ระบบจำลองการควบคุมจราจรทางรถไฟ

The Railway Traffic Simulation

โดย

นางสาว นิศากาญจน์ นิปริย์าย

รหัส 43067052

อาจารย์ที่ปรึกษา

ดร.จันทร์บูรณ์ สถิตวิริยวงศ์

รายงานนี้เป็นส่วนหนึ่งของวิชาโครงการพัฒนาระบบงาน

หลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ

ภาคเรียนที่ 1 ปีการศึกษา 2545

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ชื่อหัวข้อ	ระบบจำลองการควบคุมจราจรทางรถไฟ
นักศึกษา	นิศากาญจน์ นิปริยาย
อาจารย์ที่ปรึกษา	ดร.จันทร์บูรณ์ สถิตวิริยวงศ์
ระดับการศึกษา	วิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
แขนงวิชา	วิทยาการสารสนเทศ
ปีการศึกษา	2545

บทคัดย่อ

ระบบจำลองการควบคุมจราจรทางรถไฟ มีจุดประสงค์ในการพัฒนาออกแบบเพื่อใช้ในการนำเสนอและจำลองการควบคุมจราจรทางรถไฟ เพื่อให้ผู้ใช้สามารถทำความเข้าใจ ฝึกหัด และศึกษาเรียนรู้การทำงานของระบบรถไฟ และการจัดการควบคุมจราจรทางรถไฟ โดยระบบจำลองประกอบด้วย การทดลองการควบคุมดูแลการจราจร การแสดงข้อมูลและสถานะของอุปกรณ์ต่างๆ ที่อยู่ในเส้นทางรถไฟ การจัดการคำสั่งต่างๆ ที่ได้รับจากผู้ใช้ และการตอบสนองต่อเหตุการณ์ต่างๆ ที่เกิดขึ้นในเส้นทาง หรือในแต่ละอุปกรณ์ รวมทั้งการติดต่อสื่อสารกับส่วนโปรแกรมที่ใช้สำหรับการประมวลผล การแสดงผลของระบบจำลองนำเสนอในรูปแบบกราฟิกซึ่งสามารถตอบสนองและโต้ตอบกับผู้ใช้ได้ตลอดเวลา

Title	The Railway Traffic Simulation
Student	Nisakarn Nipariyai
Advisor	Dr.Chanboon Satitwiriyawongs
Level of Study	Master of Science in Information Technology
Major	Information Science
Academic Year	2002

Abstract

The Railway Traffic Simulation is the system that simulates the local control and supervisory system for interlocking devices of the railway traffic. The simulation operation consists of displaying information and status of objects, handling of the commands given by the operation, event response and alarm handling, train route setting and single object control, and communication with the interlocking logic. The simulation is based on the interactive graphical presentation connecting with the interlocking logic and developed for presentation, training and practicing user in order to understand and apply the simulation to the practical railway traffic system.

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

หน้า

ตารางที่

สารบัญภาพ

หน้า

รูปที่

บทที่ 1
บทนำ

การโดยสารรถไฟ เป็นทางเลือกหนึ่งสำหรับการเดินทางที่ได้รับความนิยม เนื่องจากมีความปลอดภัยสูง เสียค่าใช้จ่ายน้อย เข้าถึงแหล่งชุมชน และสามารถขนส่งได้ในปริมาณมาก ในปัจจุบันแนวโน้มในการเดินทางโดยใช้รถไฟเพิ่มสูงขึ้น อีกทั้งภายในประเทศและต่างประเทศมีการพัฒนาโครงการรถไฟหลายโครงการอย่างต่อเนื่อง ทั้งที่เป็นการสร้างเส้นทางใหม่ และการปรับปรุงโครงสร้างและพัฒนาระบบของเส้นทางเดิม จึงทำให้บุคคลที่มาเกี่ยวข้องกับระบบรถไฟมีจำนวนเพิ่มสูงขึ้น ทั้งในส่วนของผู้ปฏิบัติงาน และผู้ที่ทำงานด้านออกแบบและพัฒนาระบบรถไฟ ดังนั้นการเรียนรู้การทำงานของระบบรถไฟจึงมีความสำคัญอย่างมาก ที่จะทำให้สามารถใช้งานและพัฒนาระบบรถไฟที่มีประสิทธิภาพและมีความปลอดภัย

เนื่องจากระบบรถไฟเป็นระบบที่มีความซับซ้อน และเป็นลักษณะความรู้เฉพาะทางที่ต้องเรียนรู้จากการทำงานจริง ดังนั้นผู้ที่ต้องทำงานเกี่ยวข้องกับระบบรถไฟ จะต้องเรียนรู้การทำงานของระบบต่างๆเป็นอย่างดี เพื่อให้เกิดความเข้าใจเพื่อที่จะสามารถพัฒนาและออกแบบระบบได้อย่างมีประสิทธิภาพ ซึ่งต้องใช้ระยะเวลานานในการเรียนรู้ การศึกษาครั้งนี้จึงต้องการพัฒนาระบบจำลองการควบคุมจราจรทางรถไฟ หรือ The Railway Traffic Simulation เพื่อใช้ในการนำเสนอและจำลองการควบคุมจราจรทางรถไฟ โดยแสดงผลในรูปแบบกราฟิก และสามารถโต้ตอบกับผู้ใช้ได้ คล้ายคลึงกับเกมทางคอมพิวเตอร์ ทำให้ผู้ใช้สามารถทำความเข้าใจและเรียนรู้หลักการทำงานของระบบได้อย่างรวดเร็ว รวมทั้งเพิ่มความน่าสนใจในการศึกษาและทดลองปฏิบัติเพื่อสามารถประยุกต์ใช้กับระบบการทำงานจริง นอกจากนี้สามารถนำระบบจำลองไปใช้ในส่วนของการนำเสนอเพื่อประชาสัมพันธ์ระบบการทำงานของการจัดจราจรทางรถไฟได้อีกทางหนึ่ง

1.1 วัตถุประสงค์

ออกแบบและพัฒนาระบบจำลองการควบคุมจราจรทางรถไฟ (The Railway Traffic Simulation) โดยผลของการพัฒนาเป็นรูปแบบของซอฟต์แวร์ทำงานบนระบบปฏิบัติการ Windows 95 ขึ้นไป ที่ประกอบด้วย DirectX8.0 Runtime หรือสูงกว่า

1.2 ขอบเขตของระบบจำลอง

การพัฒนาระบบจำลองอยู่บนพื้นฐานของระบบ Signaling System และระบบ Interlocking Logic System ของบริษัท Bombardier Transportation (Thailand) โดยจำลองการทำงานมาจากระบบต้นแบบ Interlocking Logic Package for New South Wales ซึ่งพัฒนาตามมาตรฐานที่กำหนดโดย Railway Associate Cooperation (RAC) ประเทศออสเตรเลีย

การทำงานของระบบการควบคุมจราจรทางรถไฟ ประกอบด้วย

1. แสดงหลักการทำงานโดยรวมของระบบ Signaling System
2. การจัดการควบคุมดูแลการจราจร และสร้างเส้นทางรถไฟ
3. การแสดงข้อมูลและสถานะของอุปกรณ์ต่างๆ
4. การจัดการกับคำสั่งต่างๆ ที่ได้รับจากผู้ใช้
5. การตอบสนองต่อเหตุการณ์ต่างๆ ที่เกิดขึ้นในเส้นทาง
6. ประมวลผลโดยใช้หลักการของ Interlocking Logic System
7. การแสดงผลของระบบในรูปแบบกราฟิก โดยผ่าน DirectX Runtime

1.3 ประโยชน์ที่คาดว่าจะได้รับ

1. ใช้ในการนำเสนอ และจำลองการควบคุมจราจรทางรถไฟ โดยแสดงผลในรูปแบบกราฟิก และสามารถโต้ตอบกับผู้ใช้ได้ คล้ายคลึงกับเกมทางคอมพิวเตอร์
2. ผู้ใช้สามารถทำความเข้าใจและเรียนรู้หลักการทำงานของระบบการควบคุมจราจรทางรถไฟได้อย่างรวดเร็ว
3. เพิ่มความน่าสนในการศึกษาและทดลองปฏิบัติเพื่อผู้ใช้สามารถนำไปประยุกต์ใช้กับระบบการทำงานจริง
4. ใช้ในการนำเสนอเพื่อประชาสัมพันธ์ระบบการทำงานของการจัดจราจรทางรถไฟ

1.4 ขั้นตอนการพัฒนา

1. ศึกษาหลักการทำงานของระบบ Signaling System
2. ศึกษาหลักการทำงานของระบบ Interlocking Logic System
3. ศึกษาความต้องการของระบบต้นแบบที่จะนำมาใช้ในการทำระบบจำลอง
4. กำหนดขอบเขตและความต้องการของระบบจำลอง
5. วิเคราะห์ความต้องการและออกแบบระบบ
6. เลือกใช้และศึกษาเครื่องมือที่จะนำมาใช้ในการพัฒนา
7. ทำการพัฒนาระบบจำลองตามที่ออกแบบไว้

8. ทำการทดสอบระบบและปรับปรุงแก้ไขระบบเพื่อให้สามารถทำงานได้ตามขอบเขตที่กำหนดไว้ได้อย่างถูกต้องตรงกับความต้องการ
9. สรุปผลการศึกษา และเขียนรายงานรวมทั้งจัดทำคู่มือการใช้งาน

1.5 เครื่องมือที่ใช้ในการพัฒนา

1. ฮาร์ดแวร์ (Hardware)

- CPU AMD Duron 1.1 GHz
- RAM 256 MB
- S3 Compatible Display Adapter 16MB
- Digital Flat Panel (1024 x 768)

2. ซอฟต์แวร์ (Software)

- Windows Xp Home Edition
- Microsoft Visual C++ 6.0
- DirectX8.0 SDK and DirectX8.0 Runtime
- Adobe PhotoShop5.5

1.6 แผนการพัฒนา

แผนการพัฒนาเป็นไปตามขั้นตอนการพัฒนา โดยใช้ระยะเวลาดำเนินการทั้งสิ้นประมาณ 5 เดือน

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

2.1 การพัฒนาของ Railway Signaling

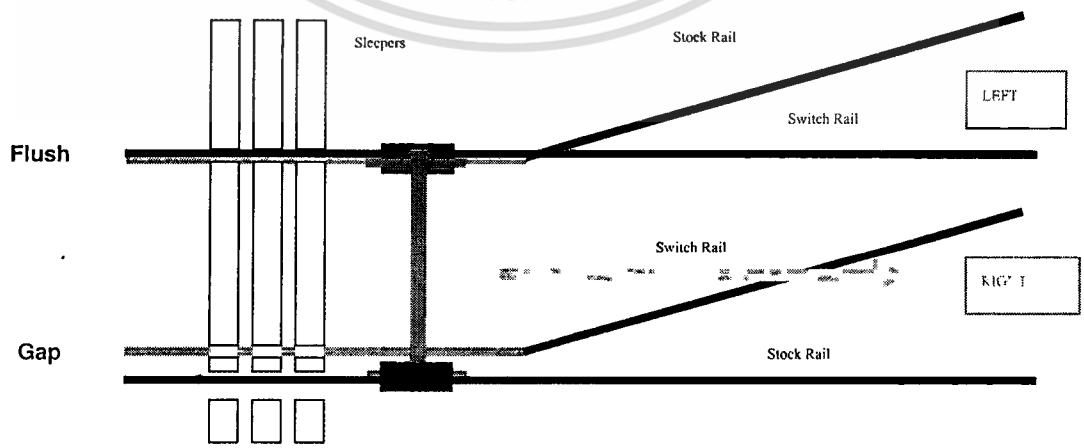
เนื่องจากรถไฟมีขนาดใหญ่ มีน้ำหนักมากและวิ่งด้วยความเร็วสูง ดังนั้นการบังคับการขับเคลื่อนของรถไฟ ต้องอาศัยปัจจัยหลายอย่างมาพิจารณาเพื่อให้เกิดความปลอดภัยมากที่สุด จึงเกิดหลักการของ Railway Signaling เพื่อใช้ในการออกแบบระบบการจัดจราจรรถไฟที่มีประสิทธิภาพและทำให้เกิดความปลอดภัย จุดประสงค์หลักของ Railway Signaling ประกอบด้วย

2.1.1 Avoiding Obstacles

คือ การหลีกเลี่ยงสิ่งกีดขวาง ในกรณีของรถไฟเนื่องจากต้องวิ่งไปตามราง ดังนั้นการหลีกเลี่ยงสิ่งกีดขาวงสามารถทำได้แค่เพียงลดความเร็วเท่านั้น ซึ่งโดยปกติรถไฟจะใช้ระยะเวลานานและระยะทางไกลในการหยุดรถ ดังนั้นจึงเป็นการยากที่คนขับรถไฟจะสามารถหยุดได้ทันเมื่อสังเกตเห็นสิ่งกีดขวาง ดังนั้นหน้าที่ของ Railway Signaling คือ ต้องออกแบบให้มีช่วง "obstacle-free" อยู่ด้านหน้าของขบวนรถไฟเสมอ ซึ่งต้องมีระยะทางไกลเพียงพอสำหรับระยะเบรกของความเร็วรถไฟ ณ ขณะนั้น

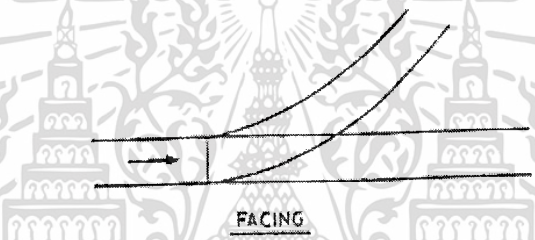
2.1.2 Junction

คือ ทางแยกของรางรถไฟ (junction of track) เนื่องจากรถไฟไม่สามารถเลี้ยวซ้ายขวาได้ตามใจชอบ แต่ต้องเลี้ยวไปตามทางที่กำหนดโดย junction of track เท่านั้น



รูปที่ 2.1 A train taking the right straight-on route

เมื่อพิจารณารูปที่ 2.1 ส่วนประกอบต่างๆ จะประกอบด้วย ส่วนที่ยึดติดอยู่กับที่ เรียกว่า “stock rails” ซึ่งยึดติดกับส่วนของหมอนรองรางรถไฟ (sleepers) เสมอ และส่วนที่สามารถเคลื่อนที่ได้ เรียกว่า “switch rails” ในขณะที่รถไฟเคลื่อนที่ผ่าน junction, switch rail ด้านหนึ่งจะเคลื่อนที่มาชิดติดกับ stock rail ของเส้นทางที่ต้องการให้รถไฟเคลื่อนที่ไป ส่วน switch rail อีกด้านหนึ่งจะเว้นระยะห่างจาก stock rail ของเส้นทางที่ไม่ได้เลือก ดังนั้นการเลือกเส้นทางว่าจะไปทางด้านซ้ายหรือด้านขวาจะกระทำโดย switch rails เท่านั้น จากรูปที่ 2.1 แสดงว่าเลือกเส้นทางไปทางด้านขวา โดยอุปกรณ์ที่ใช้สำหรับ switch เลือกเส้นทางเรียกว่า point ซึ่งแบ่งออกได้เป็นสองลักษณะตามทิศทางที่วิ่งเข้าสู่ junction ของรถไฟ คือ facing points และ trailing points



รูปที่ 2.2 Facing points give a choice of diverging routes

รูปที่ 2.3 Trailing points join 2 converging routes

ในกรณีของ facing point สิ่งที่สำคัญที่สุดคือ ต้องมั่นใจในขณะที่รถไฟวิ่งผ่าน point จะต้องไม่มีการ switch point ณ ขณะนั้น ไม่เช่นนั้นรถไฟสามารถตกราง (de-railed) ได้

2.1.3 Functions

Signaling system ประกอบด้วย 3 main functions

1. ป้องกันการชนของรถไฟที่วิ่งในเส้นทางเดียวกัน และทิศทางเดียวกัน โดยจะต้องรักษาระยะห่างระหว่างรถไฟให้อยู่ในระยะ Safe Distance
2. การเคลื่อนอย่างปลอดภัยของรถไฟบริเวณ Junction เมื่อเส้นทางต้องมีการตัดกับเส้นทางของรถไฟขบวนอื่น และจะต้องป้องกันไม่ให้มีการเคลื่อนที่ของรถไฟขบวนอื่นในทิศทางตรงกันข้ามสวนเข้ามาในเส้นทาง ดังนั้น ข้อ 1 และ 2 เป็น Functions การทำงานที่สำคัญที่สุดที่ต้องรักษาให้ถูกต้องไว้ตลอดเวลาเพื่อไม่ให้เกิดอุบัติเหตุ ทั้งสองข้อนี้เรียกว่า Vital Functions
3. เป็นขั้นตอนการควบคุมดูแล (Supervisory Process) ซึ่งเป็นส่วนหนึ่งของ Signaling System ทำหน้าที่ในการจัดการให้การวิ่งของขบวนรถไฟเป็นไปตามตารางที่กำหนดไว้ ในส่วนนี้เรียกว่า Non-Vital Function คือเป็น Function ที่ถ้าเกิดผิดพลาดก็ไม่ได้ทำให้เกิดอันตราย แต่จำเป็นต้องมี เพื่อให้การทำงานเป็นอย่างราบรื่น รวดเร็ว และเป็นระบบ

2.2 วิวัฒนาการของ Railway Signaling

ระบบ Railway Signaling มีการวิวัฒนาการเปลี่ยนแปลงตลอดมาตั้งแต่ ศตวรรษที่ 15 จนถึงระบบสมัยใหม่ที่ใช้กันอยู่ในปัจจุบัน จุดประสงค์ที่สำคัญในการพัฒนาคือ สามารถตอบสนองต่อ Vital Function และ Non – Vital Function ได้อย่างมีประสิทธิภาพมากขึ้น และสามารถรองรับระบบรถไฟที่มีความเร็วเพิ่มสูงขึ้น ต้นแบบระบบ Railway Signaling ในปัจจุบันเริ่มพัฒนาตั้งแต่ต้น ศตวรรษที่ 19 มีดังนี้

2.2.1 Signalboxes

เป็นการย้ายการควบคุม Signals และ Points มาไว้ในที่เดียวกันเพื่อสะดวกในการทำงานของพนักงานรถไฟ ทำให้สามารถควบคุมได้ดีขึ้นใช้เวลาและจำนวนคนน้อยลง และป้องกันความผิดพลาดได้มากขึ้น ซึ่งห้องควบคุมจะเรียกว่า Signalboxes จะประกอบด้วย Levers จำนวนมาก ทั้ง Levers ในส่วนของที่ควบคุมป้ายสัญญาณ (Fixed Signal) และ Levers สำหรับเปลี่ยนทิศทางของอุปกรณ์ Points

2.2.2 Interlocking

เป็นการจัดกลุ่ม Levers ของ Signal และ Point เข้าด้วยกัน ทำให้สามารถ Interlock กันได้ เช่น ขณะที่ Levers ของ Signal อยู่ในตำแหน่งท่าอนุญาตให้รถไฟผ่านไปได้ Levers ของ Points ที่เกี่ยวข้องจะถูก lock เอาไว้ไม่ให้มีการเคลื่อนที่ การใช้หลักการนี้เป็นการเพิ่มความปลอดภัยขึ้นอย่างมาก รวมทั้งลดความผิดพลาดที่อาจจะเกิดขึ้นจากมนุษย์ และป้องกันความเสียหายของอุปกรณ์

ต่างๆ จากสภาพที่ไม่ปลอดภัย ซึ่งหลักการ Interlocking เป็นพื้นฐานของออกแบบระบบ Signaling System ในปัจจุบัน

2.3 Signaling System ในปัจจุบัน

ระบบ Signaling System ในปัจจุบันประกอบด้วยส่วนต่างๆ ดังนี้

2.3.1 Signal

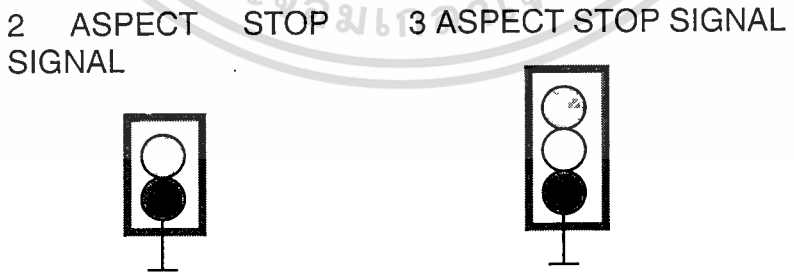
Signal หรือไฟสัญญาณ ทำหน้าที่ควบคุมการวิ่งของรถไฟ ซึ่งพนักงานขับรถไฟสามารถมองเห็นสัญญาณไฟได้จากระยะไกล ทั้งในเวลากลางวันและกลางคืน แต่ละดวงไฟของ Signals เรียกว่า Aspect อย่างไรก็ตาม Signal ที่ใช้ในระบบรถไฟมีหลายประเภท ขึ้นอยู่กับมาตรฐานของแต่ละประเทศ โดยสามารถแบ่งออกเป็นประเภทหลักๆ ได้ดังนี้

