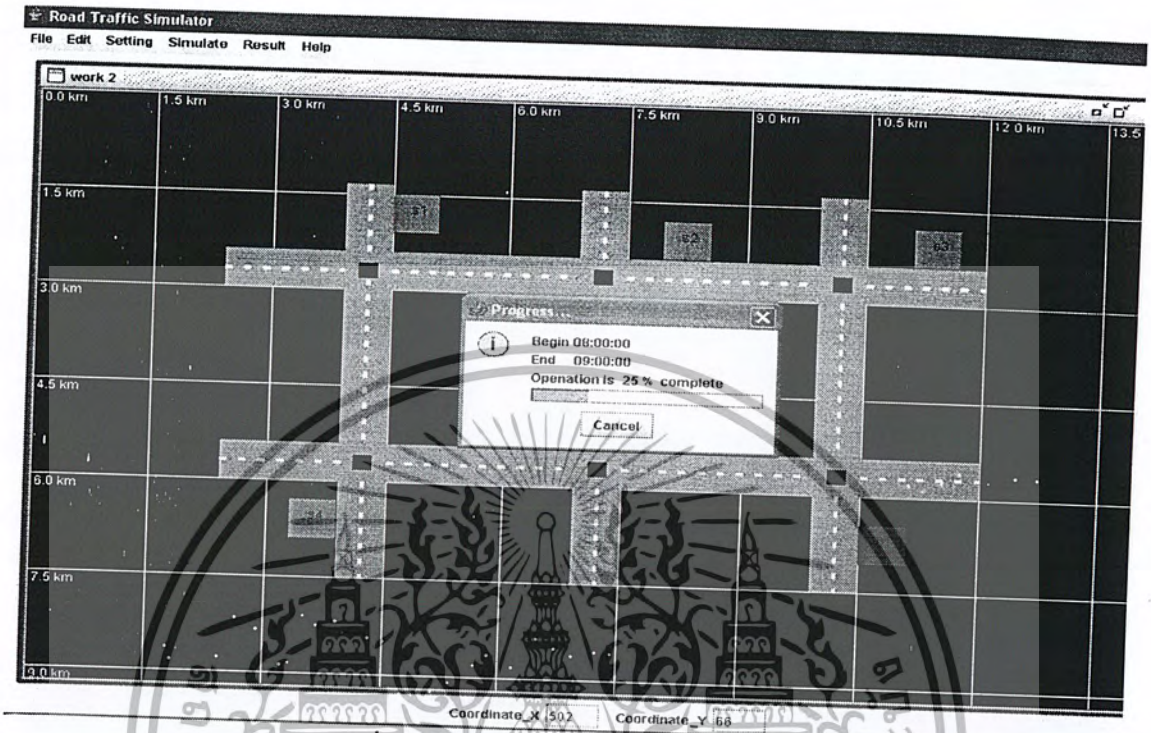


รูปที่ 4.3 (ข) แสดงการกำหนดเงื่อนไขที่ใช้ในการจำลองให้กับแต่ละจุดปล่อยรถ

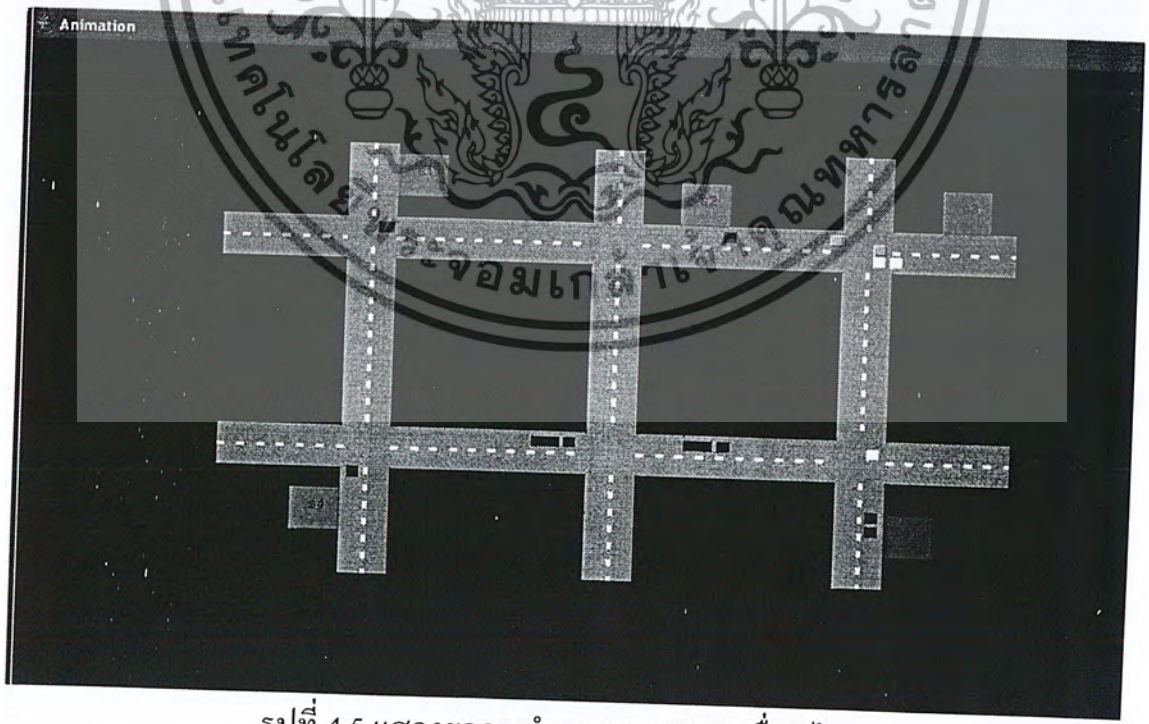
จากรูปที่ 4.3 เป็นการกำหนดเงื่อนไขที่ใช้ในการจำลองซึ่งประกอบด้วยจุดเริ่มต้น จุดหมายปลายทาง อัตราการปล่อยรถ ความเร็วของรถและระยะเวลาที่จะทำการจำลอง โดยผู้ใช้สามารถกำหนดเงื่อนไขได้ตามสถานการณ์ที่ต้องการได้

4.5 ผลที่ได้จากการจำลอง

เมื่อผู้ใช้ทำการจำลองแล้ว ผลที่ได้จากการจำลองจะแสดงออกมาในรูปของสถานะในเวลาต่างๆ ของสภาพการจราจร

