การพัฒนาโปรแกรมสำหรับแก้ปัญหาทางคณิตศาสตร์ สำหรับงานทางด้านวิทยาศาสตร์

SCIENCEFIC SOFTWARE FOR MATHMATICAL SOLVING AND SIMULATION

คเชนทร์ ตั้งเขตมงคลสุข
บุญญภัทร พงษ์เก่า
ปิติพงษ์ อักโข

ปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต
ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2550

การพัฒนาโปรแกรมสำหรับแก้ปัญหาทางคณิตศาสตร์ สำหรับงานทางด้านวิทยาศาสตร์

SCIENCEFIC SOFTWARE FOR MATHMATICAL SOLVING AND SIMULATION

คเชนทร์ ตั้งเขตมงคลสุข

บุญญภัทร พงษ์เก่า

ปิติพงษ์ อักโข

ปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต

ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์

คณะวิทยาศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2550

SCIENCEFIC SOFTWARE FOR MATHMATICAL SOLVING AND SIMULATION

KACHANE TANGKETMONGKOLSUK

BOONYAPAT PONGKAO

PITIPONG AUKKO

A SPECIAL PROJECT SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIRMENT FOR THE DEGREE OF BACHELOR OF SCIENCE

DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE

FACULTY OF SCIENCE

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

ACADEMIC YEAR 2007

หัวข้อปัญหาพิเศษ การพัฒนาโปรแกรมสำหรับแก้ปัญหาทางคณิตศาสตร์สำหรับงานทางด้านวิทยาศาสตร์
SCIENCEFIC SOFTWARE FOR MATHEMATICAL SOLVING AND SIMULATION

ชื่อนักศึกษา นายคเชนทร์ ตั้งเขตมงคลสุข 47050316
นายบุญญภัทร พงษ์เก่า 47050338
นายปีติพงษ์ อักโข 47050341

ภาควิชา คณิตศาสตร์และวิทยาการคอมพิวเตอร์

สาขาวิชา วิทยาการคอมพิวเตอร์

อาจารย์ที่ปรึกษา รองศาสตราจารย์ธีรวัฒน์ ประกอบผล
รองศาสตราจารย์ภัคคินี ชิตสกุล

ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้นับปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร วิทยาศาสตรบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์ ประจำปีการศึกษา 2550

คณะกรรมการสอบ	ลายมือชื่อ
อาจารย์ศังกรศรัณย์ ล่องชูผล ประธานกรรมการ	
ดร.นวลสวาท หิรัญสกลวงศ์ กรรมการ	
รองศาสตราจารย์ธีรวัฒน์ ประกอบผล กรรมการและอาจารย์ที่ปรึกษา	
รองศาสตราจารย์ภัคคินี ชิตสกุล กรรมการและอาจารย์ที่ปรึกษา	

(รองศาสตราจารย์ไพรบูลย์ พันธรักษ์พงษ์)
หัวหน้าภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์

แด่ทุกคนที่คอยช่วยเหลือและให้กำลังใจมาโดยตลอด

คเชนทร์

แด่คุณพ่อคุณแม่ผู้เป็นที่รัก

บุญญภัทร

แด่ครูอาจารย์ทุกท่านที่ประสิทธิ์ประสาทวิชาความรู้

ปิติพงษ์

หัวข้อปัญหาพิเศษ	การพัฒนาโปรแกรมสำหรับแก้ปัญหาทางคณิตศาสตร์สำหรับงานทางด้านวิทยาศาสตร์
ชื่อนักศึกษา	นายคเชนทร์ ตั้งเขตมงคลสุข 47050316
	นายบุญญภัทร พงษ์เก่า 47050338
	นายปีติพงษ์ อักโข 47050341
ปริญญา	วิทยาศาสตรบัณฑิต
ภาควิชา	คณิตศาสตร์และวิทยาการคอมพิวเตอร์
สาขาวิชา	วิทยาการคอมพิวเตอร์
ปีการศึกษา	2550
อาจารย์ที่ปรึกษา	รองศาสตราจารย์ธีรวัฒน์ ประกอบผล
	รองศาสตราจารย์ภัคคินี ชิตสกุล