1. **Main Running Signal**

เป็น Control Signal ที่จะต้องถูก Interlocking กับอุปกรณ์อื่นๆ ซึ่งจะต้องถูกควบคุมเพื่อให้สามารถแสดง Proceed Aspect หรือไฟสัญญาณที่อนุญาตให้รถไฟผ่านไปได้ โดยมีรายละเอียดการทำงาน ดังนี้

- Stop Signals

เป็น Signal ที่สามารถแสดง Stop Aspect เพื่อต้องการให้รถไฟหยุด ไม่ให้เคลื่อนที่ผ่าน Signal เข้าไป โดยแต่ละ Aspect จะเป็นสีต่างๆ คล้ายกับไฟจราจรโดยทั่วไป โดยสีแดงหมายถึง stop หรือให้หยุด ส่วนสีเหลือง และสีเขียวเป็น proceed aspect หรืออนุญาตให้รถไฟผ่านเข้าไปได้ ซึ่งโดยทั่วไป Stop Signals แบ่งได้เป็น 2 ประเภท ดังรูป



รูปที่ 2.4 Stop Signal

2 Aspect: สีแดงหมายถึงให้หยุด และสีเขียวหมายถึงให้รถไฟผ่านไปได้ ซึ่งจะใช้ Signal แบบนี้กับเสาสัญญาณที่อนุญาตรถไฟให้วิ่งออกไปจากสถานี หรือเรียกว่า Starter Signal

3 Aspect: สีแดงหมายถึงให้หยุด สีเหลืองเป็นไฟสัญญาณเตือนว่าให้พนักงานขับรถไฟเตรียมตัวลดความเร็วลงเพราะไฟสัญญาณข้างหน้าจะเป็นสีแดงซึ่งต้องการให้รถไฟหยุด และสีเขียวคืออนุญาตให้รถไฟผ่านไปได้ด้วยความเร็วปกติ การใช้ไฟสัญญาณแบบ 3 Aspect เป็นการรวม stop และ Warner Signal เข้าไว้ด้วยกันในสัญญาณไฟต้นเดียว ดังนั้นสามารถใช้เป็นไฟสัญญาณสำหรับอนุญาตให้รถไฟเข้ามาภายในสถานี หรือเรียกว่า Home Signal, ใช้เป็นไฟสัญญาณเตือน หรือ Warner Signal รวมทั้งในบางครั้งสามารถใช้เป็น Starter Signal ได้ด้วยในกรณีที่ระบบการจัดการจราจรมีความซับซ้อนมากขึ้น

ค่าปกติของ stop signal จะเป็นสีแดงเสมอเพื่อความปลอดภัย ดังนั้นเมื่อต้องการให้ signal แสดงสัญญาณไฟอนุญาต หรือ proceed aspect ระบบจะต้องตรวจสอบเงื่อนไขต่างๆ ว่าถูกต้องจึงจะสามารถแสดงสัญญาณไฟอนุญาตได้ และหลังจากแสดง proceed aspect แล้ว ถ้ามีเงื่อนไขเงื่อนไขหนึ่งไม่ถูกต้องครบถ้วน สัญญาณไฟจะเปลี่ยนค่าไปเป็นสีแดงทันที ดังนั้นเงื่อนไขสำหรับการแสดง proceed aspect จะต้องถูกต้องตลอดเวลา

- Warner Signal

เป็นไฟสัญญาณเตือน ประกอบด้วย aspect สีเหลืองและสีเขียว ดังนั้นจะไม่มีการหยุดรถไฟเมื่อเห็นสัญญาณไฟชนิดนี้ ทำหน้าที่ในการเตือนพนักขับรถให้ลดความเร็วลงเมื่อเห็นสัญญาณไฟสีเหลือง เนื่องจากสัญญาณไฟข้างหน้าต้องการให้รถไฟหยุดหรือแสดง aspect สีแดง แต่ถ้าเป็นสีเขียวหมายความว่าสัญญาณไฟข้างหน้าเป็น proceed aspect ดังนั้นสามารถใช้ความเร็วปกติได้

การทำงานของ Warner Signal จะตั้งค่าปกติ ไว้ที่ สัญญาณไฟสีเหลือง และเมื่อต้องการแสดงสัญญาณไฟสีเขียวจะต้องตรวจสอบเงื่อนไขที่ต้องการให้ถูกต้องตลอดเวลา ถ้าเงื่อนไขใดเงื่อนไขหนึ่งไม่เป็นจริง ก็จะเปลี่ยนสัญญาณไฟเป็นสีเหลืองทันที

2 ASPECT WARNER SIGNAL

รูปที่ 2.5 "Warner" signal tells driver to start braking

- Lamp Failure

เมื่อเกิดเหตุการณ์ Lamp Failure หรือเหตุการณ์ที่ไฟสัญญาณไม่สามารถแสดงสัญญาณไฟตามที่ระบบ Signaling System ต้องการได้ เช่น ไฟสัญญาณแตก เป็นต้น ระบบจะต้องสามารถตรวจสอบได้ว่าการทำงานของไฟสัญญาณยังเป็นปกติหรือไม่ และถ้าเกิดสิ่งผิดปกติขึ้นจะต้องหาวิธีจัดการเพื่อไม่ให้เกิดอันตราย และยังคงปลอดภัย เช่น ไฟสัญญาณแตกจนไม่สามารถแสดงสัญญาณไฟได้เลย หรือเรียกว่า dark ระบบจะสั่งให้ไฟสัญญาณอันก่อนหน้าเปลี่ยนมาแสดงค่า Stop Aspect แทน ดังรูป

รูปที่ 2.6 "Stop" signal tells driver to stop

- Aspect Sequence

Aspect Sequence คือลำดับในการแสดงสัญญาณไฟของเสาสัญญาณที่อยู่ในทิศทางเดียวกันบนเส้นทางรถไฟเดียวกัน เพื่อที่พนักงานขับรถไฟสามารถเตรียมตัวปรับความเร็วให้เหมาะสมกับตามที่ Signaling System ต้องการ ซึ่งจะทำให้เกิดความปลอดภัย โดยหลักการพื้นฐานของ Aspect Sequence คือ ถ้าไฟสัญญาณเป็นสีเหลือง แสดงว่าเสาสัญญาณข้างหน้าเป็นสีแดง ดังนั้นพนักงานขับรถต้องเตรียมตัวลดความเร็วลงเพื่อที่จะสามารถหยุดรถได้ทัน แต่ถ้าไฟสัญญาณเป็นสีเขียวแสดงว่าเสาสัญญาณข้างหน้าอนุญาตให้รถไฟเคลื่อนที่ผ่านไปได้ซึ่งอาจจะเป็นสัญญาณไฟสีเหลืองหรือสีเขียวก็ได้ ดังนั้นพนักงานขับรถสามารถใช้ความเร็วปกติได้โดยไม่ต้องเตรียมตัวชะลอความเร็ว

นอกจาก Main Running Signal แล้ว ก็ยังมี signal ในลักษณะอื่นๆ อีก แต่เนื่องจากในการศึกษาครั้งนี้เลือกใช้เฉพาะ Main Route ดังนั้นจะขอกล่าวถึง signal ประเภทอื่นเพียงเล็กน้อย

2. Subsidiary Signal

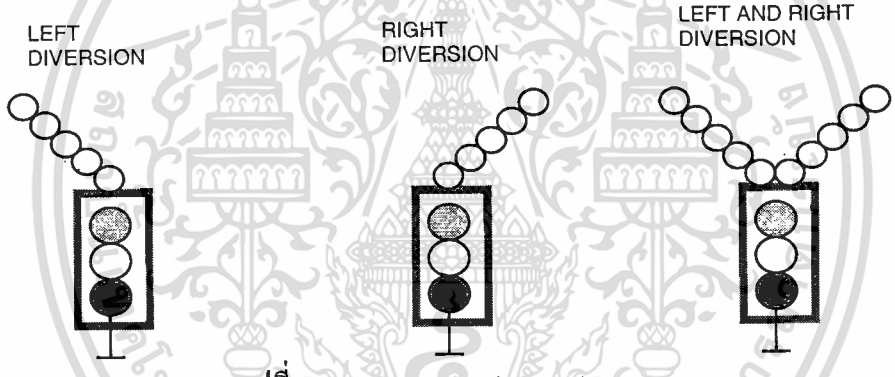
เป็น signal ที่อยู่ภายในสถานี ซึ่งอนุญาตให้มีการเคลื่อนของรถไฟเข้ามาในเส้นทางโดยที่เส้นทางไม่จำเป็นต้อง clear แต่ต้องใช้ความเร็วต่ำ โดยทั่วไปสถานีที่มีความซับซ้อนจำเป็นต้องใช้ signal ประเภทนี้เสมอ เนื่องจากสำหรับเคลื่อนที่รถไฟระหว่าง Main Line กับ Sidings หรือ Loop

Lines ซึ่งการเคลื่อนที่ประเภทนี้จะไม่มีผู้โดยสารอยู่ในขบวนรถ เช่น การต่อขบวนรถไฟ, การถ่ายเทสินค้า, การซ่อมบำรุง เป็นต้น

Subsidiary Signal แบ่งได้เป็น 2 แบบ คือ “call-on” และ “shunt ahead” ซึ่งมีจุดประสงค์ในการใช้ต่างกันคือ call-on มักใช้ร่วมกับ home signal ในกรณีที่ home signal มีปัญหาไม่สามารถแสดง proceed signal ได้ ก็จะใช้ call-on แทน เมื่อพนักงานขับรถเห็นสัญญาณไฟแบบ call-on ก็จะรู้ว่าสามารถขับรถไฟผ่านไปได้แต่ต้องใช้ความเร็วต่ำ เพื่อความปลอดภัย ส่วน shunt ahead จะใช้กับเส้นทางอื่นๆ ที่อยู่ภายในสถานีเพื่อย้ายรถไฟจากระหว่าง main line กับ loop lines

3. Junction Indicator

ทำหน้าที่แสดงสัญญาณบอกทิศทางของ junction โดยทั่วไปจะอยู่ติดกับสัญญาณไฟที่อยู่หน้า junction เพื่อที่พนักขับรถจะได้ทราบว่ารถไฟจะวิ่งไปในทิศทางไหน ตัวอย่าง Junction Indicator ดังรูปที่ 2.8

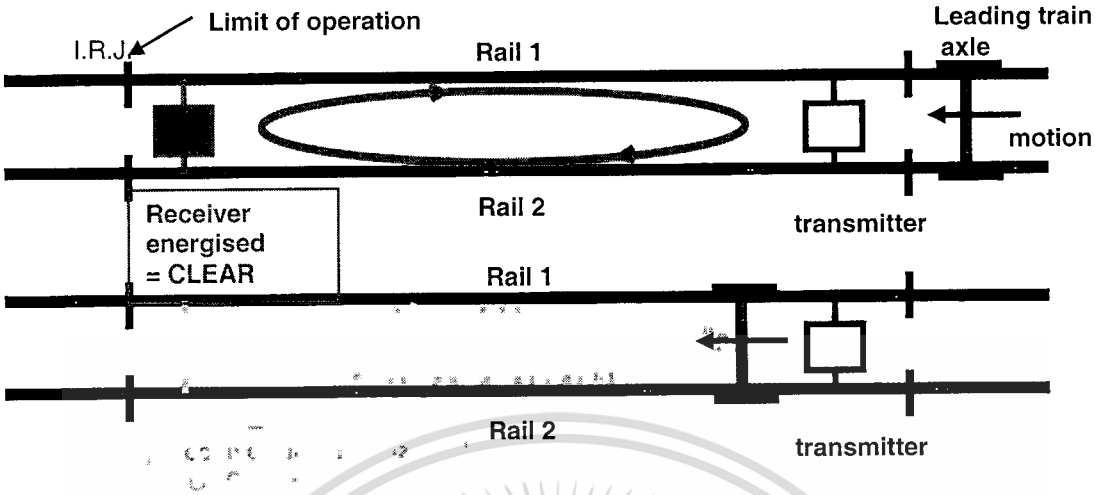


รูปที่ 2.7 Junction Indicator Aspects

2.3.2 Track Circuits

เป็นอุปกรณ์ที่อยู่ติดกับรางรถไฟ เพื่อที่ใช้ตรวจสอบว่ามีรถไฟเข้ามาในเส้นทางหรือไม่ ซึ่งจะแสดง 2 สถานะ คือ

- Track Clear เมื่อไม่มีรถไฟ หรือสิ่งกีดขวางอื่นๆ อยู่บน Track Circuit ของรางรถไฟช่วงนั้น
- Track Occupied จะแสดงสถานะนี้เมื่อมีรถไฟหรือสิ่งกีดขวางอื่นอยู่บน Track Circuit ของรางรถไฟช่วงนั้น นอกจากมีรถไฟหรือสิ่งกีดขวางแล้วอาจรวมถึงการทำงานผิดปกติของ Track Circuit หรือที่เรียก Track Failed ซึ่งจะรวมในสถานะนี้ด้วย



รูปที่ 2.8 Operation of Track Circuits

การติดตั้ง Track Circuit สามารถทำได้หลายตำแหน่ง โดยทั่วไปจะทำการติดตั้งระหว่างรางรถไฟ ซึ่งจะแบ่งออกเป็นช่วงๆ ตามจำนวนของ Track Circuit ตลอดเส้นทางของรางรถไฟ

หลักการทำงาน พิจารณารูปที่ 2.8 ใช้หลักการของ Relay ที่ใช้ low voltage power supply โดยทั่วไปประมาณ 1.5 – 2.0 volt และ Track relay มีความต้านทานอยู่ที่ประมาณ 4 – 20 ohm เพื่อให้สามารถ detect สิ่งกีดขวางอื่นๆ ที่ไม่ใช่ขบวนรถไฟ ซึ่งอาจทำจากวัสดุที่มีค่าการนำไฟฟ้าต่ำ รวมทั้งกระแสไฟฟ้าจะไม่ส่งผลกระทบต่อขบวนรถไฟ เพราะมีค่าต่ำมาก

ส่วนประกอบของ Track Circuit จะประกอบด้วย Transmitter ทำหน้าที่เป็นแหล่งกำเนิดส่งกระแสไฟฟ้าไปยังวงจร Receiver ซึ่งเมื่อได้รับกระแสไฟฟ้าก็จะครบวงจร Receiver อยู่ในสถานะ Energized และส่งสัญญาณไปบอกกับระบบควบคุมว่าขณะนี้ Track Circuit ช่วงนี้ Clear ในทางกลับกันเมื่อมีรถไฟวิ่งเข้ามาใน Track Circuit จะทำให้เกิด Short Circuit ขึ้น คือ กระแสไฟฟ้าจาก Transmitter จะวิ่งมายังขบวนรถไฟแทนที่จะเป็น Receiver ทำให้ Receiver อยู่ในสภาวะ de-energized ซึ่งเป็นการส่งสัญญาณบอกว่าขณะนี้ Track Circuit ถูก Occupied อยู่

โดยทั่วไปความยาวของรางรถไฟสำหรับ Track Circuit ในแต่ละช่วง จะอยู่ที่ 20 – 1000 เมตร ส่วนช่วงใดจะต้องใช้ความยาวเท่าใด ขึ้นอยู่กับหลักการออกแบบทางด้านวิศวกรรมโครงสร้างของระบบรถไฟ ซึ่งมีหลายปัจจัยที่ต้องนำมาพิจารณา โดยปัจจัยหลักที่สำคัญที่สุดคือเรื่องความปลอดภัย

Track Circuit จะทำหน้าที่ Interlocking กับอุปกรณ์อื่นดังนี้

- Interlocking จะไม่อนุญาตให้ signal แสดง proceed aspect ถ้า Track Circuits ที่เกี่ยวข้องกับ signal อันนั้น ไม่อยู่ในสถานะ Clear โดยปกติ Track Circuits ที่เกี่ยวข้องจะหมายถึง

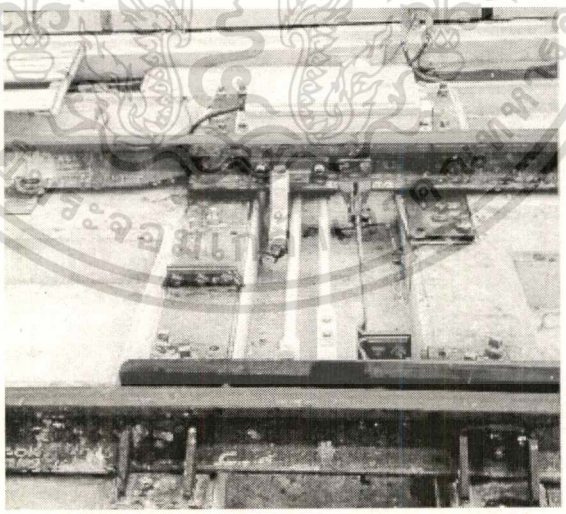
Track Circuit ที่อยู่ภายในเส้นทางของ signal ที่ต้องการให้แสดง proceed aspect แต่อาจจะมีกรณีที่รวมถึง Track Circuit ในส่วนอื่นที่ไม่ได้ในเส้นทาง จะเรียกว่า Flank Section

- Interlocking จะไม่อนุญาตให้ switch point ถ้า Track Circuit ของ point อันนั้น ไม่อยู่ในสถานะ Clear

2.3.3 Points

Point เป็นอุปกรณ์ที่ทำหน้าที่เลือกเส้นทางของ junction โดยจะทำหน้าที่ switch ส่วนที่เป็น switch rails ดังที่ได้กล่าวในตอนต้น ในปัจจุบันเป็นควบคุมการทำงานโดยใช้ไฟฟ้าและมอเตอร์ ซึ่งชื่อเรียกโดยทั่วไปคือ Electric Point Machine โดยมอเตอร์ที่ทำหน้าที่ switch point จะได้รับคำสั่งให้ทำงานจากระบบควบคุม

ส่วนประกอบของ Point จะถูกติดตั้งกับส่วนของ switch rails และ stock rails ส่วนที่ติดกับ switch rail จะทำหน้าที่ขับเคลื่อนเมื่อต้องการเปลี่ยนตำแหน่ง ส่วนที่ติดกับ stock rails จะทำหน้าที่เป็น detector ตรวจสอบว่า Point เคลื่อนที่มายังตำแหน่งที่ถูกต้องหรือยัง เมื่อ detect ได้ว่าถูกตำแหน่งแล้วก็จะส่งสัญญาณไปบอกระบบควบคุม ระบบควบคุมก็จะส่งคำสั่งให้มอเตอร์หยุดหมุน รวมทั้งเก็บค่าสถานะของ Point เอาไว้ โดย detector จะทำหน้าที่ส่งสถานะไปบอกระบบควบคุมตลอดเวลา ดังนั้นระบบควบคุมจะรู้ความเป็นไปของอุปกรณ์ได้ตลอด



รูปที่ 2.9 Point Machine

2.4 Computer Based Interlocking

Computer Based Interlocking เป็นเทคโนโลยีล่าสุดที่ใช้ทดแทน Mechanical Interlocking และ Electrical Relay Interlocking โดยอาศัยหลักการเดียวกันกับ Electrical Relay Interlocking แต่ใช้ Programmed logic เข้ามาจัดการ interlocking function ทั้งหมด โดย Functions หลักๆ ประกอบด้วย

2.4.1 Route Setting

การที่จะให้รถไฟเคลื่อนที่ในแต่ละครั้งจะต้องมีการจัดจราจรให้ ไม่สามารถเคลื่อนที่ได้ตามใจชอบ เพื่อรักษาความปลอดภัยให้เป็นไปตาม vital functions 2 ข้อข้างต้น ดังนั้นจึงต้องมีการสร้างเส้นทางสำหรับการเคลื่อนที่ของรถไฟในแต่ละครั้งเรียกว่า Route Setting โดย dispatcher จะเป็นส่งคำสั่ง (Maneuver) เข้าสู่ระบบ Interlocking ซึ่งจะรับค่า Maneuver และตรวจสอบเงื่อนไขอื่นๆ ของอุปกรณ์ต่างๆ ภายในเส้นทางรถไฟ (Yard Object) ว่ามีสถานะพร้อมที่จะ Set Route หรือไม่ ถ้าเงื่อนไขทุกอย่างถูกต้องตาม Interlocking Logic ที่ได้ทำการออกแบบไว้ ก็สามารถดำเนินการ Route Setting และแสดงสัญญาณไฟ proceed aspect อนุญาตให้รถไฟเข้ามาในเส้นทางได้ต่อไป