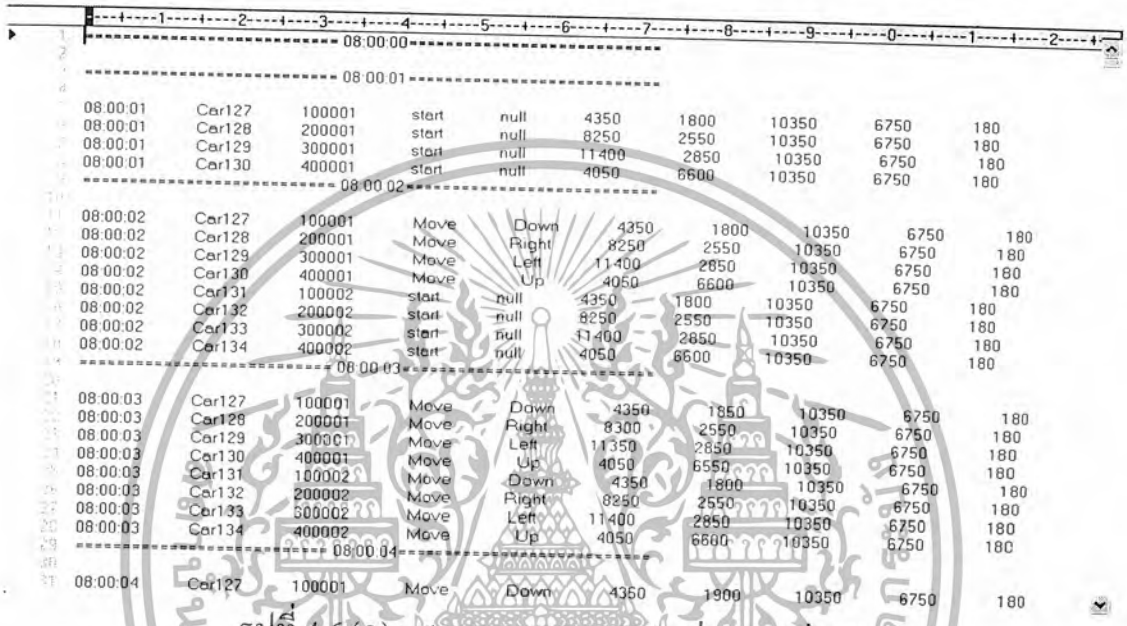


รูปที่ 4.4 แสดงความคืบหน้าการจำลอง



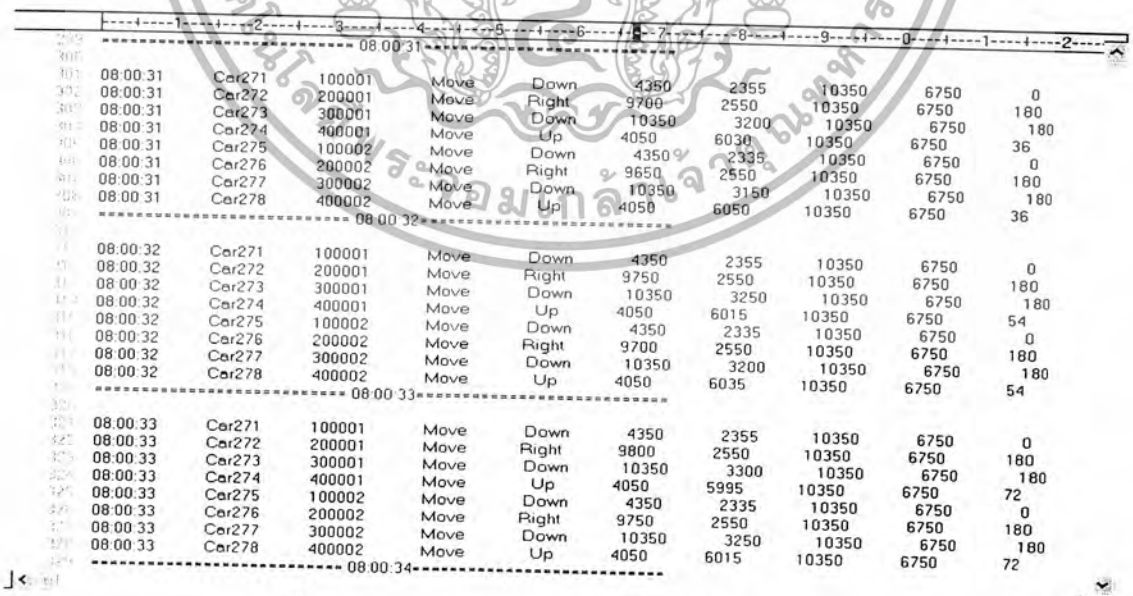
รูปที่ 4.5 แสดงผลการจำลองแบบภาพเคลื่อนไหว

จากรูปที่ 4.5 เป็นตัวอย่างการแสดงผลที่ได้จากการจำลองโดยเลือกแสดงผลแบบภาพเคลื่อนไหว ซึ่งจะแสดงการเคลื่อนที่ของรถที่อยู่บนถนนในขณะเวลาที่เราสนใจ