บทคัดย่อ

ปัญหาพิเศษนี้มีจุดประสงค์เพื่อพัฒนาโปรแกรมสำหรับแก้ปัญหาทางคณิตศาสตร์ โดยที่ผลลัพธ์จะถูกแสดงด้วยกราฟฟิค โดยโปรแกรมนี้ถูกพัฒนาขึ้นด้วยภาษาจาวาซึ่งมีข้อดีตรงที่การใช้งานได้หลายระบบปฏิบัติการ เพื่อแสดงถึงประสิทธิภาพการทำงานของโปรแกรมเราได้นำระเบียบไฟไนต์อิลิเมนท์ โดยใช้สมการลาพลาสมาเป็นกรณีศึกษาและนำมาแก้ปัญหาด้วยโปรแกรมของเรา และนำผลลัพธ์ที่ได้มาเปรียบเทียบกับโปรแกรม MATLAB ซึ่งเป็นโปรแกรมที่มีความน่าเชื่อถือและได้รับการยอมรับจากผู้ใช้งานทั่วโลก

Special Project Title	SCIENCEFIC SOFTWARE FOR MATHEMATICAL SOLVING AND SIMULATION	
Students	Mr.Kachane Tangketmongkolsuk	47050316
	Mr.Boonyapat Pongkao	47050338
	Mr.Pitipong Aukko	47050341
Degree	Bachelor of Science	
Department	Mathematics and Computer Science, Faculty of Science	
Programme	Computer Science	
Academic Year	2007	
Special Project Advisor	Associate Professor Teerawat Prakobphon	
	Associate Professor Pakkinee Chitsakul	

ABSTRACT

The purpose of this special problem is to develop a program for mathematical solving and shown the result as a graphic. This program is developing base on JAVA programming language that have an advantage to use in many platform of operating system. To show the performance of our program we choose finite elements method as a case study to solving a problem and then to approve that our program is reliable we compare output from our program with MATLAB that is a faithful program and had acceptance from users all over the world.

กิตติกรรมประกาศ

ในการทำปัญหาพิเศษเรื่องการพัฒนาโปรแกรมสำหรับแก้ปัญหาทางคณิตศาสตร์สำหรับงานทางด้านวิทยาศาสตร์สามารถสำเร็จลุล่วงไปด้วยดี ทางคณะผู้จัดทำต้องขอขอบพระคุณ รองศาสตราจารย์ภัคคินี ชิตสกุล และรองศาสตราจารย์ ธีรวัฒน์ ประกอบผล ซึ่งเป็นอาจารย์ผู้รับผิดชอบปัญหาพิเศษฉบับนี้ที่กรุณาให้คำแนะนำและเป็นที่ปรึกษาในการแก้ปัญหาต่าง ๆ รวมทั้งเป็นผู้ตรวจสอบความถูกต้องของปัญหาพิเศษฉบับนี้

นอกจากนี้คณะผู้จัดทำต้องขอขอบพระคุณ บิดา มารดา ที่ได้ให้ความสนับสนุนทางด้านกำลังใจและทุนทรัพย์จนการทำปัญหาพิเศษครั้งนี้สำเร็จด้วยดี รวมทั้งเพื่อน ๆ พี่ ๆ และน้อง ๆ ทุกคนที่ให้ความช่วยเหลือในด้านต่าง ๆ เกี่ยวกับปัญหาพิเศษไว้ ณ ที่นี้

สารบัญ

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญภาพ

ภาพที่ **หน้า**

สารบัญภาพ(ต่อ)

ภาพที่ **หน้า**

สารบัญภาพ(ต่อ)

ภาพที่ **หน้า**

สารบัญภาพ(ต่อ)