จากที่กล่าวมาระบบ Interlocking โดยทั่วไปจะต้องมีการตรวจสอบเงื่อนไขต่างๆ ดังนี้

- ไม่มี conflicting routes set
- Points อยู่ใน correct position หรือ free to be moved (not locked)

ถ้าเงื่อนไขเหล่านี้ถูกต้อง สามารถดำเนินการในขั้นตอน Route Locked จากนั้นต้องตรวจสอบเงื่อนไขต่อไปนี้

- Track Circuits ทุกตัวที่อยู่ระหว่าง entrance signal (start signal) และ exit signal (end signal) จะต้อง clear
- Track Circuits ทุกตัวที่อยู่ใน overlap จะต้อง clear
- Track Circuits ทุกตัวที่อยู่ใน Flank Protection ต้อง clear
- All points ที่เกี่ยวข้องกับ Route จะต้อง locked และ detected ใน correct position
- Exit Signal (end signal) ดวงไฟสัญญาณจะต้องติด ทำงานเป็นปกติ
- Signal อื่นๆ ที่อยู่ทิศทางตรงกันข้ามกับ Route ต้องแสดงค่าเป็น normal หรือค่า default ของ Signal นั้นๆ
- Approach locking ควบคุมการทำงานอย่างถูกต้อง

ถ้าเงื่อนไขเหล่านี้ครบถ้วนถูกต้องก็สามารถแสดงสัญญาณไฟ Proceed aspect อนุญาตให้รถไฟวิ่งเข้ามาในเส้นทางได้

นอกจาก Route Setting แล้วยังมีการทำงานในส่วนอื่นที่เป็นส่วนประกอบของการทำงานของ Route Setting ได้แก่

2.4.2 Aspect Sequence

เมื่อเงื่อนไขต่างๆ ถูกต้องครบถ้วนและ Start signal สามารถที่จะแสดงสัญญาณไฟ proceed aspect ได้ สิ่งต่อไปที่จะต้องพิจารณาคือ ควรจะแสดงสัญญาณไฟสีอะไร ระหว่างสีเขียว (clear aspect) หรือสีเหลือง (caution aspect) นั่นคือต้องพิจารณาว่า signal ที่อยู่ข้างหน้ามีสีอะไร ซึ่งก็คือสัญญาณไฟของ end signal นั่นเอง ซึ่งเป็นไปตามหลักการของ Signaling System ดังนี้

ถ้า End signal เป็นสีแดง start signal จะต้องเป็นสีเหลือง

ถ้า End signal เป็นสีเหลือง หรือสีเขียว start signal จะต้องเป็นสีเขียว

2.4.3 Route Release

แบ่งเป็น 2 ลักษณะคือ Route Release โดย dispatcher เป็นผู้ส่ง Maneuver ว่าต้องการจะยกเลิก Route เรียกว่า Route released by Route Cancel Command ส่วนอีกลักษณะคือ Route จะทำการ release หลังจากการเคลื่อนที่ผ่านของรถไฟโดยอัตโนมัติ เรียกว่า Train Operated Route (TORR) หรือ Route released by Train Passage ซึ่งเงื่อนไขของการ passage ของรถไฟต้องเป็นไปตามลำดับอย่างถูกต้อง

2.4.4 Individual Point Operation

การ switch point นอกจากจะกระทำโดย Route Setting ซึ่งต้องการให้ point อยู่ในตำแหน่งที่ route ต้องการแล้ว dispatcher สามารถส่ง Maneuver สั่งไปยัง point เพื่อทำการ switch point ได้โดยตรง เรียกว่า Individual Point Operation โดยจะสามารถ switch ได้หรือไม่ขึ้นอยู่กับเงื่อนไขว่าถูกต้องตาม Interlocking logic ที่กำหนดหรือไม่

จากที่กล่าวมานั้น เป็นหลักการทำงานโดยทั่วไปของระบบ Interlocking ซึ่งมีความสอดคล้องกับ Signaling System และสนับสนุนเพื่อให้เป็นไปตามข้อกำหนด 2 ข้อ ของ vital – functions ส่วนเงื่อนไขในรายละเอียดอื่นๆ นั้นจะขึ้นอยู่ Requirement ของแต่ละระบบ ว่าต้องการการทำงานในลักษณะใด สำหรับ Requirement ของระบบจำลองที่พัฒนาขึ้นนี้จะแสดงในบทถัดไป

บทที่ 3

การวิเคราะห์ความต้องการและออกแบบ Interlocking Logic

Interlocking Logic เป็นส่วนประมวลผลของข้อมูลที่เป็นอินพุททั้งหมด เพื่อให้ได้เอาท์พุทส่งกลับไปยังส่วนแสดงผล ดังนั้น ส่วนนี้จะเป็นส่วนที่ตรวจสอบเงื่อนไขต่างๆ ซึ่งเงื่อนไขเหล่าถูกสร้างขึ้นจาก Interlocking Logic Requirement

3.1 Interlocking Logic Requirement

Interlocking Logic Requirement ของระบบจำลองที่พัฒนาขึ้น แบ่งออกเป็นส่วนๆ ตามหน้าที่การทำงาน ดังนี้

3.1.1 Route Setting

เป็น Requirement ที่เกี่ยวข้องกับการ Setting Route ทั้งหมด ซึ่งในการพัฒนาครั้งนี้จะเน้นไปที่ Main Route ซึ่งมีเงื่อนไขและข้อกำหนดต่างๆ ดังนี้

ตารางที่ 3.1

Requirement of Route Setting

Requirement No.	Informal Description
3.1.1	The route can be locked if there is no conflicting route in the opposite direction.
3.1.2	The route can be locked once the point in route is in the required position or free for switching.
3.1.3	The route can be locked once the point which provides flank protection is available.
3.1.4	The route shall lock points in the required position.
3.1.5	If the route locking is success, all objects in the route shall be changed status to be 'Route Locked' and display white-color route indication.

3.1.2 Signal Controls

เป็น requirement ที่กำหนดในการแสดง signal ว่าจะมีเงื่อนไขอะไรบ้างที่ต้องการก่อนที่ start signal จะสามารถแสดง proceed aspect รวมทั้งเงื่อนไขในการกำหนด aspect sequence

ตารางที่ 3.2

Requirement of Signal Controls

Requirement No.	Informal Description
3.2.1	All objects in the route have already in 'Route Locked' status.
3.2.2	All points in the route have been detected in required position.
3.2.3	All points providing flank protection has been detected in required position.
3.2.4	All track circuits clear.
3.2.5	The start signal can be display proceed aspect if all above conditions are fulfilled all time. If any condition can not be fulfilled, the start signal shall be replaced to red aspect.
3.2.6	The start signal can display yellow aspect once the next signal display red aspect.
3.2.7	The start signal can display green aspect once the next signal display yellow or green aspect.
3.2.8	If the green lamp has failed or broken, the signal shall degrade to display yellow aspect.
3.2.8	If the yellow lamp has failed or broken, the signal shall degrade to display red aspect.
3.2.9	If the red signal has failed or broken, the signal show dark and the system shall force the signal in rear to display red aspect.

3.1.3 Route Releasing

เป็นข้อกำหนดในเรื่องการยกเลิก route ว่ามีลักษณะใดบ้าง และมีเงื่อนไขอย่างไร

ตารางที่ 3.3

Requirement of Route Releasing

Requirement No.	Informal Description
3.3.1	Manual Route Release is operated by sending Route Cancel Command to the start signal.
3.3.2	The route shall be automatically released after the valid train passage.
3.3.3	The valid train passage consists of the previous track has been released and the current track clear.
3.3.4	The route start to be automatically released once the first track has been occupied and then the berth track clears.

3.1.4 Individual Point Operation

เนื่องจาก Point เป็น object ที่นอกจากจะเป็นส่วนประกอบใน route แล้ว ยังสามารถที่จะส่ง command เพื่อสั่งให้ Point ทำงานได้โดยตรง ซึ่งมีเงื่อนไขต่างๆ ดังนี้

ตารางที่ 3.4

Requirement of Individual Point Operation

Requirement No.	Informal Description
3.4.1	Point can be switched by individual commands.
3.4.2	If the initial point position is left, the point shall be switch to right position.
3.4.3	If the initial point position is right, the point shall be switch to left postion.
3.4.4	If the point has already been locked by route, the command shall be rejected.
3.4.5	If the track circuit of point has been occupied, the point can not be switch and the command shall be rejected.

3.2 ออกแบบ Interlocking Logic

ขั้นตอนนี้เริ่มจากการออกแบบและกำหนดตัวแปรต่างๆ ที่จะนำมาใช้ใน Interlocking Logic โดยแบ่งประเภทของตัวแปรได้ ดังนี้

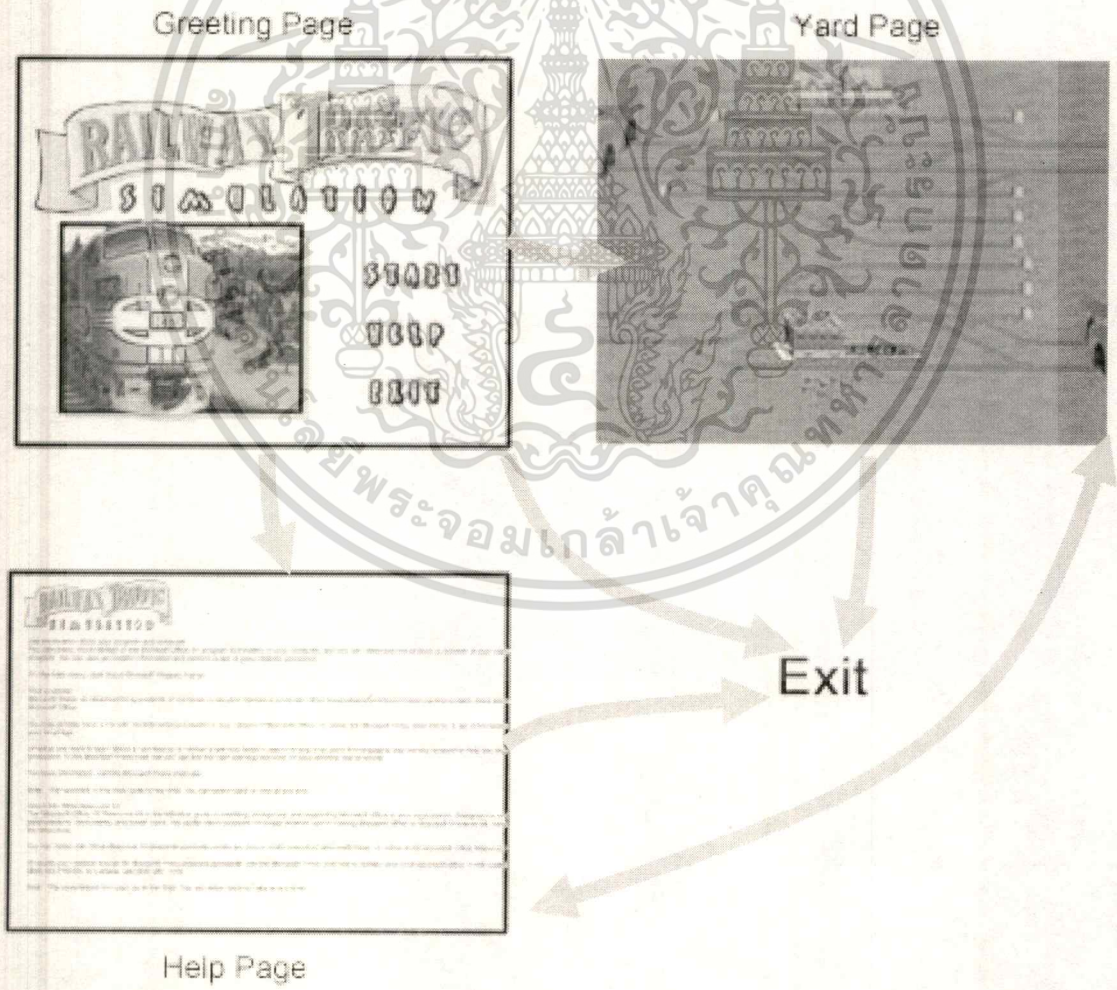
- **Global Variables** เป็นตัวแปรที่นำมาใช้สำหรับทุกๆส่วนประกอบภายในระบบจำลอง เพื่อดำเนินการที่เกี่ยวข้องกับ Route Setting, Route Releasing หรือ Signal Control ซึ่งต้องเกี่ยวข้องกับทุกอุปกรณ์ที่เป็นส่วนประกอบของเส้นทาง
- **Check Variables** เป็นค่าที่ตรวจสอบสถานะของแต่ละอุปกรณ์ในขณะนั้น ซึ่งจะนำมาใช้เป็นเงื่อนไขในการประมวลผลใน Interlocking logic ประกอบด้วยค่าต่างๆ หลายค่าแบ่งตามชนิดของอุปกรณ์
- **Command Variables** เก็บค่าคำสั่งที่ต้องการให้ดำเนินการ โดยเมื่อ Interlocking Logic รับ Command เข้ามาแล้วพิจารณาเงื่อนไขต่างๆ ว่าสามารถปฏิบัติงานตามที่ Command นั้นต้องการได้หรือไม่
- **Control Variables** เมื่อ Interlocking Logic ได้รับ Command แล้วประมวลผลได้ว่าสามารถปฏิบัติงานตามที่ Command นั้นต้องการ ก็จะส่งค่า Control ออกไปเพื่อสั่งให้อุปกรณ์ เปลี่ยนแปลงค่าเพื่อตอบสนองต่อ Command
- **Local Variables** เป็นตัวแปรใช้ในการประมวลผลเฉพาะภายใน Interlocking Logic หรืออาจใช้เก็บค่าอินพุทที่ได้รับเข้าหรือผลลัพธ์บางค่าก่อนที่ทำการส่งออกไปยังส่วนแสดงผล
- **Event Variables** เป็นตัวแปรที่ใช้เก็บค่าเหตุการณ์ต่างๆ ที่ได้รับจากผู้ใช้ โดยส่วนที่ประมวลผลและตอบสนอง Event จะเป็นส่วนของ Functional Logic จากนั้นส่งค่า Check กลับมาให้ส่วนของ Interlocking Logic เพื่อใช้ในการประมวลผลต่อไป

บทที่ 4

การพัฒนาของระบบจำลอง

4.1 โครงสร้างของระบบจำลอง (Simulation Structure)

การออกแบบระบบจำลองเป็นการนำเสนอในรูปแบบของกราฟิกที่สามารถ Interactive กับผู้ใช้ได้ตลอดเวลา นั่นคือสามารถรับอินพุทจากผู้ใช้ผ่านการใช้เมาส์และคีย์บอร์ดและแสดงเอาท์พุทที่ได้ในเวลาเดียวกัน ซึ่งเป็นลักษณะการทำงานแบบ event-driven programming นั่นคือ โปรแกรมสามารถประมวลผลตอบสนองต่อการกระทำของผู้ใช้ ในขณะเดียวกันก็สามารถดำเนินการกับส่วนอื่นๆ ได้ในเวลาเดียวกัน คล้ายคลึงกับการทำงานของเกมทางคอมพิวเตอร์ มีโครงสร้างหลัก ดังรูป



รูปที่ 4.1 แสดงความสัมพันธ์ของแต่ละหน้าของระบบจำลอง

จากรูประบบจำลองจะประกอบด้วย 3 หน้าหลักคือ

- Greeting Page: เป็นหน้าแรกเมื่อเริ่มต้นระบบจำลองซึ่งจะเชื่อมโยงไปยังหน้าอื่นๆทั้งหมด
- Yard Page: เป็นส่วนที่สำหรับดำเนินการของระบบจำลอง โดยประกอบด้วยเส้นทางรถไฟในรูปแบบต่างๆ โดยแบ่งการจัดจราจรออกเป็นสองรูปแบบหลักๆ คือ การจัดจราจรภายในสถานี และการจัดจราจรระหว่างสถานี โดยสร้างสถานีจำลองขึ้น 2 สถานี คือสถานี A และสถานี B
- Help Page: แสดงคู่มือวิธีการใช้และข้อมูลอื่นๆเกี่ยวกับระบบจำลอง

เมื่อพิจารณารูป จะเห็นว่า Greeting จะเป็นหน้าแรกเมื่อเริ่มต้นระบบ และสามารถเชื่อมโยงไปยัง Yard Page, Help Page หรือต้องการออกจากเกมก็ได้

ส่วน Yard Page เมื่อเข้ามาแล้วระหว่างที่กำลังใช้งานระบบจำลองสามารถเรียก Help Page ออกมาดูได้ และจาก Help Page ก็สามารถกลับเข้าสู่ Yard Page สลับไปมาได้ตลอดเวลา และเมื่อต้องการออกจบการทำงานสามารถออกจากระบบได้ทั้งจาก Greeting, Yard และ Help Pages

4.2 DirectX8.0 SDK

เนื่องจากการศึกษาครั้งนี้เลือกใช้ Microsoft Visual C++ ในการพัฒนา โดยเรียกใช้โครงสร้าง Interface หรือ COM (Component Object Model) จาก DirectX8.0 SKD ซึ่งทำหน้าที่เป็นตัวกลางระหว่างส่วนของ Application Program กับส่วนอุปกรณ์อินพุท และเอาท์พุท ทั้งหมด ดังนั้นโครงสร้างโดยรวมของระบบจำลอง จะใช้โครงสร้าง Programming ที่กำหนดโดย DirectX ซึ่งประกอบด้วย 2 ส่วนหลัก คือ WinMain() และ WndProc()

4.2.1 WinMain()

เป็นส่วน main program ของ Application ที่ใช้บน Windows ดังนั้นเมื่อเรียกใช้ Application ก็จะเริ่มการประมวลผลที่ส่วนนี้

การออกแบบระบบจำลอง แบ่งการทำงานภายใน WinMain() ออกเป็น 3 ส่วนคือ

1. สร้าง Window หลักของระบบจำลอง โดยมีขั้นตอน คือ
    - กำหนดคุณลักษณะของ Window Class ที่เราต้องการใช้
    - Register class ที่ได้กำหนดไว้ ซึ่งเป็นการบอก Windows ให้รู้จัก class ที่เรากำหนดค่าขึ้น
    - Create window โดยอ้างถึง class ที่เรา Register ไว้แล้ว ซึ่ง window นี้จะเป็นส่วนที่แสดงผลทางหน้าจอของระบบจำลอง
    - แสดงค่า window ที่สร้างไว้ออกทางหน้าจอ

2. Initialization ของระบบจำลอง เนื่องจากจะต้องมีการเรียกใช้ Resources ต่างๆ เพื่อใช้ในการประมวล ดังนั้นจึงต้องมีการกำหนดค่าต่างๆ อย่างเหมาะสม รายละเอียดส่วนนี้อยู่ในส่วน Initialization ในหัวข้อ 4.3
3. Message Loop ส่วนนี้จะทำการตรวจสอบว่ามี message ส่งเข้ามาหรือไม่ ถ้ามีก็จะส่งต่อไปยัง WndProc() เพื่อดำเนินการที่เกี่ยวข้องกับ message ต่อไป ในกรณีที่ไม่มี message ก็จะเริ่มเรียกใช้ส่วนประมวลผลของระบบจำลอง และแสดงภาพกราฟิก

4.2.2 WndProc()

WndProc ย่อมาจาก Window Procedure ซึ่ง function นี้ทำหน้าที่ในการจัดการ message ที่ถูกส่งเข้ามาทั้งหมดได้ตลอดเวลาที่เริ่มใช้งาน Application โดยจะเขียนโปรแกรมรองรับไว้ในส่วนนี้ว่าต้องการให้ Application ของเราตอบสนองต่อ Message อะไรบ้างและอย่างไร