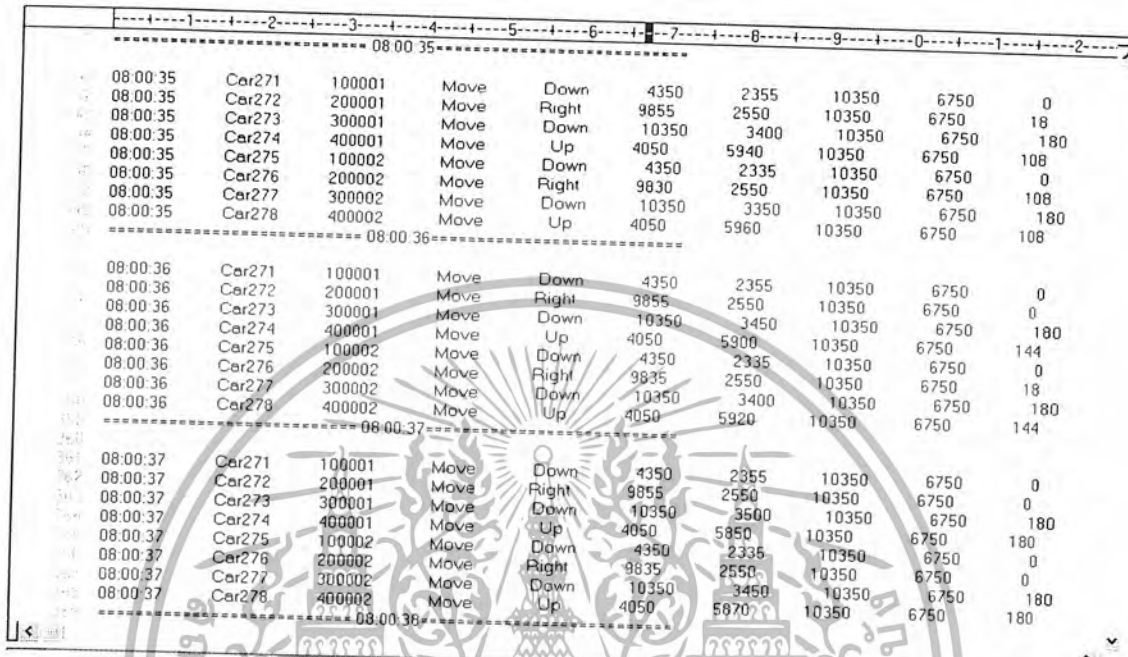
```
08:00:00
08:00:01
08:00:01  Car127  100001  start  null  4350  1800  10350  6750  180
08:00:01  Car128  200001  start  null  8250  2550  10350  6750  180
08:00:01  Car129  300001  start  null  11400  2850  10350  6750  180
08:00:01  Car130  400001  start  null  4050  6600  10350  6750  180
08:00:02
08:00:02  Car127  100001  Move  Down  4350  1800  10350  6750  180
08:00:02  Car128  200001  Move  Right  8250  2550  10350  6750  180
08:00:02  Car129  300001  Move  Left  11400  2850  10350  6750  180
08:00:02  Car130  400001  Move  Up  4050  6600  10350  6750  180
08:00:02  Car131  100002  start  null  4350  1800  10350  6750  180
08:00:02  Car132  200002  start  null  8250  2550  10350  6750  180
08:00:02  Car133  300002  start  null  11400  2850  10350  6750  180
08:00:02  Car134  400002  start  null  4050  6600  10350  6750  180
08:00:03
08:00:03  Car127  100001  Move  Down  4350  1850  10350  6750  180
08:00:03  Car128  200001  Move  Right  8300  2550  10350  6750  180
08:00:03  Car129  300001  Move  Left  11350  2850  10350  6750  180
08:00:03  Car130  400001  Move  Up  4050  6550  10350  6750  180
08:00:03  Car131  100002  Move  Down  4350  1800  10350  6750  180
08:00:03  Car132  200002  Move  Right  8250  2550  10350  6750  180
08:00:03  Car133  300002  Move  Left  11400  2850  10350  6750  180
08:00:03  Car134  400002  Move  Up  4050  6600  10350  6750  180
08:00:04
08:00:04  Car127  100001  Move  Down  4350  1900  10350  6750  180
```