ภาพที่ **หน้า**

สารบัญตาราง

ตารางที่ **หน้า**

คำอธิบายสัญลักษณ์และคำย่อ

1. ∂ เป็นสัญลักษณ์แทนการอนุพันธ์บางส่วน

   เช่น $\frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}$

2. $\sum$ เป็นสัญลักษณ์แทนการบวกสะสม ถ้าในส่วนของตัวแปรภาษาจะหมายถึงเซตของอินพุตอักขระที่เป็นไปได้ทั้งหมด

3. | เครื่องหมายแทนหรือ เช่น A|B (A หรือ B)

4. * สัญลักษณ์ใน expression หมายถึงมีค่า 0 หรือมากกว่า

5. + สัญลักษณ์ใน expression หมายถึงมีค่าตั้งแต่ 1 ขึ้นไป

6. $\int$ เป็นสัญลักษณ์แทนการหาผลรวมของค่าภายในพื้นที่ที่กำหนด

7. **const** เป็นคำย่อมาจาก constant หมายถึง ค่าคงที่

8. ::= หมายถึงการ Derivation

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

เนื่องจากในปัจจุบัน เกิดการแข่งขันในเรื่องการพัฒนาซอฟต์แวร์กันมากขึ้นระหว่างบริษัทผู้พัฒนาซอฟต์แวร์ ทำให้มีการศึกษาค้นคว้าวิธีการเพื่อที่จะลดงบประมาณหรือต้นทุนในการพัฒนาซอฟต์แวร์ลง ซึ่งมีวิธีการที่น่าสนใจแบบหนึ่งคือการนำซอฟต์แวร์กลับมาใช้ใหม่โดยเฉพาะในส่วนของข้อกำหนดความต้องการของซอฟต์แวร์ ปัญหาพิเศษนี้ จึงได้ทำการศึกษาถึงวิธีการค้นคืนเอกสารความต้องการซอฟต์แวร์ ในรูปแบบของเอกสารการบรรยายยูสเคส โดยนำแบบจำลองแอลเอสไอที่นิยมใช้ในการค้นคืนข้อมูลตามเว็บไซต์ เข้ามาประยุกต์ใช้ในการค้นคืนเอกสารการบรรยายยูสเคสเพื่อนำกลับมาปรับใช้ใหม่ ซึ่งจะช่วยทำให้ลดระยะเวลาและงบประมาณในการพัฒนาซอฟต์แวร์ลงได้

1.2 วัตถุประสงค์

1.2.1 เพื่อศึกษาว่าการค้นคืนเอกสารการบรรยายยูสเคสโดยใช้แบบจำลองแอลเอสไอสามารถให้ผลที่ดีกว่าการค้นคืนเอกสารการบรรยายยูสเคสโดยใช้วิธีการทั่วไป

1.2.2 เพื่อสร้างเครื่องมือที่จะนำมาช่วยในการค้นคืนเอกสารการบรรยายยูสเคส

1.3 ข้อจำกัดและขอบเขตของปัญหา

1.3.1 เอกสารการบรรยายยูสเคสที่นำมาใช้ทดลองในปัญหาพิเศษนี้ จะต้องอยู่ในรูปแบบที่กำหนดไว้เท่านั้น [1]

1.3.2 รูปแบบและชุดข้อมูลของการทดลองจะต้องเป็นไปตามการทดลองของเอกสารอ้างอิง [1] เพราะจะต้องมีการนำผลการทดลองที่ได้จากปัญหาพิเศษนี้ไปเปรียบเทียบกับผลการทดลองของเอกสารดังกล่าว

1.4 ขั้นตอนการดำเนินการ

1. ศึกษารายละเอียดของการค้นคืนข้อมูลสารสนเทศ
2. ศึกษารายละเอียดของแบบจำลองแอลเอสไอ
3. กำหนดขอบเขตของการศึกษาและการทดลอง
4. วิเคราะห์ และออกแบบระบบงาน
5. พัฒนาระบบงาน
6. ทำการทดลอง และปรับปรุงระบบงาน
7. สรุปผล และจัดทำเอกสาร