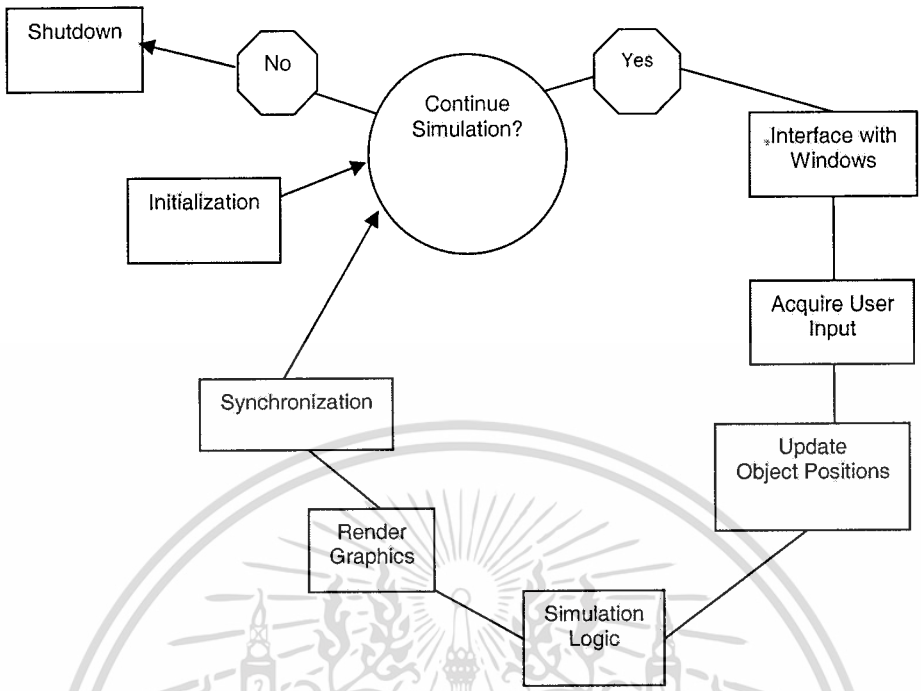
ระบบจำลองเลือกใช้การตอบสนองต่อ Message ต่างๆ ดังนี้

1. Message ในการสร้าง Window จะเรียกใช้เพียงครั้งเดียวเมื่อมีการเริ่มต้นระบบจำลองเท่านั้น
2. Message ในการวาดภาพ ในทุกรอบของการประมวลผลจะต้องมีการวาดภาพใหม่ลงบน Window ทุกครั้ง
3. Message ในการรับค่าจาก Keyboard ซึ่งส่วนนี้จะตรวจสอบว่ากด Key อะไร อยู่ภายใต้เงื่อนไขอะไรบ้าง และจะดำเนินการส่วนไหนต่อไป
4. Message ในการรับค่าการกดเมาส์ ซึ่งประกอบด้วย 2 ส่วนคือ ค่าปุ่มที่กด และค่าตำแหน่งที่มีการกดเมาส์
5. Message ของการเคลื่อนที่ของเมาส์ เป็นการตอบสนองต่อการเคลื่อนที่ของเมาส์โดยไม่ได้กดปุ่มใดๆ ของเมาส์ ดังนั้นส่วนที่จะนำไปตรวจสอบจะเป็นเฉพาะค่าตำแหน่งของเมาส์

การทำงานของ WndProc() จะเกี่ยวข้องกับการประมวลผลของระบบจำลองตลอดเวลา เพราะเป็นด่านแรกที่จะรับ input จากผู้ใช้ ซึ่งรายละเอียดว่าจะมีการตอบสนองต่อ Message อย่างไรบ้างจะกล่าวถึงในส่วนต่อๆ ไป

4.3 Simulation Loop

เนื่องจากระบบจำลองต้อง Interactive กับผู้ใช้ตลอดเวลา ดังนั้นตั้งแต่เริ่มต้นการทำงานของระบบจนจบการทำงานของระบบ จะมีการประมวลผลอย่างต่อเนื่องเป็นลักษณะวนลูป ดังนั้นการออกแบบขั้นตอนการทำงานภายในระบบจึงเป็นลักษณะดังรูป เรียกว่า Simulation Loop



รูปที่ 4.2 Simulation Loop

เมื่อทำการเชื่อมโยงข้อมูลที่ได้จากส่วนโครงสร้างหลัก ซึ่งประกอบด้วย 3 ส่วน(Greeting, Yard และ Help pages) กับส่วนของ Simulation Loop เข้าด้วยกัน สามารถทำการออกแบบแต่ละขั้นตอนของ Simulation Loop ได้ดังนี้

4.3.1 Initialization

Initialization เริ่มทำงานทันทีเมื่อมีการเริ่มการทำงานของระบบจำลอง ซึ่งทำหน้าที่ในการจัดการเกี่ยวกับ resource ต่างๆ ที่จะมีการเรียกใช้เพื่อสามารถ run โปรแกรมได้ถูกต้อง เช่นการจอง memory สำหรับ textures ต่างๆ ของรูปภาพ นอกจากนี้ยังเป็นส่วนที่ทำหน้าที่ต่อกับการ์ดแสดงผล (Display Adapter) เพื่อตั้งค่าต่างๆ ให้เหมาะสมเพื่อให้แสดงผลออกมาทางจอภาพ นอกจากนี้ยังติดต่อกับ hardware ส่วนอื่นๆ เช่น input devices ต่างๆ

ขั้นตอนที่ใช้เวลามากที่สุดคือ การโหลดข้อมูลต่างๆ จาก hard disk มายัง main memory ดังนั้นจึงควรที่จะทำในส่วนนี้เพียงครั้งเดียวให้เรียบร้อย เพื่อที่เวลาเริ่มส่วนที่เป็น simulation จริงๆ จะได้ไม่เสียเวลาเพื่อโหลดข้อมูลอีกครั้ง

ดังนั้นข้อมูลทุกอย่างของทั้ง 3 ส่วนจะถูกโหลดเข้ามาเก็บพร้อมๆ กันในส่วนนี้ โดยจะต้องสร้างโครงสร้างที่เหมาะสมรองรับ โดยสร้างเป็น function เรียกว่า SimulationInit() ซึ่งมีขั้นตอนคือ

1. สร้าง interface pointer สำหรับ IDirect3D8 โดยใช้ Direct3DCreate()
2. รับค่าจาก Display Adapter หรือการ์ดแสดงผล ที่ตั้งค่าไว้เพื่อสามารถแสดงผลออกทางหน้าจอได้อย่างถูกต้อง ซึ่งใช้ IDirect3D8::GetAdapterDisplayMode()
3. ตั้งค่า Format ต่างๆ ของ Back buffer โดยใช้ structure D3DPRESENT_PARAMETER
4. สร้าง interface pointer สำหรับ IDirect3DDevice8 โดยใช้ IDirect3D8::CreateDevice()
5. เมื่อเสร็จขั้นตอน 1-4 ซึ่งเป็นตั้งค่าของ Resources ต่างๆ ให้พร้อมใช้งานร่วมกับ DirectX ขั้นตอนต่อไป คือ เตรียมสำหรับโหลดกราฟิกต่างๆ เข้ามาเก็บไว้ โดยเริ่มจาก Clear ค่าต่างๆ ใน Back buffer โดยใช้ IDirect3DDevice8::Clear()
6. ตั้งค่า pointer สำหรับ Back buffer โดยใช้ IDirect3DDevice8::GetBackBuffer()
7. ขั้นตอน 5 – 6 เป็นการเตรียม Back buffer เพื่อใช้รองรับกราฟิกที่จะโหลดเข้ามา โดยเริ่มจากการโหลดรูปมาเก็บไว้โดยใช้ GDI (Graphic Device Interface) และกำหนด handle สำหรับ Bitmap เป็นตัวรองรับ โดยใช้ฟังก์ชั่น LoadImage() ซึ่งเป็น standard function ของ GDI ในขั้นตอนนี้เป็นการโหลดรูปกราฟิกจาก hard disk มาเก็บไว้ที่ memory
8. จากนั้นรับข้อมูลรูป มาเก็บไว้ที่ structure BITMAP โดยใช้ standard function GetObject()
9. เมื่อได้ข้อมูลมาเก็บไว้ใน structure เรียบร้อยแล้ว ก็ทำลบข้อมูลที่โหลดมาออกจาก memory ได้เพื่อเป็นการคืนหน่วยความจำ โดยใช้ DeleteObject()
10. สร้าง Surface ของ DirectX จะใช้ข้อมูลจาก structure BITMAP มากำหนดโครงสร้างของ Surface โดยสร้าง Surface จาก IDirect3DDevice8::CreateImageSurface()
11. จากนั้นจะโหลดรูปมายัง Surface ที่สร้างขึ้นโดยตรงอีกครั้ง โดยใช้ utility function D3DXLoadSurfaceFromFile() ที่ต้องทำเช่นนี้เนื่องจาก function จะเปลี่ยน format ของรูปให้เป็น format ที่ Surface ต้องการโดยอัตโนมัติ
12. ขั้นตอน 7 – 11 สำหรับการโหลดรูป 1 รูป ซึ่งต้องทำเช่นนี้สำหรับทุกๆ รูปที่มีการใช้ในระบบจำลอง

4.3.2 Check Game Status

เป็นตรวจสอบว่าผู้ใช้ต้องการออกจากระบบจำลองหรือไม่ ซึ่งจะตรวจสอบในทุกๆ รอบการทำงาน ดังนั้น ผู้ใช้สามารถออกจากระบบได้ตลอดเวลาไม่ว่ากำลังอยู่ที่ส่วนใดของระบบจำลอง (Greeting, Yard หรือ Help pages)

4.3.3 Interface with Windows

เนื่องจากการทำงานของ Windows เป็นลักษณะใช้ Resoucrce ร่วมกัน (Shared Environment) ดังนั้นจึงต้องมีการติดต่อกับ Windows ตลอดเวลาเพื่อให้สามารถใช้ระบบจำลองได้อย่างต่อเนื่อง รวมทั้งสามารถเรียกใช้การทำงานในส่วนอื่นๆ ได้ในเวลาเดียวกัน

ในระบบจำลองจะเรียกใช้ผ่าน WndProc() ดังที่ได้กล่าวไปแล้ว

4.3.4 Acquire User Input

คือการรับค่าต่างๆ จากผู้ใช้ เพื่อสามารถตอบสนองแบบ Interactive โดยผ่านคีย์บอร์ด และเมาส์ ซึ่งออกแบบการรับค่าอินพุทจากผู้ใช้ในส่วนต่างๆ ดังนี้

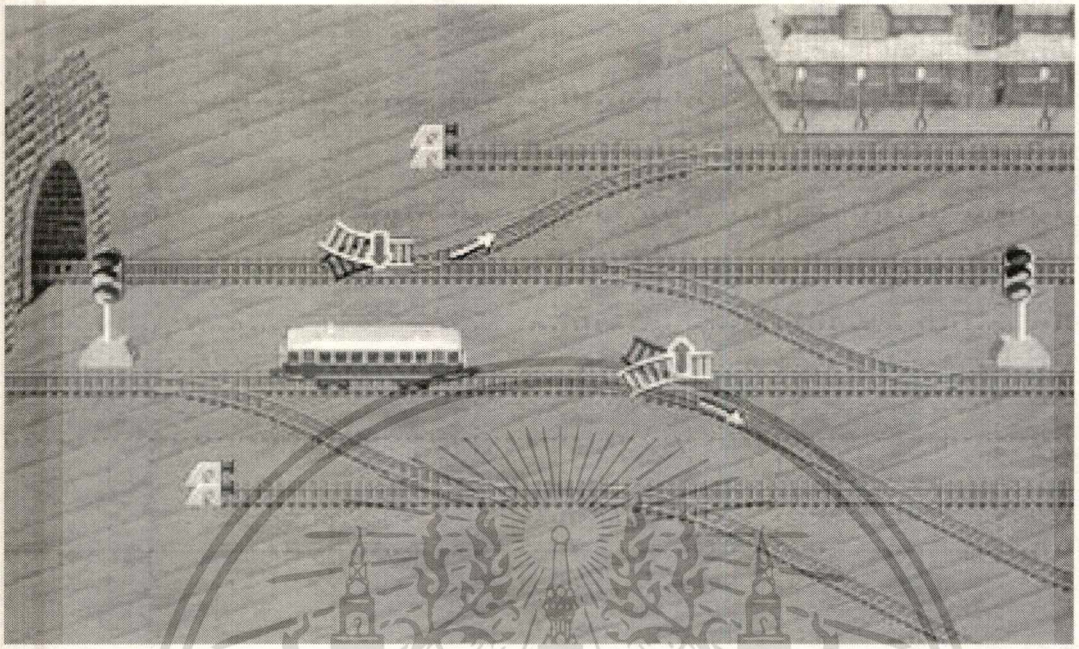
1. **Greeting Page**

รูปที่ 4.3 Greeting Page

จากรูป Greeting Page สามารถเชื่อมโยงไปยังส่วนอื่นๆ โดยมีการรับอินพุท คือ

- เมื่อคลิกเมาส์ที่ “START” จะเข้าสู่ส่วน Yard Page
- คลิกเมาส์ที่ “HELP” เข้าสู่ส่วน Help Page
- คลิกเมาส์ที่ “EXIT” เพื่อออกจากโปรแกรม
- กดปุ่ม Esc เพื่อออกจากโปรแกรม
- กดปุ่ม F1 เพื่อเข้าสู่ Help Page

2. Yard Page



รูปที่ 4.4 Yard Page

เป็นส่วนที่มีความซับซ้อนมากที่สุด แบ่งการรับค่าอินพุทเป็น 2 แบบ แบบแรกเชื่อมโยงไปยังส่วนอื่น ได้แก่

- กดปุ่ม F1 เพื่อแสดงหน้า Help Page
- กดปุ่ม Esc เพื่อออกจบภารทำงาน

แบบที่สอง คือส่วนรับค่าอินพุทเพื่อนำไปประมวลผล ได้แก่

- คลิกซ้ายที่ตัวรถไฟที่อยู่กับที่ เป็นการเลือกให้รถไฟเตรียมตัวรับค่าจุดหมายปลายทาง โดยจะปรากฏ กรอบว่าได้คลิกเลือกรถไฟ จากนั้นก็คลิกซ้ายบริเวณ Yard ที่จะให้รถไฟวิ่งไป
- คลิกซ้ายที่ตัวรถไฟขณะกำลังเคลื่อนที่ จะเป็นการหยุดรถไฟ
- เมาส์เคลื่อนที่เหนือบริเวณ Point จะปรากฏรูปลูกศรสีแดงดังที่เห็นในภาพ เพื่อแสดงว่าสามารถสั่ง Individual Point Command เพื่อเปลี่ยนทิศทางของ Point โดยการคลิกซ้ายอีกครั้ง โดยลูกศรสีขาวจะปรากฏอยู่ตลอดเวลาเพื่อบอกทิศทางปัจจุบันของ Point
- ถ้าคลิกขวาบริเวณ Point จะปรากฏเมนูให้เลือกเพื่อกำหนด Event ที่สามารถเกิดขึ้นกับ Point เช่น Point Not Detected เป็นต้น
- คลิกขวาที่ Signal จะปรากฏแถบเมนูให้เลือกเพื่อกำหนด Event และ Command ที่สามารถเกิดขึ้นกับ Signal ได้ เช่น Green Lamp Broken เป็นต้น

- คลิกขวาที่ Track Circuit ปรากฏแถบเมนูเพื่อกำหนด Event ที่สามารถเกิดขึ้นกับ Track Circuits เช่น Track Circuit Clear, Track Circuit Occupied

3. **Help Page**

Help Page จะรับอินพุทคือการกดปุ่ม F1 เพื่อกลับไปยังหน้า Yard Page และกดปุ่ม Esc เพื่อจบการทำงาน

4.3.5 Update Object Position

ขั้นตอนนี้ต่อเนื่องมาจากขั้นตอนของการรับค่าอินพุท เมื่อกำหนดค่าต่างๆ เข้ามาก็จะเกิดการเปลี่ยนแปลงของ object ซึ่งในระบบจำลอง object ที่มีการเปลี่ยนแปลงตำแหน่งคือรถไฟ คือเมื่อเราสั่งรถไฟเคลื่อนที่ ก็ต้องมีการตรวจสอบว่าตำแหน่งปัจจุบันของรถไฟอยู่ที่ตำแหน่งไหน และจุดหมายปลายทางอยู่ที่ใด เมื่อรถไฟเคลื่อนที่ไปก็จะมีการเปลี่ยนแปลงของ Track Circuit คือเมื่อรถไฟอยู่ที่เหนือ Track Circuit อันไหน Track Circuit อันนั้นก็จะแสดงค่าว่าถูก Occupied รวมทั้งถ้าเป็นการเคลื่อนที่ลักษณะ Train Passage ผ่านเข้ามาในเส้นทาง ก็จะทำให้เกิดการทำงานแบบ Route Released by Train Passage

ซึ่งการดำเนินการเหล่านี้จะถูกแบ่งออกเป็น stages ต่างๆ และกำหนด flag สำหรับแต่ละ stage เพื่อที่สามารถตรวจสอบได้ว่าขณะนี้กำลังอยู่ใน stage อะไร และต้องทำอะไรต่อไป

4.3.6 Simulation Logic

เป็นส่วนที่สำคัญที่สุด และมีความซับซ้อนมากที่สุด โดย Simulation Logic จะแบ่งออกเป็น 2 ส่วนหลัก ได้แก่

1. **Interlocking Logic**

เป็นส่วนที่กำหนดเงื่อนไขต่างๆ ตาม Interlocking Requirement ที่ออกแบบในบทที่ 3

2. **Functional Logic**

เป็นส่วนที่ทำหน้าที่เหมือนเป็น Interface ระหว่าง Interlocking Logic กับส่วนอื่นๆ ของโปรแกรม หน้าที่หลักคือ จะเป็นส่วนที่รับค่าอินพุทจากผู้ใช้ และตรวจสอบเงื่อนไขต่างๆ ของ object อื่นๆ ที่เกี่ยวข้องทั้งหมด จากนั้นก็จะเปลี่ยนแปลงค่าต่างๆ ให้อยู่ในรูปแบบที่ Interlocking Logic เข้าใจ ซึ่ง Interlocking Logic จะนำอินพุทที่ได้มาประมวลผลว่าถูกต้องตามเงื่อนไขที่กำหนดตาม Simulation Requirement หรือไม่ แล้วจะส่งผลที่ได้กลับมายังส่วนของ Functional Logic อีกครั้ง เพื่อส่งต่อไปยังส่วนแสดงผลกราฟิกทางหน้าจอ ซึ่งการทำงานบางส่วน Functional Logic ก็เป็นคนจัดการโดยไม่เกี่ยวข้องกับ Interlocking Logic เช่น เมื่อรถไฟเคลื่อนที่ผ่าน Track Circuit

และ Track Circuit ต้องแสดงผลทางหน้าจอว่าถูก Occupied ในลักษณะนี้เรียกว่า Event Handling เป็นต้น

Functional Logic สามารถแบ่งการทำงานออกเป็นส่วนๆ ได้ดังนี้

- Main Structure (Greeting, Yard, Help Pages)
- Command Handling
- Event Handling
- Communication with Interlocking Logic
- Communication with Graphic Render

4.3.7 Render Graphics

คือส่วนที่แสดงผลออกทางหน้าจอ ซึ่งทำงานโดยใช้โครงสร้างต่างๆ ของ DirectX ในการแสดงผล มีขั้นตอนต่างๆ ดังนี้

1. Clear back buffer โดยใช้ IDirect3DDevicd8::Clear() เพื่อไม่ให้มีค่าเก่าค้างอยู่
2. Validate Device เนื่องจากการทำงานบน Windows เราสามารถเรียกใช้หลายๆ Application พร้อมกัน และอาจจะให้ Application อื่นขึ้นมาเป็น Active ดังนั้นเมื่อระบบจำลอง ของเราต้องการที่จะ Active อีกครั้ง จำเป็นต้องตรวจสอบว่า Device ที่มีอยู่ยังใช้งานได้หรือไม่ ถ้ายังใช้งานได้ ก็จะทำงานในขั้นตอนถัดไป
3. สร้างค่า pointer สำหรับ back buffer โดยใช้ IDirect3DDevicd8::GetBackBuffer()
4. Clear ค่า back buffer อีกครั้งถ้าเป็น Device ที่ Active ขึ้นมาใหม่
5. Lock back buffer เพื่อที่จะสามารถใช้งาน back buffer ได้
6. Render คือส่วนที่แสดงกราฟิก ขั้นตอนคือ จาก Graphic Surface ของรูปที่เราต้องการจะแสดงซึ่งเราสร้าง Surface นี้ไว้แล้วในขั้นตอนของ Initialization จะทำการ copy ข้อมูลรูปทั้งหมดมายัง memory ของ back buffer
7. Unlock back buffer เมื่อเรา copy ข้อมูลเสร็จเรียบร้อยแล้วเพื่อคืนค่า resource ซึ่งจะทำให้ Direct3D สามารถนำ back buffer ไปดำเนินการต่อได้
8. Present ค่าจาก back buffer ไปยัง primary surface นั่นคือหน้าจอแสดงผลนั่นเอง