รูปที่ 4.6 (ก) แสดงผลการทดลอง ณ ช่วงเวลาต่าง ๆ



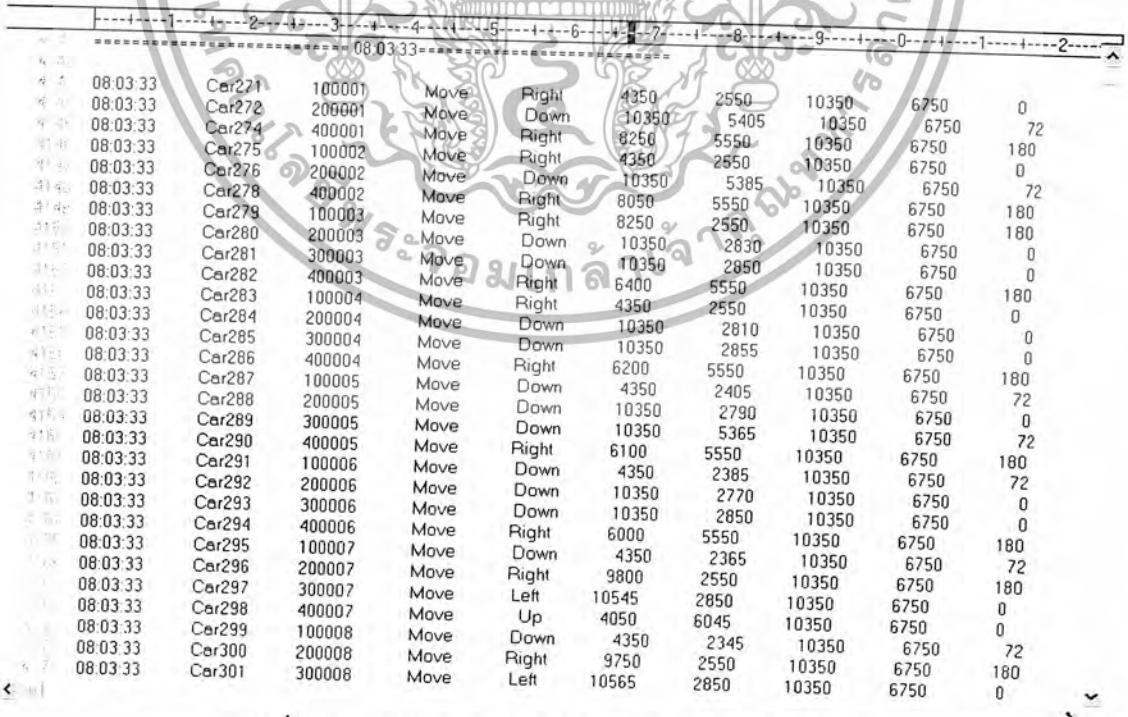
```
08:00:31
08:00:31  Car271  100001  Move  Down  4350  2355  10350  6750  0
08:00:31  Car272  200001  Move  Right  9700  2550  10350  6750  180
08:00:31  Car273  300001  Move  Down  10350  3200  10350  6750  180
08:00:31  Car274  400001  Move  Up  4050  6030  10350  6750  36
08:00:31  Car275  100002  Move  Down  4350  2335  10350  6750  0
08:00:31  Car276  200002  Move  Right  9650  2550  10350  6750  180
08:00:31  Car277  300002  Move  Down  10350  3150  10350  6750  180
08:00:31  Car278  400002  Move  Up  4050  6050  10350  6750  36
08:00:32
08:00:32  Car271  100001  Move  Down  4350  2355  10350  6750  0
08:00:32  Car272  200001  Move  Right  9750  2550  10350  6750  180
08:00:32  Car273  300001  Move  Down  10350  3250  10350  6750  180
08:00:32  Car274  400001  Move  Up  4050  6015  10350  6750  54
08:00:32  Car275  100002  Move  Down  4350  2335  10350  6750  0
08:00:32  Car276  200002  Move  Right  9700  2550  10350  6750  180
08:00:32  Car277  300002  Move  Down  10350  3200  10350  6750  180
08:00:32  Car278  400002  Move  Up  4050  6035  10350  6750  54
08:00:33
08:00:33  Car271  100001  Move  Down  4350  2355  10350  6750  0
08:00:33  Car272  200001  Move  Right  9800  2550  10350  6750  180
08:00:33  Car273  300001  Move  Down  10350  3300  10350  6750  180
08:00:33  Car274  400001  Move  Up  4050  5995  10350  6750  72
08:00:33  Car275  100002  Move  Down  4350  2335  10350  6750  0
08:00:33  Car276  200002  Move  Right  9750  2550  10350  6750  180
08:00:33  Car277  300002  Move  Down  10350  3250  10350  6750  180
08:00:33  Car278  400002  Move  Up  4050  6015  10350  6750  72
08:00:34
```