ส่วนที่สำคัญคือส่วน Render ซึ่งจะต้องการ Render ไว้ทั้งหมดที่สามารถเกิดขึ้นได้ในระบบจำลอง แล้วสร้าง flag กำกับเพื่อใช้ในการกำหนดเงื่อนไขว่าจะเรียกใช้ Render แต่ละแบบเพื่อตอบสนองต่ออินพุทที่ส่งเข้ามาได้อย่างถูกต้อง ซึ่งส่วนที่ตัดสินใจว่าจะแสดง Render ของรูปชุดไหนขึ้นมาแสดง คือ Functional Logic

ส่วนของ Render ที่เตรียมไว้เพื่อรอการเรียกใช้ มีดังนี้

- Background ของ Greeting, Yard, และ Help page
- การเคลื่อนที่ของรถไฟแต่ละแบบเพื่อให้สอดคล้องกับ Yard
- แสดงค่า Active เมื่อการคลิกที่ตัวรถไฟ และหายไปเมื่อคลิกอีกครั้ง หรือรถไฟเริ่มเคลื่อนที่
- การแสดงค่าสถานะของ Track Circuits เพื่อตอบสนองต่อการเคลื่อนที่รถไฟ, ผลลัพธ์จาก Setting Route Command, และ Event ที่ได้รับเป็นอินพุทจากผู้ใช้
- แสดงค่า Event Menu ของ Track Circuit เมื่อคลิกขวา
- แสดงค่า signal ไฟสีแดง, เหลือง, เขียว และเมื่อ Lamp Broken
- แสดงค่า Command และ Event Menus ของ signal เมื่อคลิกซ้าย และขวาตามลำดับ
- แสดงรูป Switch Point Command เมื่อเมาส์ชี้ไป Point
- แสดงค่า Event Menu ของ Point เมื่อคลิกขวาที่ Point
- แสดงค่าทิศทางปัจจุบันของ Point ซึ่งจะเปลี่ยนเมื่อได้รับผลลัพธ์จาก Setting Route Command หรือ Switch Point Command

4.3.8 Synchronization

เป็นส่วนที่ทำหน้าที่ปรับความเร็วของการแสดงผลให้สอดคล้องกับความเร็วของ CPU โดยการกำหนด Frame rate ที่เหมาะสม โดยกำหนดไว้ที่ 30 frames/sec

4.3.9 Shutdown

ทำหน้าที่คืนค่า Resource ต่างๆ ที่มีการเรียกใช้ก่อนที่จะจบการทำงาน

4.4 Object Design

จากหัวข้อ 4.3 Simulation Loop เป็นการกำหนดโครงสร้างการทำงานทั้งหมดของระบบจำลอง ทุกๆ Object ที่อยู่ในระบบจำลองจะต้องมีการกำหนดการทำงานและคุณสมบัติต่างๆ เพื่อให้สอดคล้องกับขั้นตอนต่างๆ ของ Simulation Loop

เพื่อให้ง่ายในการออกแบบและพัฒนาจึงใช้หลักการของ Object Oriented Programming โดยแบ่ง Objects ออกเป็นชนิดต่างๆ เรียกว่า Object Types แล้วกำหนดการทำงานและคุณสมบัติของแต่ละชนิดนำมาสร้างเป็น Class ของแต่ละ Object Type แล้วนำ Class ที่ได้มาสร้าง Objects แต่ละตัวที่ปรากฏอยู่ในระบบจำลอง

โดยแบ่งการทำงานออกเป็น Class ต่างๆ ได้ดังนี้

4.4.1 Train Animation Class

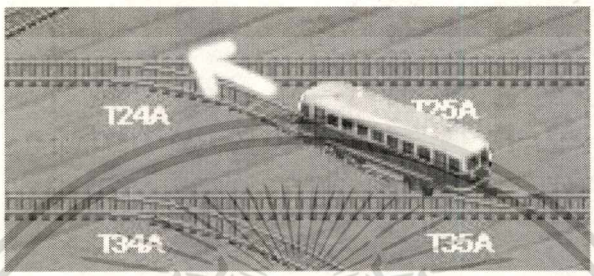
วิธีการพัฒนา เริ่มจากออกแบบลักษณะภาพเคลื่อนไหวของรถไฟ (Train Animation) ทั้งหมดที่สามารถเกิดขึ้นได้ในระบบจำลองเพื่อให้สอดคล้องกับเส้นทางการจราจร ซึ่งแบ่งภาพเคลื่อนไหวออกเป็นหลายๆ ลักษณะ แต่ละลักษณะประกอบด้วยภาพนิ่งหลายๆ ภาพ ที่มีการเปลี่ยนแปลงที่ต่อเนื่องกัน เมื่อแสดงภาพนิ่งทีละภาพด้วยความเร็วที่เหมาะสมก็จะมองเห็นเป็นภาพเคลื่อนไหวอย่างที่ต้องการ นอกจากนี้ จะต้องมีการเปลี่ยนแปลงตำแหน่งของภาพเพื่อให้เกิดการเคลื่อนที่จากตำแหน่งเดิม ดังนั้น ต้องมีการกำหนดความเร็วในการเคลื่อนที่ ว่าต้องการให้เคลื่อนที่จากตำแหน่งเดิมในทิศทางแนวตั้งกี่พิกเซล และแนวนอนกี่พิกเซล โดยความเร็วในการเคลื่อนที่และความเร็วการโหลดภาพนิ่งแต่ละภาพต้องมีความสอดคล้องกัน เพื่อให้เกิดภาพเคลื่อนไหวที่เหมาะสม

รูปที่ 4.5 ภาพนิ่งของรถไฟเพื่อนำมาสร้างภาพเคลื่อนไหว

จากรูป เป็นตัวอย่างภาพนิ่งที่เตรียมไว้เพื่อนำมาสร้างเป็นภาพเคลื่อนไหว จากนั้นสร้าง Class ของ Object Types จุดประสงค์หลักคือ เลือกภาพนิ่ง และกำหนดความเร็วในการเคลื่อนที่ได้อย่างเหมาะสม แล้วนำมาสร้างเป็นแต่ละ Object ซึ่งก็คือภาพเคลื่อนไหวในแต่ละแบบ สามารถแบ่งได้ 12 แบบ ดังนี้

- **STOP** เป็นสถานะเริ่มต้นของรถไฟซึ่งรถไฟอยู่กับที่ ไม่มีความเร็ว เตรียมรับอินพุทจากผู้ใช้ว่าต้องการให้รถไฟเคลื่อนที่ไปในทิศทางใด ภาพนิ่งที่ใช้คือภาพแนวตรงภาพเดียว
- **MOVEDOWN** รถไฟเคลื่อนที่จากบนลงล่างในแนวตรง โดยใช้ภาพนิ่งแนวตรง และกำหนดความเร็วทางด้านแนวตั้ง เท่านั้น การเคลื่อนที่ในลักษณะนี้ใช้ในกรณีที่ผู้ใช้ต้องการเปลี่ยนตำแหน่งของรถไฟจากเส้นทางรถไฟแถวหนึ่ง ไปยังอีกแถวหนึ่งซึ่งเป็นการเริ่มต้นตำแหน่งใหม่ (Reset Position) เพื่อเพิ่มความสะดวกให้แก่ผู้ใช้ในการใช้งานระบบจำลอง ทำให้สามารถเริ่มต้นการใช้งานที่ตำแหน่งใดก็ได้ของเส้นทางรถไฟทั้งหมด
- **MOVE LEFT** รถไฟเคลื่อนที่ในแนวนอนโดยไปทางด้านซ้าย โดยใช้ภาพนิ่งในแนวตรง และกำหนดความเร็วให้ทางแนวนอนทางด้านซ้าย

- **MOVE RIGHT** ทำนองเดียวกัน LEFT แต่กำหนดความเร็วให้เคลื่อนที่ไปทางด้านขวา
- **MOVE UPLEFT** คือการเคลื่อนที่ไปตามเส้นทางรถไฟ โดยเริ่มจากเส้นทางด้านล่างขึ้นไปทางด้านบน และทิศทางการเคลื่อนที่ไปทางด้านซ้าย ดังนั้นรถไฟจะมีการเลี้ยวสองครั้ง ในการเลี้ยวแต่ละครั้งจะต้องมีการเปลี่ยนแปลงภาพนิ่งเพื่อให้การเคลื่อนไหวของรถไฟให้สอดคล้องกับเส้นทาง รวมทั้งต้องกำหนดความเร็วของแต่ละภาพนิ่งให้เหมาะสม



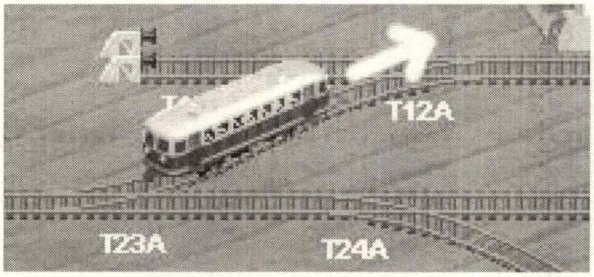
รูปที่ 4.6 การเคลื่อนที่ของรถไฟในลักษณะ UPLEFT

จากรูป เป็นตัวอย่างการเคลื่อนที่แบบ UPLEFT เพื่อต้องการเคลื่อนที่รถไฟจากเส้นทางจาก Track หมายเลข T35A ไปยัง Track หมายเลข T24A

- **MOVE DOWNRIGHT** คือการเคลื่อนที่ของรถไฟจากเส้นทางด้านบนมายังด้านล่าง โดยทิศทางไปทางด้านขวา ซึ่งใช้ภาพนิ่งชุดเดียวกับการเคลื่อนไหวแบบ UPLEFT แต่กำหนดความเร็วในการเคลื่อนที่ในทิศทางตรงกันข้าม คือไปทางด้านขวา ดังรูป

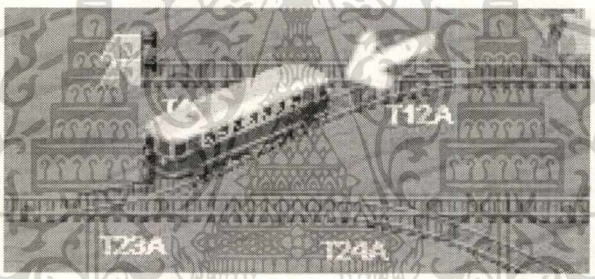
รูปที่ 4.7 การเคลื่อนที่ของรถไฟในลักษณะ DOWNRIGHT

- **UPRIGHT** คือการเคลื่อนที่ของรถไฟไปตามเส้นทาง จากเส้นทางด้านล่างไปยังด้านบน โดยทิศทางไปทางด้านขวา ดังรูป



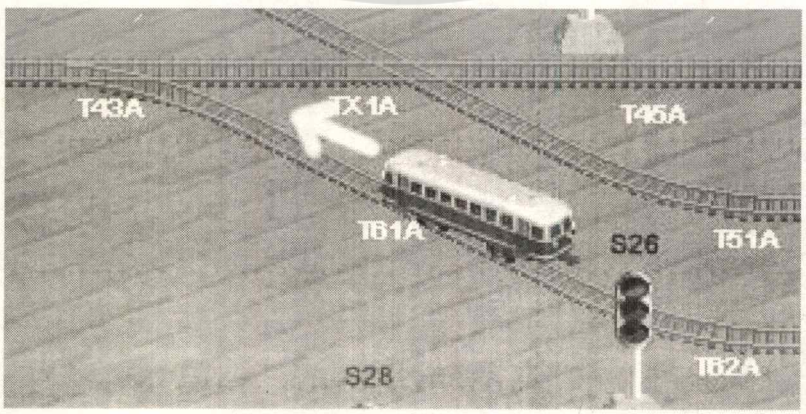
รูปที่ 4.8 การเคลื่อนที่ของรถไฟในลักษณะ UPRIGHT

- **MOVE DOWNLEFT** คือการเคลื่อนที่ของรถไฟไปตามเส้นทางจากเส้นทางด้านบนมายังเส้นทางด้านล่าง โดยทิศทางการเคลื่อนที่จะไปทางด้านซ้าย ชุดของภาพนิ่งที่ใช้เป็นชุดเดียวกับการเคลื่อนที่แบบ UPRIGHT แต่กำหนดความเร็วของการเคลื่อนที่ในทิศทางตรงกันข้าม ดังรูป



รูปที่ 4.9 การเคลื่อนที่ของรถไฟในลักษณะ DOWNLEFT

- **MOVE UPLEFT2** คล้ายคลึงกับการเคลื่อนที่แบบ UPLEFT แต่ระยะทางระหว่างเส้นทางด้านล่างกับด้านบนมีระยะเพิ่มขึ้น ดังนั้นการแสดงภาพนิ่งสำหรับในช่วงนี้จะใช้ระยะเวลามากขึ้นกว่าเดิม ดังรูป



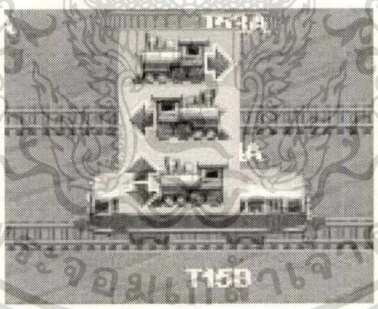
รูปที่ 4.10 การเคลื่อนที่ของรถไฟในลักษณะ UPLEFT2

- **MOVE DOWNRIGHT2** คล้ายคลึงกับการเคลื่อนที่แบบ DOWNRIGHT แต่ระยะทางระหว่างเส้นทางด้านล่างกับด้านบนมีระยะเพิ่มขึ้น
- **MOVE UPRIGHT2** คล้ายคลึงกับการเคลื่อนที่แบบ UPRIGHT แต่ระยะทางระหว่างเส้นทางด้านล่างกับด้านบนมีระยะเพิ่มขึ้น
- **MOVE DOWNLEFT2** คล้ายคลึงกับการเคลื่อนที่แบบ DOWNLEFT แต่ระยะทางระหว่างเส้นทางด้านล่างกับด้านบนมีระยะเพิ่มขึ้น

4.4.2 Train Control Class

เป็นส่วนควบคุมการเคลื่อนที่ของรถไฟ เพื่อที่ผู้ใช้สามารถกำหนดการเคลื่อนที่ได้ตามที่ต้องการ ซึ่งเป็นการจำลองว่าผู้ใช้เป็นพนักงานขับรถไฟ สามารถสั่งให้รถไฟเคลื่อนที่ หรือหยุดรถไฟที่กำลังเคลื่อนที่ได้ตลอดเวลา โดยรับอินพุทจากผู้ใช้ผ่านการคลิกเมาส์บนตัวรถไฟ ดังนั้นจึงต้องสร้างส่วนประกอบต่างๆ ของ Class เพื่อสามารถตอบสนองต่อผู้ใช้ ซึ่งแบ่งออกเป็น 2 ส่วนหลักคือ

- **ต้องการให้รถไฟเคลื่อนที่** ผู้ใช้ต้องทำการคลิกขวาที่บริเวณตัวรถไฟ จะปรากฏ Event Box ดังรูป จากนั้นใช้เมาส์คลิกซ้ายเพื่อเลือก Event ที่ต้องการอีกครั้ง



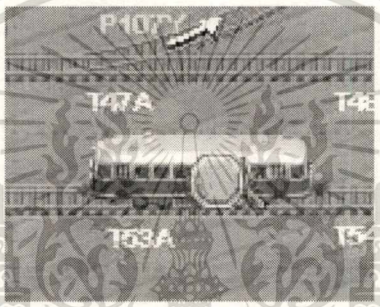
รูปที่ 4.11 แสดง Event Box เมื่อทำการคลิกขวาบริเวณตัวรถไฟ

เริ่มต้นการทำงานจาก เมื่อมีการคลิกเมาส์ จะต้องรับค่าตำแหน่งการคลิกของเมาส์ มาเปรียบเทียบกับตำแหน่งของรถไฟว่าเป็นการคลิกบริเวณรถไฟหรือไม่ โดยอินพุทที่ได้รับมีสองส่วนคือการคลิกเมาส์จากผู้ใช้ และตำแหน่งของรถไฟซึ่งได้รับจากการ Object ของ Train Animation Class ที่มีการเรียกใช้ ณ ขณะนั้น ถ้าตำแหน่งของการคลิกเมาส์และตำแหน่งรถไฟสอดคล้องกัน จะต้องทำการตอบสนองโดยแสดงภาพ Event Box

Event Box ประกอบด้วย Events 3 ชนิด ซึ่งจะปรากฏแถบสีเมื่อเมาส์เคลื่อนที่ผ่านแต่ละ Event เพื่อให้สังเกตได้ง่ายเมื่อต้องการทำการเลือกโดยการคลิกซ้าย

ค่าของ Event ที่เลือกจะถูกเก็บไว้ในตัวแปรเพื่อนำไปใช้งานต่อไป ในการกำหนดรูปแบบของ Train Animation

- **ต้องการหยุดรถไฟ** เมื่อผู้ใช้ต้องการหยุดรถไฟที่กำลังเคลื่อน สามารถทำได้โดยการคลิกซ้ายไปที่บริเวณตัวรถไฟ โดยขณะที่รถไฟกำลังเคลื่อนที่อยู่ ถ้าตำแหน่งของเมาส์ชี้ไปที่บริเวณตัวรถไฟจะปรากฏรูปภาพวงกลมสีแดง เพื่อแสดงว่าผู้ใช้สามารถหยุดรถไฟได้ตลอดเวลา ดังรูป



รูปที่ 4.12 แสดงรูปเพื่อหยุดรถไฟที่กำลังเคลื่อนที่

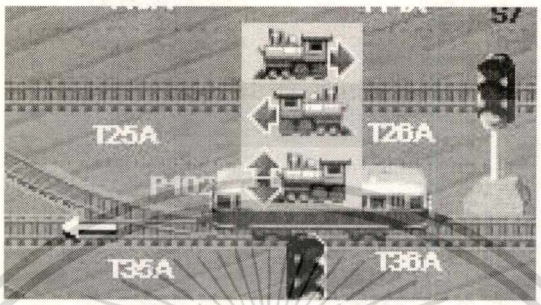
เมื่อผู้ใช้คลิกซ้าย ค่า Event ในการหยุดรถไฟจะถูกเก็บไว้ในตัวแปร เช่นเดียวกับ Events ต่างๆ ที่ได้รับจากการคลิกบน Event Box

4.4.3 ความสัมพันธ์ระหว่าง Train Animation Class และ Train Control Class

ค่า Event ต่างๆ ที่ได้จาก Train Control Class จะถูกนำมากำหนดรูปแบบของ Train Animation ที่เหมาะสมที่จะถูกแสดงทางหน้าจอเพื่อตอบสนองต่อ Event นั้นๆ โดยสิ่งที่ต้องนำมาพิจารณาร่วมด้วยคือ ตำแหน่งของรถไฟ และการกำหนดความสัมพันธ์ระหว่างตำแหน่งรถไฟกับเส้นทางรถไฟ รวมทั้งตำแหน่งของ Point (Point Direction) ว่าอยู่ในทิศทางไหน โดยทำการแบ่งเส้นทางรถไฟออกเป็นแถวๆ ในแต่ละสถานี ในแต่ละแถวเส้นทางจะถูกแบ่งออกเป็นช่วงๆ และพิจารณาว่าเส้นทางช่วงนี้ควรใช้การเคลื่อนที่แบบใด เมื่อตำแหน่งของรถไฟเคลื่อนเข้าสู่เส้นทางช่วงนี้ก็เปลี่ยนการเคลื่อนที่ให้เหมาะสมกับเส้นทาง แบ่งได้เป็น 2 ลักษณะ

- **เส้นทางในแนวตรง** ถ้าผู้ใช้กำหนด Event ให้รถไฟเคลื่อนที่ไปทางด้านซ้าย Train Animation ที่ถูกเรียกใช้คือ MOVE LEFT แต่ถ้าผู้ใช้กำหนด Event ให้รถไฟเคลื่อนที่ไปทางด้านขวา Train Animation ที่ถูกเรียกใช้คือ MOVE RIGHT

- **เส้นทางที่เป็น Point** ถ้า Event ที่ผู้ใช้กำหนดทำให้รถไฟเคลื่อนที่เข้าสู่ทางแยก หรือเป็นลักษณะของ Facing Point การเคลื่อนที่ของรถไฟในช่วงนี้จะต้องพิจารณาทิศทางของ Point ด้วย เช่น ตัวอย่างดังรูป

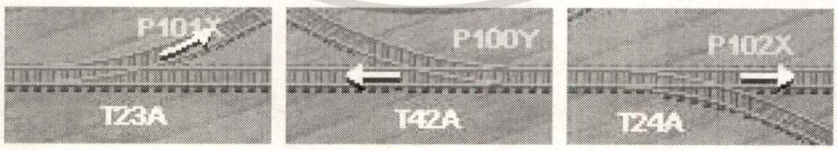


รูปที่ 4.13 แสดงการเคลื่อนที่ของรถไฟเมื่อเข้าสู่ทางแยก

จากรูป เมื่อผู้ใช้กำหนด Event ให้รถไฟเคลื่อนไปทางด้านซ้ายเข้าสู่ Track หมายเลข T35A ซึ่งเป็นทางแยก โดยทิศทางของ Point อยู่ทางซ้ายซึ่งเป็นทางตรง ดังนั้น Train Animation ที่จะถูกเรียกใช้คือ MOVE LEFT แต่ถ้าทิศทางของ Point อยู่ทางด้านขวา Train Animation ที่จะถูกเรียกใช้คือ MOVE UPLEFT เพื่อรถไฟสามารถเคลื่อนไปยังเส้นทางด้านบน เป็นต้น

4.4.4 Point Class

Point เป็นอุปกรณ์สำหรับเลือกเส้นทางรถไฟ ซึ่งติดตั้งบริเวณที่เป็นทางแยก การออกแบบเริ่มจากการกำหนดรูปแบบของ Point ที่จะต้องนำมาใช้ภายใน Yard แล้วสร้างภาพกราฟิกสำหรับทุกรูปแบบ ตัวอย่าง Point ในรูปแบบต่างๆ ดังรูป



A **B** **C**

รูปที่ 4.14 ตัวอย่างรูปแบบต่างๆ ของ Point

Point แต่ละรูปแบบจะมีหลักการทำงานเหมือนกันคือ ทำหน้าที่เปลี่ยนเส้นทางของรถไฟให้ไปทางซ้ายหรือขวาโดยดูในทิศทางที่เป็นลักษณะ Facing Point คือมองจากทางร่วมเข้าไปหาทางแยก จากรูปที่ 4.14 ทิศทางของ Point แสดงในรูปของลูกศรสีขาว ซึ่งทั้งหมดแสดงว่า Point

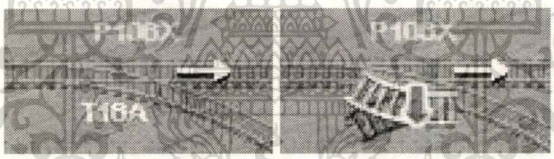
อยู่ทางซ้าย ดังนั้นต้องมีการตรวจสอบตำแหน่งของ Point ตลอดเวลาเพื่อสามารถแสดงภาพทิศทางของ Point ได้อย่างถูกต้อง

การโต้ตอบระหว่าง Point กับผู้ใช้ระบบ แบ่งออกได้เป็น 2 ลักษณะ Individual Point Command และการกำหนด Event

- **Individual Point Command** คือการส่งคำสั่ง หรือ Command เป็นการส่งคำสั่งเพื่อให้ Point ทำงานตามหลักการของ Interlocking System

คำสั่งที่นำมาใช้ในระบบจำลอง ได้แก่ Point Switch Command โดยทำงานในลักษณะ Toggle ซึ่งหมายความว่าถ้าส่งคำสั่งไปยัง Point ที่มีทิศทางอยู่ทางด้านซ้าย เมื่อได้รับคำสั่ง Point จะเปลี่ยนทิศทางมาทางด้านขวา ในทำนองเดียวกัน ถ้า Point มีทิศทางอยู่ทางด้านขวา ถ้าได้รับคำสั่งจะเปลี่ยนทิศทางมาทางด้านซ้าย

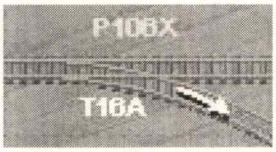
ผู้ใช้สามารถส่งคำสั่งเพื่อเปลี่ยนทิศทางของ Point ได้ตลอดเวลา โดยเมื่อตำแหน่งของเมาส์ชี้อยู่ภายในบริเวณของ Point จะปรากฏรูปภาพที่แสดงว่าสามารถ คลิกซ้ายเพื่อส่งคำสั่งในการเปลี่ยนทิศทางของ Point นั้นๆ ได้ ดังรูป



A B

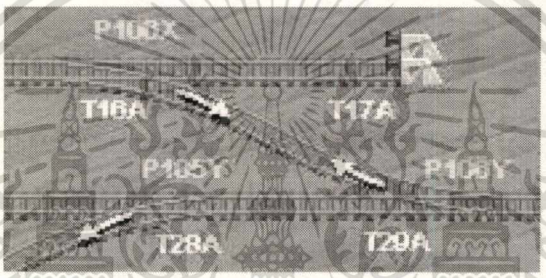
รูปที่ 4.15 แสดงรูปเพื่อบอกผู้ใช้ว่าสามารถส่งคำสั่งเปลี่ยนทิศทาง Point

ในรูป A เป็นสภาพปกติของ Point โดยตัวแหน่งอยู่ทางด้านซ้าย เมื่อเคลื่อนเมาส์มาชี้บริเวณ Point จะปรากฏรูป ดังรูป B เพื่อเป็นการบอกผู้ใช้ว่าสามารถส่งคำสั่งเพื่อเปลี่ยนทิศทางได้ โดยการคลิกเมาส์ทางซ้าย เมื่อระบบได้รับคำสั่งจากผู้ใช้ จะทำการตอบสนองโดยเปลี่ยนทิศทาง Point ดังรูป



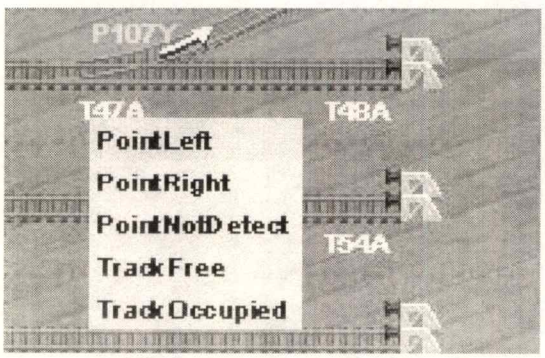
รูปที่ 4.16 Point เปลี่ยนทิศทางตอบสนองต่อคำสั่ง

ตามหลักการของ Interlocking System อุปกรณ์ Point สามารถกำหนดเป็นลักษณะของ Single Point คือทำงานเป็นอิสระไม่เกี่ยวข้องกับ Point ตัวอื่นๆ หรือกำหนดเป็นลักษณะเป็นชุด ซึ่ง ถ้า 1 ชุดประกอบด้วย Point 2 ตัว เรียกว่า Couple Point หรือถ้าประกอบด้วย Point 3 ตัว เรียกว่า Triple Point ซึ่ง Points ที่อยู่ภายในชุดเดียวกันจะต้องมีทิศทางเดียวกันเสมอเพื่อเป็นการ Interlock เพื่อให้เกิดความปลอดภัย ดังนั้น ถ้า Point ตัวใดตัวหนึ่งที่อยู่ภายในชุดเดียวกันได้รับคำสั่งให้ทำการเปลี่ยนทิศทาง Points ตัวอื่นๆ ที่อยู่ภายในชุดเดียวกันจะต้องเปลี่ยนทิศทางไปในทิศทางเดียวกัน ตัวอย่าง จากรูปที่ 4.17 Point หมายเลข P106X เป็น Couple Point ดังนั้นเมื่อเปลี่ยนทิศทางตามคำสั่งที่ได้รับ ทำให้ Point ที่อยู่ภายในชุดเดียวกันคือ P106Y เปลี่ยนทิศทางเช่นเดียวกัน ดังรูป



รูปที่ 4.17 Couple Point หมายเลข P106X และ P106Y

- **Event** เป็นการกำหนดเหตุการณ์ที่สามารถเกิดขึ้นได้บริเวณ Yard เช่น Point มีสิ่งกีดขวางทำให้ไม่สามารถตรวจสอบตำแหน่งได้, หรือมีการเปลี่ยนทิศทางของ Point จาก Yard โดยตรง (Hand Crank) โดยไม่ผ่านการส่งคำสั่งจากระบบ Interlocking หรือเพื่อการกำหนด Position เริ่มต้นของ Point (Reset Position) ก่อนเริ่มมีการทดลองการจัดจราจร เป็นต้น ผู้ใช้สามารถกำหนด Event โดยการคลิกขวาที่บริเวณ Point ระบบจะตอบสนองโดยการแสดง Event Box ของอุปกรณ์ Point ขึ้นมา ดังรูป



รูปที่ 4.18 แสดง Event Box เมื่อคลิกขวาบริเวณ Point

จากรูปที่ 4.18 แสดง Event ต่างๆ ที่สามารถเกิดขึ้นได้กับ Point โดยเมื่อเลื่อนตำแหน่งเมาส์ไปตาม Event Box จะปรากฏแถบสี เพื่อผู้ใช้สามารถคลิกซ้ายเพื่อเลือกกำหนด Event ที่ต้องการได้ ซึ่งจะเป็นการกำหนดค่าให้กับตัวแปร Event เพื่อใช้ในการตอบสนอง เช่นเมื่อเลือก PointLeft เป็นการบังคับทิศทางของ Point ให้อยู่ทางซ้ายเสมอ ไม่ว่าเดิม Point จะอยู่ในสถานะใด

ในกรณีของ Track Free และ Track Occupied เป็น Event ที่กำหนดให้กับ Track ที่ติดตั้งอยู่กับ Point ซึ่งจะกล่าวอีกครั้งในหัวข้อ Track Class

ในกรณีที่เลือก Point Not Detect เป็นการกำหนดให้ Point ไม่สามารถตรวจสอบได้ว่าขณะนั้นอยู่ในทิศทางใด ซึ่งสามารถเกิดขึ้นได้ในเหตุการณ์จริง เช่น มีสิ่งกีดขวางทำให้ไม่สามารถเปลี่ยนทิศทาง Point ไปในทิศทางที่ต้องการ หรือเป็นช่วงเวลาที่ Point กำลังเคลื่อนที่อยู่ เป็นต้น ในกรณีนี้ระบบจำลอง จะตอบสนองต่อ Event นี้ คือ ไม่แสดงลูกศรสีขาวที่บ่งบอกทิศทางของ Point นั้นๆ ทำให้ผู้ใช้รู้ว่าขณะนี้ Point อยู่ในสถานะที่ไม่สามารถบอกได้ว่าอยู่ในทิศทางใด ดังรูป

รูปที่ 4.19 แสดงการตอบสนองเมื่อกำหนด Event เป็น Point Not Detect

ในกรณีของ Couple Point จะตอบสนองต่อ Event เหมือนกันเสมอ ยกเว้น Event ที่กำหนดให้กับ Track ซึ่งถือว่าไม่ได้เป็น Event ของอุปกรณ์ Point โดยตรง แต่กำหนดให้กับ Track ที่ติดตั้งอยู่กับ Point ดังนั้น ถ้าเรากำหนด Track Occupied ให้กับ Point หมายเลข P107Y ผลตอบสนองจะปรากฏที่ P107Y เพียงตัวเดียว ส่วน Point หมายเลข P107X ซึ่งเป็น Couple Point กัน จะไม่เกิดการเปลี่ยนแปลงใดๆ

เมื่อกำหนดการทำงานและคุณสมบัติต่างๆ ของ Point เรียบร้อยแล้ว ก็จะทำการสร้าง Object Points แต่ละหมายเลขจาก Point Class ที่สร้างขึ้น โดยกำหนดรูปแบบ Point ที่ต้องการสำหรับแต่ละหมายเลข เพื่อให้สามารถแสดงค่าสถานะต่างๆ ได้อย่างถูกต้อง

4.4.5 Track Class

Track คือส่วนของรางรถไฟในแต่ละช่วงซึ่งจะมีการติดตั้งอุปกรณ์เพื่อตรวจสอบว่ามีรถไฟวิ่งเข้ามาในรางรถไฟช่วงนั้นหรือไม่ โดยเรียกอุปกรณ์นั้นว่า Track Circuit

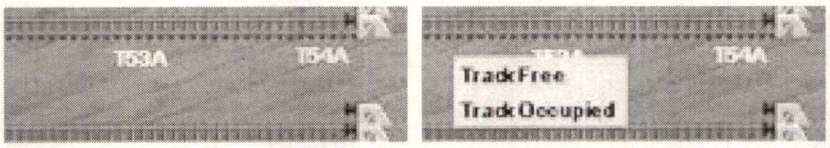
Track Circuit เป็นอุปกรณ์ที่ไม่มีการส่งคำสั่ง เป็นเพียงอุปกรณ์ที่ทำหน้าที่ในการตรวจสอบสถานะของรางรถไฟ และส่งค่าที่ได้ให้แก่ Interlocking System เพื่อใช้ในการจัดจราจรให้เป็นไปตามหลักของ Interlocking Logic ดังนั้นในระบบจำลองจะมีการทำงานเฉพาะส่วนที่กำหนด Event และแสดงผลตอบสนองต่อ Event นั้นๆ

ขั้นตอนการออกแบบเริ่มจากการสร้างรูปแบบของรางรถไฟแบบต่างๆ แล้วนำมาจัดวางให้เป็น Yard ดังรูป ซึ่งเป็นเส้นทางที่สร้างขึ้นสำหรับ Station A

รูปที่ 4.20 Yard ภายใน Station A

Track จะถูกแบ่งออกเป็นช่วงๆ และกำหนดหมายเลข Track Circuits สำหรับช่วงนั้นๆ Track Circuits ประกอบด้วย 2 สถานะ คือ Track Free และ Occupied ดังนั้นสถานะเหล่านี้จะถูกนำมากำหนดเป็นค่าของ Event เพื่อใช้ในการทำงานระบบจำลอง เช่น ต้องการดูการทำงานว่าจะสามารถสร้างเส้นทาง (Route Setting) ได้หรือไม่ ถ้าภายในเส้นทางที่ต้องการมี Track ที่ถูก Occupied อยู่ เป็นต้น

การกำหนด Event ทำได้โดยคลิกเมาส์ทางด้านขวาบริเวณ Track Circuits ซึ่งระบบจำลองจะตอบสนองโดยแสดง Event Box ดังรูป



A B

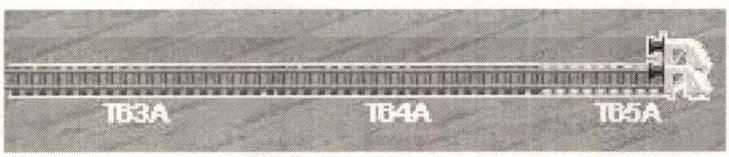
รูปที่ 4.21 การแสดง Event Box เมื่อคลิกขวาบริเวณ Track หมายเลข T53A

สถานะปกติของ Track Circuits คือ Track Free ดังนั้นถ้าผู้ใช้กำหนด Event ให้เป็น Track Occupied ภาพของ Track Circuits ในช่วงนั้นจะมีสีแดงเพื่อแสดงว่ามีสถานะเป็น Track Occupied ดังรูป

รูปที่ 4.22 Track หมายเลข T53A มีสถานะเป็น Track Occupied

การแสดงค่าของ Track Occupied นอกจากตอบสนองต่อ Event ที่กำหนดโดยผู้ใช้แล้ว จะตอบสนองต่อการเคลื่อนที่ของรถไฟด้วยเช่นกัน ดังรูปที่ 4.22 จะเห็นว่า Track Circuit ที่อยู่ในบริเวณของรถไฟจะแสดงค่าเป็นสีแดงเพื่อแสดงว่ากำลังถูก Occupied

นอกจากนี้ การแสดงค่าของ Track Circuits จะตอบสนองต่อการสร้างเส้นทาง (Route Setting) คือถ้า Track Circuit เป็นส่วนหนึ่งในเส้นทางที่สามารถสร้างขึ้นได้สำเร็จ (Route Locking) จะตอบสนองโดยแสดงค่าเป็นสีขาว ดังรูป



รูปที่ 4.23 แสดงสีขาวเมื่อ Track Circuits เป็นส่วนหนึ่งของ Route

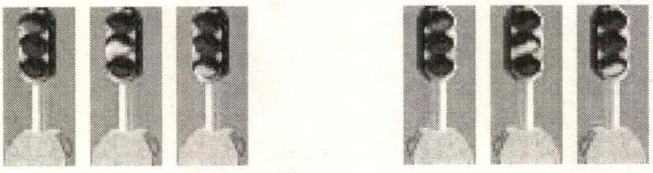
คุณสมบัติต่างๆ ของ Track Circuits จะรวมถึง Track Circuit ที่ติดตั้งอยู่กับ Point ด้วยเช่นกัน ดังนั้น ถ้า Point ถูก Occupied ก็จะแสดงค่าเป็นสีแดง หรือเป็นส่วนหนึ่งใน Route ก็จะแสดงค่าสีขาวเช่นเดียวกัน

4.4.6 Signal Class

Signal เป็นอุปกรณ์ที่แสดงไฟสัญญาณเพื่อบอกว่าจะอนุญาตให้รถไฟผ่านเข้ามาในเส้นทางได้หรือไม่

ประเภทของ Signal ที่ใช้ในระบบจำลองเป็นชนิด Main Signal ประกอบด้วยไฟสัญญาณ 3 ดวง (3-Aspect Signal) คือ สีแดง สีเขียว และสีเหลือง ซึ่งสีแดงเป็นสัญญาณที่ไม่อนุญาตให้รถไฟผ่านเข้ามาในเส้นทาง หรือเรียกว่า Stop Aspect โดยปกติ Signal จะตั้งค่าไว้ที่สีแดงเสมอเพื่อความปลอดภัย ส่วนไฟสัญญาณสีเหลืองและสีเขียวเป็นสัญญาณไฟที่อนุญาตให้รถไฟวิ่งผ่านเข้ามาในเส้นทาง หรือเรียกว่า Proceed Aspect สิ่งที่แตกต่างกันระหว่างสีเหลืองและสีเขียวคือ ความเร็วของรถไฟที่ใช้ นั่นคือ ถ้าสัญญาณสีเหลืองหมายความว่าสัญญาณไฟที่อยู่ด้านหน้า (Next Signal) เป็น Stop Aspect จึงต้องแสดงสีเหลืองเพื่อเป็นการเตือนพนักงานขับรถไฟให้ลดความเร็วรถไฟลงเพื่อทำเตรียมการหยุดเมื่อรถไฟวิ่งไปถึงสัญญาณไฟที่อยู่ด้านหน้า ที่ต้องทำเช่นนี้เนื่องจากรถไฟใช้ระยะทางยาวในการหยุดรถไฟ ทำให้ไม่สามารถหยุดรถไฟได้อย่างกะทันหัน จึงต้องมีการเตือนล่วงหน้า ในกรณีของสัญญาณไฟสีเขียวเป็นการอนุญาตให้รถไฟวิ่งผ่านเข้าโดยใช้ความเร็วปกติ เนื่องจากสัญญาณไฟที่อยู่ด้านหน้าแสดงค่า Proceed Aspect จึงไม่มีความจำเป็นที่จะต้องลดความเร็วลง

การแสดงภาพ Signal ประกอบด้วย 6 ภาพหลัก เพื่อแสดงไฟสัญญาณสีต่างๆ โดยแบ่งเป็น 2 ชุด เพื่อใช้สำหรับการสร้างเส้นทาง (Route Setting) ที่มีทิศทางทั้งสองด้าน คือ Route ที่อนุญาตรถไฟวิ่งจากทางด้านซ้ายไปทางด้านขวา และจากทางขวาไปทางด้านซ้าย ดังรูป

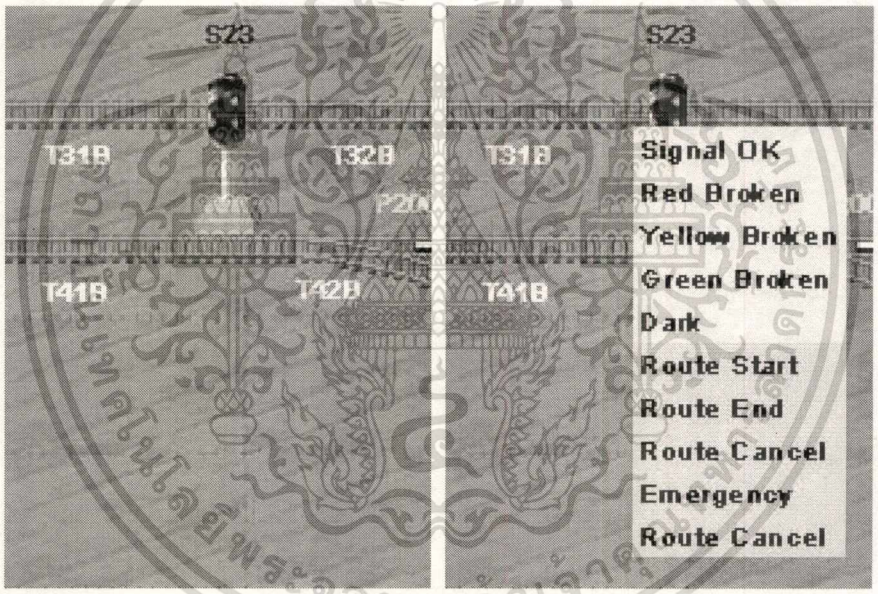


Left-to- Right Route **Right-to-Left Route**

รูปที่ 4.24 Signal แบบต่างๆ

Signal เป็นอุปกรณ์ที่มีการทำงานความซับซ้อนในการตรวจสอบเงื่อนไขต่างๆ และเกี่ยวข้องกับส่วนอื่นๆ ทั้งหมด ทั้งในส่วนของอุปกรณ์ชนิดอื่น และส่วนที่เกี่ยวข้อง Route ซึ่งสามารถแบ่งการทำงานออกเป็นส่วนต่างๆ ได้ ดังนี้