รูปที่ 4.6 (ข) แสดงผลการทดลอง ณ ช่วงเวลาต่าง ๆ



08:00:35									
08:00:35	Car271	100001	Move	Down	4350	2355	10350	6750	0
08:00:35	Car272	200001	Move	Right	9855	2550	10350	6750	18
08:00:35	Car273	300001	Move	Down	10350	3400	10350	6750	180
08:00:35	Car274	400001	Move	Up	4050	5940	10350	6750	108
08:00:35	Car275	100002	Move	Down	4350	2335	10350	6750	0
08:00:35	Car276	200002	Move	Right	9830	2550	10350	6750	108
08:00:35	Car277	300002	Move	Down	10350	3350	10350	6750	180
08:00:35	Car278	400002	Move	Up	4050	5960	10350	6750	108
08:00:36									
08:00:36	Car271	100001	Move	Down	4350	2355	10350	6750	0
08:00:36	Car272	200001	Move	Right	9855	2550	10350	6750	0
08:00:36	Car273	300001	Move	Down	10350	3450	10350	6750	180
08:00:36	Car274	400001	Move	Up	4050	5900	10350	6750	144
08:00:36	Car275	100002	Move	Down	4350	2335	10350	6750	0
08:00:36	Car276	200002	Move	Right	9835	2550	10350	6750	18
08:00:36	Car277	300002	Move	Down	10350	3400	10350	6750	180
08:00:36	Car278	400002	Move	Up	4050	5920	10350	6750	144
08:00:37									
08:00:37	Car271	100001	Move	Down	4350	2355	10350	6750	0
08:00:37	Car272	200001	Move	Right	9855	2550	10350	6750	0
08:00:37	Car273	300001	Move	Down	10350	3500	10350	6750	180
08:00:37	Car274	400001	Move	Up	4050	5850	10350	6750	180
08:00:37	Car275	100002	Move	Down	4350	2335	10350	6750	0
08:00:37	Car276	200002	Move	Right	9835	2550	10350	6750	0
08:00:37	Car277	300002	Move	Down	10350	3450	10350	6750	180
08:00:37	Car278	400002	Move	Up	4050	5870	10350	6750	180

รูปที่ 4.6 (ค) แสดงผลการทดลอง ณ ช่วงเวลาต่าง ๆ



08:03:33									
08:03:33	Car271	100001	Move	Right	4350	2550	10350	6750	0
08:03:33	Car272	200001	Move	Down	10350	5405	10350	6750	72
08:03:33	Car274	400001	Move	Right	8250	5550	10350	6750	180
08:03:33	Car275	100002	Move	Right	4350	2550	10350	6750	0
08:03:33	Car276	200002	Move	Down	10350	5385	10350	6750	72
08:03:33	Car278	400002	Move	Right	8050	5550	10350	6750	180
08:03:33	Car279	100003	Move	Right	8250	2550	10350	6750	180
08:03:33	Car280	200003	Move	Down	10350	2830	10350	6750	0
08:03:33	Car281	300003	Move	Down	10350	2850	10350	6750	0
08:03:33	Car282	400003	Move	Right	6400	5550	10350	6750	180
08:03:33	Car283	100004	Move	Right	4350	2550	10350	6750	0
08:03:33	Car284	200004	Move	Down	10350	2810	10350	6750	0
08:03:33	Car285	300004	Move	Down	10350	2855	10350	6750	0
08:03:33	Car286	400004	Move	Right	6200	5550	10350	6750	180
08:03:33	Car287	100005	Move	Down	4350	2405	10350	6750	72
08:03:33	Car288	200005	Move	Down	10350	2790	10350	6750	0
08:03:33	Car289	300005	Move	Down	10350	5365	10350	6750	72
08:03:33	Car290	400005	Move	Right	6100	5550	10350	6750	180
08:03:33	Car291	100006	Move	Down	4350	2385	10350	6750	72
08:03:33	Car292	200006	Move	Down	10350	2770	10350	6750	0
08:03:33	Car293	300006	Move	Down	10350	2850	10350	6750	0
08:03:33	Car294	400006	Move	Right	6000	5550	10350	6750	180
08:03:33	Car295	100007	Move	Down	4350	2365	10350	6750	72
08:03:33	Car296	200007	Move	Right	9800	2550	10350	6750	180
08:03:33	Car297	300007	Move	Left	10545	2850	10350	6750	0
08:03:33	Car298	400007	Move	Up	4050	6045	10350	6750	0
08:03:33	Car299	100008	Move	Down	4350	2345	10350	6750	72
08:03:33	Car300	200008	Move	Right	9750	2550	10350	6750	180
08:03:33	Car301	300008	Move	Left	10565	2850	10350	6750	0

รูปที่ 4.6 (ง) แสดงผลการทดลอง ณ ช่วงเวลาต่าง ๆ