Command และ Event เป็นส่วนที่ได้รับการกำหนดค่าจากผู้ใช้ โดยการคลิกเมาส์ทางด้านขวาบริเวณ Signal ระบบจำลองจะทำการตอบสนองโดยการแสดง Event Box และ Command Box ขึ้นมาพร้อมกัน แยกความแตกต่างด้วยสี คือ Event Box มีสีเหลือง ส่วน Command Box มีสีส้ม เพื่อความชัดเจน ผู้ใช้สามารถเลือกกำหนด Event และ Command ที่ต้องการโดยการคลิกเมาส์ทางด้านซ้ายเพื่อเลือกค่า ค่าของ Event และ Command จะถูกเก็บไว้ในตัวแปร เพื่อใช้ในการดำเนินในการตอบสนองต่อ Event หรือ Command นั้นๆ ต่อไป



รูปที่ 4.25 แสดง Event Box และ Command Box เมื่อคลิกเมาส์ขวาที่ Signal

4.4.6.1 Command

Command ที่ Signal จะได้รับแบ่งออกได้เป็น Command ที่เกี่ยวข้องกับ Route Setting ได้แก่ Route Start Command กับ Route End Command และ Command ที่เกี่ยวข้องกับ Route Releasing ได้แก่ Route Cancel Command กับ Emergency Route Cancel Command

- **Route Start Command** เป็น Command เพื่อใช้ในการสร้างเส้นทาง (Route Setting) โดยต้องการให้ Signal ที่ได้รับ Command นี้ ทำหน้าที่เป็น Start Signal ซึ่งจะแสดงไฟสัญญาณอนุญาตให้รถไฟเข้ามาในเส้นทางได้

เมื่อ Signal ได้รับค่า Command จะไม่สามารถส่ง Command ไปคิดเงื่อนไขตามหลักการของ Interlocking Logic ได้โดยทันที จะต้องมีการตรวจสอบความถูกต้องของ Command ที่ได้รับก่อน นั่นคือ เมื่อได้รับ Command จะคงสถานะที่เรียกว่า Route Reservedซึ่งแสดงทางกราฟิกในรูปของสัญญาณไฟสีแดงกะพริบ (Blink Red Signal)

- **Route End Command** Route Start Command จะส่งไปยังส่วนของ Interlocking Logic ก็ต่อเมื่อมีการส่ง Route End Command มายังอุปกรณ์ที่เป็นถูกกำหนดเป็นอุปกรณ์ปลายทาง เรียกว่า End Object ซึ่งถ้าเป็น End Object เป็น Signal จะถูกเรียกว่า End Signal

เมื่อ End Signal ได้รับ Route End Command จะทำการตรวจสอบเงื่อนไขว่า End Signal และ Start Signal มีความสอดคล้องหรือไม่ คือมีความเป็นไปได้หรือไม่ที่จะทำการสร้างเส้นทางระหว่างกัน โดยระหว่างที่มีการตรวจสอบเงื่อนไขต่างๆ จะคงสถานะ Route Reservedซึ่งแสดงสัญญาณไฟสีแดงกะพริบเช่นเดียวกับ Start Signal

ถ้าเงื่อนไขถูกต้องจะทำการส่ง Commands ไปยังส่วน Interlocking Logic เพื่อดำเนินการในส่วนอื่นๆ ต่อไป แต่ถ้าเงื่อนไขไม่ถูกต้อง Command จะถูก Rejected และค่าสถานะต่างๆ ของ Start Signal และ End Signal จะถูกตั้งค่าสู่สถานะเริ่มต้น (Reset)

การทำงานในส่วนตรวจสอบ Command จะกระทำโดย Functional Logic ดังที่กล่าวมาแล้วเพื่อป้องกันไม่ให้เกิด Non-valid Command

- **Route Cancel Command** ใช้สำหรับยกเลิกเส้นทาง โดยส่ง Command ไปที่ Start Signal ซึ่ง Functional จะทำการตรวจสอบว่ามีการสร้างเส้นทางโดยใช้ Start Signal ตัวนั้นหรือไม่

ถ้าเส้นทางแสดงว่าเป็น Valid Command สามารถส่งต่อไปยัง Interlocking Logic ได้ แต่ถ้าไม่มีการสร้างเส้นทางไว้ หรือว่าเส้นทางเริ่มมีการ Released โดย Train Passage แล้ว แสดงว่าเป็น Non-valid Command และ Command จะถูก Rejected

ดังนั้น Route Cancel เป็น Command ที่ใช้สำหรับการยกเลิกเส้นทางที่ยังสมบูรณ์อยู่เท่านั้น ไม่สามารถยกเลิกเส้นทางที่มีรถไฟเริ่มวิ่งเข้าในเส้นทางแล้ว

- **Emergency Route Cancel Command** ทำงานคล้ายคลึงกับ Route Cancel Command แต่ใช้สำหรับการยกเลิกเส้นทางที่มีปัญหา ไม่สามารถใช้ Route Cancel Command ได้ โดยส่ง Command ไปที่ End Signal ซึ่งสามารถยกเลิกเส้นทางได้ทั้งหมดที่ Route Cancel Command ไม่สามารถทำได้

จุดประสงค์ในการแยกเป็น 2 Commands เพื่อความปลอดภัย เพราะในกรณีของ Route Cancel Command ถ้ายังมีรถไฟอยู่ในเส้นทาง แล้วอนุญาตให้ยกเลิกเส้นทางได้จะทำให้เกิดอันตรายจากการสร้างเส้นทางอื่นเข้ามาในทิศทางตรงข้าม หรือมีการเปลี่ยนทิศทางของ Point เป็นต้น

ดังนั้น Emergency Route Cancel เหมือนเป็นการละเมิดกฎความปลอดภัย แต่จำเป็นต้องมี เนื่องอาจเกิดกรณีที่ไม่มีรถไฟอยู่ในเส้นทางจริงๆ แต่อุปกรณ์ Track Circuits มีความผิดพลาด (Track Circuit Failed) ซึ่งทำให้ส่งค่า Track Occupied ทั้งๆ ที่ไม่มีรถไฟอยู่ในเส้นทาง ถ้าไม่สามารถยกเลิกเส้นทางเดิม ก็ไม่สามารถสร้างเส้นทางใหม่ที่อาจมีการใช้อุปกรณ์ภายในเส้นทางร่วมกับเส้นทางเดิม จึงต้องมีการสร้าง Emergency Route Cancel Command ขึ้นมารองรับในกรณีนี้

โดยปกติในการทำงานจริง Emergency Route Cancel ถือว่าเป็น Critical Command คือไม่สามารถสั่งงานได้ตามปกติ ผู้ที่สามารถใช้ Command นี้จะต้องได้รับ Authority เท่านั้น และมีการบันทึกหรือการเก็บค่าต่างๆ เพื่อสามารถตรวจสอบและเก็บไว้เป็นหลักฐาน เพื่อป้องกันความผิดพลาด

ในระบบจำลอง สร้างขึ้นเพื่อผู้ใช้สามารถศึกษาได้ว่าเงื่อนไขใดบ้างที่สามารถใช้ Route Cancel Command และเงื่อนไขแบบใดต้องใช้ Emergency Route Cancel Command

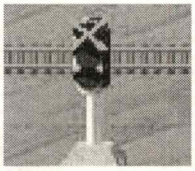
4.4.6.2 Event

Event เป็นเหตุการณ์ที่สามารถเกิดขึ้นได้กับอุปกรณ์ โดยตรง โดยไม่เกี่ยวข้องกับการทำงานของ Interlocking System แต่ Interlocking จะต้องเตรียมเงื่อนไขต่างๆ เพื่อสามารถจัดการกับเหตุการณ์เหล่านี้ เพื่อให้เกิดความปลอดภัย ดังนั้นในการพัฒนาระบบจำลองผู้ใช้สามารถกำหนดได้ว่าขณะนี้ต้องการให้เกิดเหตุการณ์ใดบ้างกับอุปกรณ์ต่างๆ ภายใน Yard และระบบจะมีการตอบสนองต่อเหตุการณ์เหล่านั้นอย่างไร

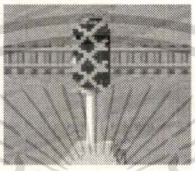
Event สำหรับ Signal คือไฟสัญญาณแตก หรือทำงานไม่ปกติเรียกว่า Lamp Broken หรือ Lamp Failed ทำให้ Signal ไม่สามารถแสดงสัญญาณตามที่ Interlocking Logic ต้องการได้ ในกรณีนี้ Interlocking Logic ยังคงส่งค่าสัญญาณไฟที่ต้องการค่าเดิม แต่ส่วนของ Functional Logic ที่ทำหน้าที่จัดการเกี่ยวกับ Event ของ Signal จะทำการเปลี่ยนแปลงค่าการแสดงสัญญาณไฟให้เป็นสัญญาณไฟสีอื่นที่ปลอดภัยมากกว่า โดยเรียกการทำงานแบบนี้ว่า Signal Degradation แล้วส่งค่าสถานะของสัญญาณไฟที่ได้รับการ Degradation แล้ว กลับไปยังส่วนของ Interlocking Logic ทำให้ทราบว่าเกิดเหตุการณ์ Lamp Broken จากการเปรียบเทียบระหว่างค่าสัญญาณไฟที่ Interlocking Logic ต้องการ กับค่าสถานะของสัญญาณไฟที่ส่งกลับมาไม่ตรงกัน หลังจากนั้น Interlocking Logic จะนำค่าสถานะของสัญญาณไฟมาเป็นอินพุทเพื่อตรวจสอบเงื่อนไขในส่วนอื่นๆ ต่อไป

จากรูปที่ 5.25 Event Box มีสีเหลือง ประกอบด้วย Events ที่สามารถกำหนดได้ในระบบจำลอง ได้แก่ Signal OK, Red Broken, Yellow Broken, Green Broken. และ Dark ซึ่งเมื่อผู้ใช้เลือกก็จะส่งค่าไปเก็บไว้ในตัวแปร โดย Functional Logic จะแสดงค่าเพื่อตอบสนองต่อ Events ต่างๆ ดังนี้

- **เมื่อไฟสัญญาณใดๆ แตก** ตอบสนองโดยแสดงค่ากากบาทสีแดงบริเวณดวงไฟสัญญาณที่แตก ดังรูป



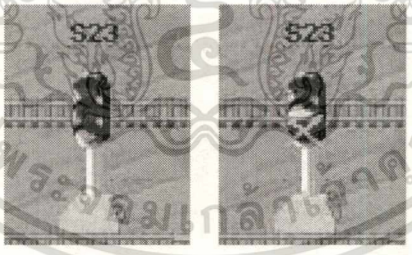
รูปที่ 4.26 Red Broken



รูปที่ 4.27 Red Broken, Yellow Broken, Green Broken หรือ Dark

- **ไฟสัญญาณสีเขียวแตก หรือ Green Broken** สามารถเกิดได้หลายกรณี ได้แก่

ในกรณีที่ Signalกำลังแสดงไฟสัญญาณสีเขียว แล้วสัญญาณไฟสีเขียวเกิดแตก จะตอบสนองโดยแสดงกากบาทที่ตำแหน่งสัญญาณไฟสีเขียว และ Signal Degradation ไปเป็นสัญญาณไฟสีเหลือง ดังรูป



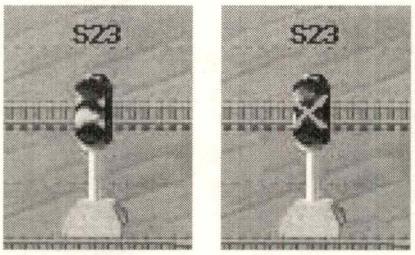
Before **Green Broken**

รูปที่ 4.28 การตอบสนองเมื่อ Green Broken ขณะแสดงสัญญาณไฟสีเขียว

ในกรณีที่ Signal กำลังแสดงสัญญาณไฟสีอื่น เมื่อสัญญาณไฟเขียวแตกจะไม่มีผลให้สัญญาณไฟเปลี่ยนแปลง แสดงเฉพาะกากบาทบนตำแหน่งของไฟสีเขียวเท่านั้น

- **ไฟสัญญาณสีเหลืองแตก หรือ Yellow Broken**

ในกรณีที่ Signal กำลังแสดงไฟสัญญาณสีเหลือง แล้วสัญญาณไฟสีเหลืองเกิดแตก ทำนองเดียวกับ Green Broken คือ แสดงกากบาทบนตำแหน่งไฟสีเหลือง และ Signal Degradation ไปเป็นสัญญาณไฟสีแดง ดังรูป



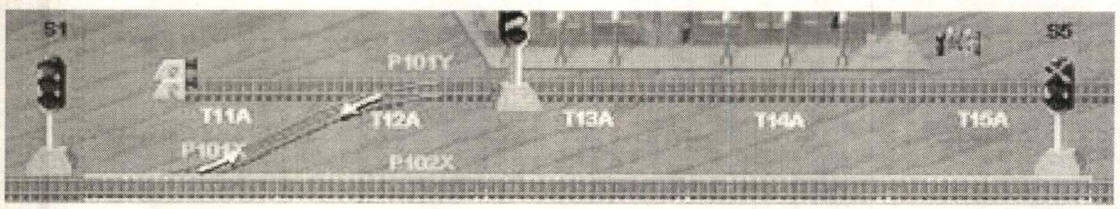
รูปที่ 4.29 การตอบสนองเมื่อ Yellow Broken ขณะแสดงสัญญาณไฟสีเหลือง

- **ไฟสัญญาณสีแดงแตก หรือ Red Broken**

ในกรณีที่ Signal กำลังแสดงไฟสัญญาณสีแดง แล้วสัญญาณไฟสีแดงเกิดแตก ตอบสนองต่อ Event โดยแสดงกากบาทบนตำแหน่งไฟสีแดง ส่วน Signal จะแสดงว่าอยู่ในสถานะ Dark ดังนั้นเพื่อให้เกิดความปลอดภัยตามหลักของ Interlocking Logic จะทำการตอบสนองโดยจะบังคับให้ Signal อันก่อนหน้า (Previous Signal) ของ Signal ที่เกิด Red Broken เปลี่ยนมีไฟสัญญาณเป็นสีแดง ไม่ว่าก่อนหน้านั้นจะแสดงสีอะไรก็ตาม

การตอบสนองต่อ Red Broken ดังรูป

รูปที่ 4.30 ก่อนเกิด Red Broken ที่ S5



รูปที่ 4.31 เกิด Red Broken ที่ S5 ทำให้ S1 เปลี่ยนเป็นสัญญาณไฟสีแดง เพื่อความปลอดภัย

- **สัญญาณไฟแตกทั้งหมด หรือ Dark**

เป็น Event ที่กำหนดให้สัญญาณไฟทุกสีแตกพร้อมกัน เพื่อความสะดวกสำหรับผู้ใช้ในการสั่งค่าเพียงครั้งเดียว ซึ่งผลตอบสนองคือการแสดงกากบาทที่ตำแหน่งสัญญาณไฟทุกดวง และไม่แสดงสีสัญญาณไฟ หรืออยู่ในสถานะ Dark ดังรูปที่ 4.31 ซึ่งทำให้มีผลต่อการแสดงค่าของ Previous Signal เช่นเดียวกับกรณี Red Broken

- **Signal OK**

เป็น Event ที่แทนการซ่อมดวงไฟเรียบร้อยแล้ว ทำให้สัญญาณไฟกลับมางานได้ตามปกติ ซึ่งจะแสดงค่าสัญญาณไฟก่อนที่เกิดการ Broken

4.4.7 Route Setting

เป็นการจัดจราจรเพื่อให้รถไฟเคลื่อนที่ไปบนเส้นทางที่มีการกำหนดให้เท่านั้น การพัฒนาระบบจำลองในส่วนนี้เป็นการทำงานร่วมกันจากทุกๆ อุปกรณ์ โดยเริ่มจากตรวจสอบ Command ที่ได้รับจาก Start Signal และ End Signal เพื่อจะได้รู้ว่าผู้ใช้ต้องการสร้างเส้นทางใด จากนั้นตรวจสอบไปที่อุปกรณ์แต่ละตัวที่เกี่ยวข้องกับเส้นทางนั้นว่ามีสถานะและเงื่อนไขต่างๆ ตรงตามที่ Interlocking Logic กำหนดหรือไม่ ในช่วงที่มีการตรวจสอบเงื่อนไขต่างๆ เรียกว่า Route Reserved และถ้าเงื่อนไขต่างๆ ถูกต้อง จะเปลี่ยนเป็น Route Locked

เมื่อ Route Locked แล้วอุปกรณ์ต่างๆ ที่เกี่ยวข้องกับเส้นทางนั้นจะไม่สามารถถูกเรียกใช้จากเส้นทางอื่นๆ รวมทั้งไม่รับคำสั่งใดๆ ที่อาจส่งมายังอุปกรณ์นั้นโดยตรง เช่น ในกรณีของ Point ถ้า Point เป็นส่วนหนึ่งของ Route จะไม่สามารถส่ง Individual Point Command เพื่อเปลี่ยนทิศทางไปทางอื่นได้

เงื่อนไขต่างๆ ที่กำหนดภายใน Interlocking Logic เป็นไปตาม Interlocking Logic Requirements ในบทที่ 3

ขั้นตอนในการดำเนินการในส่วนของ Route Setting มีดังนี้

- **ตรวจสอบ Command**

เป็นหน้าที่ของ Functional Logic ที่ทำการตรวจสอบว่า Command ที่ได้รับสำหรับการสร้างเส้นทางถูกต้องหรือไม่ ถ้าไม่ถูกต้องก็จะไม่การดำเนินการใดๆ โดย Command ที่ถูกต้องคือ Command ส่งมาจาก Start Signal และ End Signal ที่มีความสอดคล้องกัน สามารถสร้างเส้นทางระหว่างกันได้

ถ้า Command ถูกต้อง (Valid Command) ก็จะตรวจสอบว่าเส้นทางนี้ประกอบด้วยอุปกรณ์ใดบ้าง และต้องการในสถานะใด จากนั้นก็จะส่งความต้องการไปยังอุปกรณ์แต่ละตัวเพื่อตรวจสอบตามเงื่อนไขของ Interlocking Logic ซึ่งดำเนินการกับอุปกรณ์แต่ละชนิดดังนี้

- **Route Start**

ตรวจสอบที่ Signal ซึ่งจะทำหน้าที่เป็น Start Signal โดยเงื่อนไขสำหรับ Route Start คือ

- ไม่ถูก Locked เป็น Start Signal อยู่ก่อน
- ไม่ถูก Locked เป็น Intermediated Signal ของเส้นทางอื่น
- สามารถเป็น End Signal ของเส้นทางอื่นได้ เพื่อรองรับการสร้างเส้นทางต่อเนื่องกัน

- **Intermediated Signals**

คือ Signal ที่ถูก Locked อยู่ภายในเส้นทางโดยที่ไม่ได้เป็น Start Signal หรือ End Signal เงื่อนไข มีดังนี้

- ไม่ถูก Locked เป็น Start Signal หรือ End Signal ของเส้นทางอื่น
- ไม่ถูก Locked เป็น Intermediated Signal อยู่ก่อน
- ไม่ถูก Locked เป็น Intermediated Signal ในทิศทางตรงกันข้าม