บทที่ 5
สรุปผลการดำเนินงานและวิจารณ์

5.1 สรุปผลการดำเนินงาน

จากผลการดำเนินงานเป็นไปตามเป้าหมายที่ตั้งไว้ ซึ่งถึงแม้จะใช้เวลาในการวิเคราะห์และแก้ไขปัญหาต่าง ๆ ค่อนข้างนานเนื่องมาจากปัญหาในการหาเส้นทางที่ต้องนำวิธีการมาดัดแปลงให้เข้ากับโปรแกรมที่ใช้งานได้ค่อนข้างจะยุ่งยากและซับซ้อน แต่ในที่สุดแล้วก็สามารถที่จะให้โปรแกรมทำงานได้ตามที่คาดหวังไว้ในตอนแรก

5.2 ปัญหาในการดำเนินงาน

ในขั้นแรกของการออกแบบ จากความต้องการหลักของระบบที่จะต้องให้ผลจากการทำการจำลองนี้สามารถที่จะบันทึกค่าหรือวิเคราะห์ข้อมูลที่ได้จากการทำการจำลองของระบบ ซึ่งจากปัญหาเบื้องต้น ทำให้ผู้ออกแบบนึกถึงส่วนแรกที่รับจากผู้ใช้โปรแกรมก่อนว่าผู้ใช้สามารถทำอะไรกับโปรแกรมได้บ้าง แล้วเราจะนำค่าจากสิ่งที่ผู้ใช้ป้อนหรือกำหนดในส่วนของการสร้างเส้นทางหรือสภาวะแวดล้อมต่าง ๆ นั้น มาใช้ในการพิจารณาเพื่อกำหนดเงื่อนไขในการตัดสินใจทำในส่วนต่าง ๆ อย่างไร ซึ่งจากการนำปัญหาในส่วนเกิดจากการป้อนค่าหรือกำหนดค่าของผู้ใช้ว่ามีผลอย่างไรกับโปรแกรมนั้น ทำให้ต้องเสียเวลาเป็นอย่างมากในการที่จะคิดวิธีหรือเงื่อนไขที่จะนำมากำหนดความเป็นไปได้ของการทำงานในส่วนแรก ซึ่งหลังจากที่คิดรูปและเงื่อนไขที่จะตอบสนองกับผู้ใช้ในส่วนแรกของโปรแกรมได้แล้วก็มีปัญหาตามมาอีก คือเมื่อได้ค่าต่าง ๆ แล้วเราจะนำค่านั้นไปใช้ในการทำการจำลองอย่างไร เช่น ความเร็วของรถที่ได้จากผู้ใช้ จะต้องนำมาคำนวณหรือมีวิธีจัดการอย่างไรเพื่อให้สัมพันธ์กับการทำงานของโปรแกรม และปัญหาที่พบอีกปัญหาที่เป็นส่วนที่ทำให้การออกแบบระบบล่าช้า ก็เนื่องมาจากปัญหาในการหาเส้นทางในการที่จะไปให้ถึงปลายทางของรถให้ได้ ซึ่งมีอัลกอริทึมในการหาเส้นให้เลือกอยู่หลายวิธี แต่จะนำวิธีเหล่านั้นมาประยุกต์ใช้กับระบบที่กำลังออกแบบได้อย่างไร ซึ่งในการวิเคราะห์ในส่วนนี้ เป็นส่วนที่ค่อนข้างจะเป็นปัญหาใหญ่ ของระบบเลยทีเดียวก็ว่าได้ เพราะถ้ายังไม่สามารถหาเส้นทางที่จะไปหรือทิศทางในการเคลื่อนที่ได้แล้ว ก็จะมองภาพรวมของระบบไม่ออก ทำให้เป็นปัญหาในระบบการทำงานหลัก คือการทำการจำลองระบบการจราจร ซึ่งจะต้องมีการ

เคลื่อนที่ของรถตามเส้นทาง แต่ถ้าไม่มีเส้นทางให้รถเคลื่อนที่ไปแล้ว ก็จะไม่สามารถทดสอบ รูปแบบและวิธีการที่ใช้ในการทำการจำลองได้เพราะระบบไม่สมบูรณ์

5.3 ข้อจำกัดของโครงงาน

1. ข้อจำกัดในด้านของสภาพแวดล้อม ซึ่งผู้ใช้งานโปรแกรมจะไม่สามารถสร้างรูปแบบถนนที่ใกล้เคียงความเป็นจริงได้ เนื่องมาจากถนนที่มีอยู่จริงนั้น มีปัญหาให้พิจารณาเยอะ ทำให้เป็นปัญหาในการพิจารณาทิศทางการเคลื่อนที่ของโปรแกรม
2. ข้อจำกัดในด้านของการวิเคราะห์ระบบจากโปรแกรม ซึ่งในส่วนนี้โปรแกรมเป็นแต่เพียงแค่เครื่องมือในการแปลงความต้องการเบื้องต้นของผู้ใช้ให้ได้ ผลออกมาเป็นข้อมูลตามวิธีของโปรแกรม แต่ยังไม่สามารถแก้ไขปัญหาต่าง ๆ ที่เกิดขึ้นจากสภาพการจราจรที่เห็นได้ ปัญหาในส่วนนี้ผู้จำลองจำเป็นต้องตั้งปรับค่าพารามิเตอร์ต่าง ๆ ให้เหมาะสมเองเพื่อให้ได้ค่าจากการจำลองที่เหมาะสมตามความต้องการของผู้ใช้

5.4 แนวทางในการพัฒนาต่อ

ควรที่จะทำให้โปรแกรมสามารถที่จะวิเคราะห์ข้อมูลที่ได้จากการทำการจำลองสภาพการจราจรในแต่ละครั้ง พร้อมทั้งนำข้อมูลเดิมเก็บไว้เพื่อนเป็นสถิติเพื่อนใช้ในการวิเคราะห์ต่อไป

บรรณานุกรม

[1] มานพ วราภักดิ์, "การจำลองเบื้องต้น", ศูนย์ผลิตตำราเรียน สถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ, พิมพ์ครั้งที่ 1 พ.ศ. 2547

[2] วีรศักดิ์ ซึงถาวร, "Fundamental of JAVA Programming Volume 1", บริษัทซีเอ็ดยูเคชั่น, พิมพ์ครั้งที่ 1 พ.ศ. 2546

[3] ศิริจันทร์ ทองประเสริฐ, "การจำลองแบบปัญหา", โรงพิมพ์จุฬาลงกรณ์มหาวิทยาลัย, พ.ศ. 2535