- **Track Circuits และ Track Crossing**

เงื่อนไข คือ ไม่ถูก Locked ในทิศทางอื่น และไม่เคยถูก Locked มาก่อน

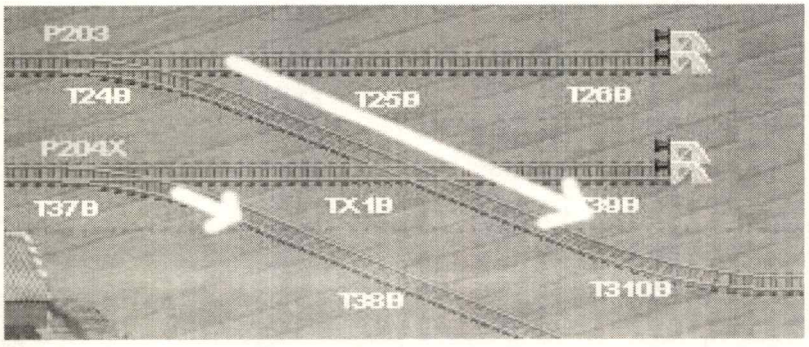
- **Points in Route**

คือ อุปกรณ์ Point ที่ถูก Locked อยู่ภายในเส้นทาง มีเงื่อนไขดังนี้

- Point อยู่ในทิศทางที่เส้นทางต้องการ(Required Position) หรือถ้าไม่อยู่ในทิศทางที่ต้องการ จะต้องสามารถเปลี่ยนทิศทางได้ นั่นคือ ไม่ถูก Occupied รวมทั้ง Couple Point ด้วย
- ไม่ถูก Locked โดยเส้นทางอื่น

- **Flank Protection Points**

คือ อุปกรณ์ Point ที่ไม่ได้อยู่ในเส้นทาง แต่ต้องมีการกำหนดทิศทางตามที่เส้นทางต้องการ เพื่อเป็นการ Interlock ป้องกันไม่ให้รถไฟจากเส้นทางอื่นวิ่งเข้ามาในเส้นทางที่กำหนด ตัวอย่าง ดังรูป



รูปที่ 4.32 Point หมายเลข P204X ทำหน้าที่เป็น Flank

จากรูปที่ 4.32 มีเส้นทางผ่าน Track Crossing หมายเลข TX1B ตามลูกศรด้านบน ดังนั้น เส้นทางนี้ต้องการ Flank Protection ซึ่งทำหน้าที่โดย P204X ทิศทางตามลูกศรด้านล่าง ซึ่งจะทำให้ไม่มีรถไฟวิ่งผ่าน P204X เข้าชนเส้นทางของ TX1B

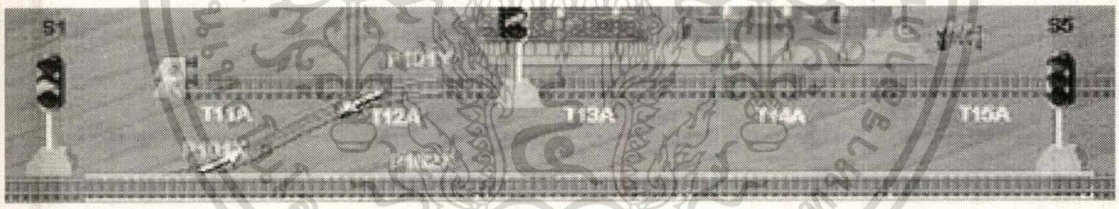
เงื่อนไขของ Flank Protection Point คือ อยู่ในทิศทางที่เส้นทางต้องการ (Required Position) หรือถ้าไม่อยู่ในทิศทางที่ต้องการจะต้องสามารถเปลี่ยนทิศทางได้ นั่นคือ ไม่ถูก Occupied

- **Route End**

ตรวจสอบเงื่อนไขที่ Signal เพื่อทำหน้าที่เป็น End Signal

- ไม่ถูก Locked เป็น End Signal อยู่ก่อน
- ไม่ถูก Locked เป็น Intermediated Signal ของเส้นทางอื่น
- สามารถเป็น Start Signal ของเส้นทางอื่นได้ เพื่อรองรับการสร้างเส้นทางต่อเนื่องกัน

ถ้าเงื่อนไขในทุกอุปกรณ์ถูกต้องทั้งหมดก็เปลี่ยนสถานะของเส้นทางจาก Route Reserved เป็น Route Locked และกำหนดสถานะนี้ให้กับอุปกรณ์ทุกตัวที่อยู่ภายในเส้นทางด้วย จากนั้นก็จะแสดงผลทางหน้าจอเพื่อบอกผู้ใช้ว่าเข้าสู่ Route Locked แล้วโดยใช้สีขาวแสดงบนเส้นทางจาก Start Signal จนถึง End Signal ดังรูป



รูปที่ 4.33 เมื่อสร้างเส้นทางจาก S1 ไปยัง S5 สำเร็จ จะแสดงสีขาวบริเวณเส้นทาง

4.4.8 Signal Control

เป็นขั้นตอนต่อเนื่องหลังจากสร้างเส้นทางเรียบร้อยแล้ว ก็จะมีการตรวจสอบเงื่อนไขต่างๆ เพื่อแสดงสัญญาณไฟอนุญาตให้รถไฟวิ่งเข้ามาในเส้นทาง (Proceed Aspect) แยกเป็นแต่ละส่วนดังนี้

- **Route Start**

Start Signal มีเงื่อนไข คือ ต้องอยู่ในสถานะ Route Locked และถูก Locked เป็น Start Signal

- **Intermediated Signals** อยู่ในสถานะ Route Locked

- **Track Circuits และ Track Crossings** ต้องไม่ถูก Occupied และอยู่ในสถานะ Route Locked รวมถึง Track Circuits ที่ติดตั้งอยู่กับอุปกรณ์ Points และ Signal
- **Points in Route** ต้องอยู่ในทิศทางที่ถูกต้องตามที่เส้นทางต้องการ (Required Position) และอยู่ในสถานะ Route Locked
- **Flank Protection Points** ต้องอยู่ในทิศทางที่ถูกต้องตามที่เส้นทางต้องการ (Required Position) และอยู่ในสถานะ Flank Protection
- **Route End** ต้องอยู่ในสถานะ Route Locked และถูก Locked เป็น End Signal

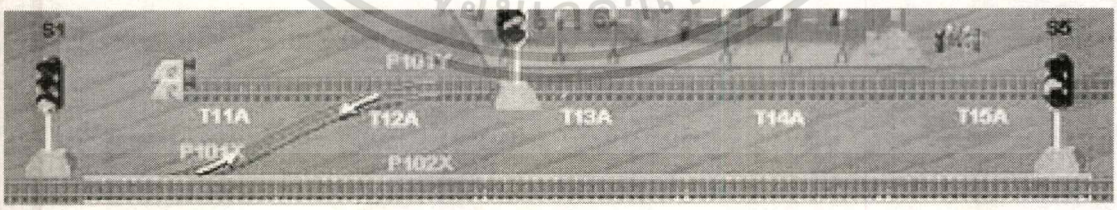
ถ้าเงื่อนไขข้างต้นจะต้องถูกต้องทั้งหมด Start Signal จะได้รับการอนุญาตให้แสดง Proceed Aspect

เงื่อนไขในการแสดงค่า Proceed Aspect ต้องเป็นจริงตลอดเวลา ถ้ามีช่วงใดช่วงหนึ่งที่เงื่อนไขไม่เป็นจริงเพียงข้อเดียว Start Signal จะเปลี่ยนเป็นแสดงค่า Stop Aspect ทันที

- **Aspect Sequence**

เมื่อ Start Signal ได้รับอนุญาตให้แสดง Proceed Aspect ขั้นตอนต่อไปคือพิจารณาควรจะแสดงสีอะไร เนื่องจากค่า Proceed Aspect เป็นไปได้ทั้งสัญญาณไฟสีเหลือง และสัญญาณไฟสีเขียว ดังนั้น Start Signal จะแสดงสัญญาณไฟสีอะไรขึ้นอยู่กับลำดับการแสดงไฟสัญญาณ (Aspect Sequence) ซึ่งมีเงื่อนไขตาม Interlocking Logic วิธีการคือ ตรวจสอบสถานะไฟสัญญาณของ End Signal แล้วนำมากำหนดค่าสัญญาณไฟของ Start Signal ดังนี้

- ถ้า End Signal มีสีแดง แล้ว Start Signal แสดงสีเหลือง ดังรูปที่ 4.34
- ถ้า End Signal มีสีเหลือง หรือ สีเขียว แล้ว Start Signal แสดงสีเขียว ดังรูปที่ 4.34



รูปที่ 4.34 S1 เป็น Start Signal แสดงไฟสีเขียวเมื่อ S5 ซึ่งเป็น End Signal แสดงไฟสีเหลือง

- ถ้า End Signal ดวงไฟแตกจนทำให้มีสถานะเป็น Dark แล้ว Start Signal แสดงสีแดง ดังรูปที่ 4.34

4.4.9 Route Releasing

คือการที่ยกเลิกเส้นทางจากสถานะ Route Locked สู่สภาพปกติ แบ่งได้เป็น 2 ลักษณะคือ Route Release จากการได้รับ Command จากผู้ใช้ ส่วนอีกลักษณะคือ Route จะ Released by Train Passage

4.4.9.1 Route Released by Command

ในส่วนที่ได้รับ Command แบ่งย่อยออกเป็น 2 ประเภทคือ

- **Route Cancel Command**

เมื่อ Start Signal ได้รับ Command นี้ Functional Logic จะทำการตรวจสอบว่าเป็น Command ที่ถูกต้องหรือไม่ รายละเอียดในหัวข้อ Command ของ Signal Class ถ้า Command ถูกต้อง จะถูกส่งต่อมายัง Interlocking Logic เพื่อตรวจสอบเงื่อนไขว่าจะสามารถ Release Route ได้หรือไม่ โดยตรวจสอบที่อุปกรณ์ต่างๆ ที่เป็นส่วนประกอบของเส้นทาง

เงื่อนไขที่ตรวจสอบ มีดังนี้

- อุปกรณ์ทุกตัวต้องยังคงมีสถานะ Route Locked
- Track Circuits และ Track Crossing ไม่มีการ Occupied ภายในเส้นทาง

การตอบสนอง ต่อ Command มีดังนี้

- ทุกอุปกรณ์ จะเปลี่ยนสถานะจาก Route Locked สู่สภาพปกติ และไม่แสดงสีขาวบริเวณเส้นทาง
- Start Signal เปลี่ยนการแสดงไฟสัญญาณจาก Proceed Aspect เป็น Stop Aspect
- ถ้า End Signal ทำหน้าที่เป็น Start Signal ของเส้นทางอื่นด้วย จะ Released เฉพาะสถานะที่ถูก Locked เป็น Route End เท่านั้น แต่ยังคงสถานะ Start Signal ของเส้นทางอื่นอยู่ ไม่เกี่ยวข้องกัน

- **Emergency Route Cancel Command**

เป็น Critical Command ที่ใช้ยกเลิกเส้นทางในกรณีอื่นๆ ที่ไม่สามารถใช้ Route Cancel Command ได้

เงื่อนไขที่ตรวจสอบ มีดังนี้

- End Signal ยังคงมีสถานะ Route Locked
- อุปกรณ์อื่นๆ ที่เป็นส่วนประกอบของเส้นทาง ไม่จำเป็นต้องมีสถานะ Route Locked
- Track Circuits และ Track Crossing สามารถถูก Occupied ได้

การตอบสนอง ต่อ Command

- ทุกอุปกรณ์ จะเปลี่ยนสถานะจาก Route Locked สู่สภาพปกติ และไม่แสดงสีขาวบริเวณเส้นทาง
- Start Signal เปลี่ยนการแสดงไฟสัญญาณจาก Proceed Aspect เป็น Stop Aspect
- ถ้า End Signal ทำหน้าที่เป็น Start Signal ของเส้นทางอื่นด้วย จะ Released เฉพาะสถานะที่ถูก Locked เป็น Route End เท่านั้น แต่ยังคงสถานะ Start Signal ของเส้นทางอื่นอยู่ ไม่เกี่ยวข้องกัน
- Track Circuit และ Track Crossing ที่ถูก Occupied ยังคงแสดงสถานะ Occupied อยู่ ซึ่งเป็นสีแดง

4.4.9.2 Route Released by Train Passage

Route จะทำการ Released หลังจากการเคลื่อนที่ผ่านของรถไฟโดยอัตโนมัติ ซึ่งเงื่อนไขในการเคลื่อนที่ของรถไฟต้องเป็นไปตามลำดับอย่างถูกต้อง โดยเริ่มจาก Start Signal จนถึง End Signal ดังนั้นสิ่งที่พิจารณาคือ ตำแหน่งของรถไฟ และสถานะของอุปกรณ์ที่เป็นส่วนประกอบของเส้นทาง โดยมีเงื่อนไขต่างๆ ดังนี้

- **Route Start**

ตรวจสอบที่ Signal ซึ่งจะทำหน้าที่เป็น Start Signal โดยเงื่อนไขสำหรับ Route Start คือ

- Signal ถูก Locked ในสถานะเป็น Start Signal
- Track Circuit ที่ติดตั้งอยู่ที่ Start Signal ถูก Occupied และมีการเก็บค่าตัวแปร เรียกว่า Approach Section ถูก Occupied
- First Section ของเส้นทางถูก Occupied และมีการเก็บค่าตัวแปร
- ถ้า Approach Section Free (หลังจาก Occupied) และ First Section ถูก Occupied แล้ว Route Start เริ่ม Released โดยเปลี่ยนแปลงค่าจาก Route Locked ไปสู่สภาพปกติ

- **Intermediated Signals**

Release ตามเงื่อนไข Train Passage คือ

- อยู่ในสถานะ Route Locked
- Start Signal Released
- Previous Section Released
- Current Section Free

- **Track Circuits และ Track Crossing**

เช่นเดียวกับ Intermediated Signal

- **Points in Route**

เช่นเดียวกับ Intermediated Signal

- **Flank Protection Points**

Released จากการเป็น Flank Protection เมื่อช่วงของเส้นทางที่ทำหน้าที่ป้องกัน Released ไปแล้ว

- **Route End**

เป็นไปได้ 2 กรณี

กรณีแรก เช่นเดียวกับ Intermediated Signal คือรถไฟเคลื่อนที่ผ่านไปยัง Route ต่อไป

กรณีที่สอง รถไฟมาจอดที่บริเวณ End Signal เป็นระยะเวลาหนึ่ง สำหรับระบบจำลองใช้เวลาไว้ที่ประมาณ 5 วินาที โดยเริ่มนับเวลาเมื่อ Track Circuit ของ End Signal ถูก Occupied เมื่อเวลาหมด ก็จะสามารถ Released ได้โดยอัตโนมัติ เงื่อนไขที่สนใจคือ

- อยู่ในสถานะ Route Locked โดยถูก Locked เป็น End Signal
- Track Circuit ของ End Signal ถูก Occupied
- เวลาหมด

หลังจาก Route Released ได้ทั้งหมดจะตั้งค่าตัวแปรต่างๆ ที่เกี่ยวข้องในการดำเนินการยกเลิกเส้นทางไปที่ค่าเริ่มต้น เพื่อรอการสร้างเส้นทางใหม่

บทที่ 5
สรุปการพัฒนาระบบจำลอง

5.1 ผลการพัฒนาระบบจำลอง

ระบบจำลองการควบคุมจราจรทางรถไฟเป็นซอฟต์แวร์ที่ใช้ในการนำเสนอและจำลองการควบคุมจราจรทางรถไฟ เพื่อให้ผู้ใช้สามารถทำความเข้าใจ ฝึกหัดและเรียนรู้การทำงานของระบบรถไฟ และการจัดการควบคุมจราจรทางรถไฟ โดยใช้พื้นฐานของระบบ Signaling System และระบบ Interlocking Logic System ของบริษัท Bombardier Transportation (Thailand) โดยพัฒนาออกแบบมาจากความต้องการส่วนหนึ่งของระบบต้นแบบ Interlocking Logic Package for New South Wales

การออกแบบระบบจำลอง ใช้โครงสร้างของโปรแกรมแบบ Interactive และแสดงผลในรูปแบบกราฟิก คล้ายคลึงกับเกมทางคอมพิวเตอร์ ทำให้โปรแกรมสามารถประมวลผลและแสดงผลตอบสนองต่อการกระทำของผู้ใช้ได้ตลอดเวลา

5.2 การทำงานของระบบจำลอง

การทำงานของระบบจำลอง ประกอบด้วย

1. การจัดการควบคุมดูแลการจราจร เป็นการควบคุมการทำงานของการจัดจราจรทั้งหมด ซึ่งก็คือ การกำหนดเส้นทางต่างๆ เตรียมไว้เพื่อรองรับการเคลื่อนที่ของรถไฟ ซึ่งมีหลายรูปแบบ เช่น การสร้างเส้นทางภายในสถานีเดียวกัน การสร้างเส้นทางระหว่างสถานี หรือการสร้างเส้นทางต่อเนื่องกันหลายๆ เส้นทาง เป็นต้น
2. การแสดงข้อมูลและสถานะของอุปกรณ์ต่างๆ ซึ่งการแสดงผลจะแตกต่างกันในแต่ละชนิดของอุปกรณ์
3. การจัดการกับคำสั่งต่างๆ ที่ได้รับจากผู้ใช้ ผู้ใช้สามารถส่งคำสั่งต่างๆ เข้าสู่ระบบจำลองผ่านอุปกรณ์เมาส์และคีย์บอร์ด โดยคำสั่งแบ่งออกเป็น 2 ประเภทหลัก คือ คำสั่งที่เกี่ยวข้องกับการกำหนดเส้นทางทั้งหมด และคำสั่งที่สั่งการทำงานไปที่อุปกรณ์โดยตรง ได้แก่คำสั่งที่ใช้ในการเปลี่ยนทิศทางของอุปกรณ์ Point (Individual Point Command)
4. การตอบสนองต่อเหตุการณ์ต่างๆ ที่เกิดขึ้นในเส้นทาง ซึ่งเหตุการณ์ต่างๆ สามารถกำหนดโดยผู้ใช้ และเหตุการณ์ที่เป็นผลต่อเนื่องมาจากการตอบสนองของระบบ เช่น การเคลื่อนที่ของรถไฟ เป็นต้น

5. ประมวลผลโดยใช้หลักการของ Interlocking Logic System
6. การแสดงผลของระบบในรูปแบบกราฟิก โดยผ่าน DirectX Runtime เพื่อที่ผู้ใช้สามารถเห็นได้อย่างชัดเจนเข้าใจง่ายและเพิ่มความน่าสนใจ

5.3 ประโยชน์ที่ได้รับ

เนื่องจากระบบการจัดจราจรทางรถไฟมีความซับซ้อน ผู้ที่ทำงานใช้เวลาศึกษาและต้องอาศัยประสบการณ์จากการทำงานเพื่อที่จะสามารถเรียนรู้และเกิดความเข้าใจ ดังนั้นการพัฒนาระบบจำลองขึ้นมาจะเป็นการช่วยทำให้ผู้ที่ทดลองใช้สามารถทำความเข้าใจและเรียนรู้หลักการทำงานเบื้องต้นของระบบการควบคุมจราจรทางรถไฟได้อย่างรวดเร็ว รวมทั้งเพิ่มความน่าสนใจในการศึกษาและทดลองปฏิบัติ เพื่อเป็นพื้นฐานสำหรับการประยุกต์ใช้กับระบบการทำงานจริงต่อไป นอกจากนี้สามารถนำระบบจำลองไปใช้ในส่วนการนำเสนอเพื่อประชาสัมพันธ์ระบบการทำงานของการจัดจราจรทางรถไฟได้อีกทางหนึ่ง

บรรณานุกรม

มานพ พรเพียรวิชานนท์. 2544. **ก้าวสู่ Game Developer มือโปรกับ DirectX**. กรุงเทพฯ:วิตตี้ กรุ๊ป.

De Sousa, B. M. 2002. **Game Programming All In One**. California: Premier Press.

Keepin, J. 2000. **Interlocking Logic Manual ILL950_RT**. Reading: DaimlerChrysler Rail System (Signal).

Kleimola, J. 1998. **Ebiscreen V2.0**. Stockholm: ABB Daimler-Benz Transportation Signal.

Murphy, E. 1999. **Introduction To Railway Signaling**. Bangkok: DaimlerChrysler Rail System (Signal).

Snook, V. 2000. **RAC User Case Description**. Sidney: DaimlerChrysler Rail System (Signal).

Walsh, P. 2001. **The Zen of Direct3D Game Programming**. California: Prima Tech.

ประวัติผู้เขียน

ชื่อผู้เขียน	นางสาว นิศากาญจน์ นิปริยาย
วันเกิด	2 กันยายน 2519
สถานที่เกิด	นครศรีธรรมราช
วุฒิการศึกษาระดับปริญญาตรี	วิศวกรรมศาสตร์บัณฑิต (วิศวกรรมไฟฟ้า)
สถานที่สำเร็จการศึกษา	มหาวิทยาลัยมหิดล
สถานที่ทำงาน	Bombardier Transportation (Thailand)