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1. ทำให้ทราบได้ว่าการนำแบบจำลองแอลเอสไอมาใช้ในการค้นคืนเอกสารการบรรยายยูสเคสจะให้ผลลัพธ์ที่ดีกว่าการใช้วิธีการค้นคืนเอกสารโดยทั่วไป หรือไม่
2. สามารถนำวิธีการที่นำเสนอไปค้นคืนเอกสารการบรรยายยูสเคสเพื่อนำกลับมาปรับใช้ใหม่ ซึ่งจะช่วยลดต้นทุนหรืองบประมาณของการพัฒนาซอฟต์แวร์ลงได้
3. สามารถนำเครื่องมือที่พัฒนาขึ้นมาไปใช้ในการค้นคืนเอกสารการบรรยายยูสเคสได้

คู่มือการทำปัญหาพิเศษเล่มนี้มีรายละเอียดดังนี้ บทที่ 1 กล่าวถึงบทนำของการทำปัญหาพิเศษ บทที่ 2 กล่าวถึงทฤษฎีหรืองานวิจัยที่เกี่ยวข้อง บทที่ 3 กล่าวถึงวิธีดำเนินงานวิจัย บทที่ 4 กล่าวถึงผลการทดลอง บทที่ 5 กล่าวถึงสรุปผลการทดลอง การอภิปราย และข้อเสนอแนะของการทำปัญหาพิเศษ ในส่วนของภาคผนวกนั้น ภาคผนวก ก. กล่าวถึง วิธีการติดตั้งโปรแกรม ภาคผนวก ข. กล่าวถึงระบบที่ใช้ในการทดลอง

บทที่ 2
ทฤษฎีและหลักการที่นำมาใช้ในการพัฒนาโปรแกรม

ทฤษฎีและหลักการที่นำมาใช้ในการพัฒนาโปรแกรม แบ่งได้เป็นสามส่วนใหญ่ๆคือ

1. พื้นฐานและหลักการของการสร้างตัวแปลภาษา (Interpreter)
2. หลักการทำงานและการใช้งานเครื่องมือที่จะนำมาใช้ในการสร้างตัวแปลภาษา
3. พื้นฐานหลักการของคอมพิวเตอร์กราฟิก (Computer Graphics) โอเพนจีแอล (OpenGL) และจาวาโอเพนจีแอล (JOGL)
4. ทฤษฎีและหลักการทางคณิตศาสตร์ที่เกี่ยวข้องกับโปรแกรม

2.1 ทฤษฎีของตัวแปลภาษาและหลักการของการสร้าง

นิยามของตัวแปลภาษาแบบอินเตอร์พรีเตอร์คือ เป็นโปรแกรมชนิดหนึ่งที่รับคำสั่งที่เขียนขึ้นด้วยภาษาหนึ่งเป็นอินพุตซึ่งเรียกว่าซอร์สโปรแกรม (Source Program) โดยการรับเข้ามาทำการประมวลผลแล้วจะสร้างผลลัพธ์ออกมาเป็นโปรแกรมที่เขียนขึ้นออกมาทันทีทีละคำสั่งหรือบรรทัด โดยเราจะเรียกว่าเป็นโปรแกรมเป้าหมายออกมา ซึ่งถ้ามีข้อผิดพลาดก็จะแสดงข้อความผิดพลาดให้เห็น

2.1.1 โครงสร้างของตัวแปลภาษาทั่วไป (ทั้ง Compiler และ Interpreter)

- **Lexical analyzer** (Lexer หรือ Lex) จะทำหน้าที่อ่านโปรแกรมต้นฉบับเรียกว่าซอร์สโปรแกรม Lexer จะทำการสแกนข้อมูล (scan) ในโปรแกรมต้นฉบับซึ่ง Lexer จะมองเห็นเป็นคำสั่งที่เรียงกันอยู่เป็นบรรทัดยาวๆ Lexer จะทำการสแกนตัวอักษรทั้งหมดในบรรทัดนั้นซึ่งก็คือคำสั่งทั้งหมดที่มีอยู่ในโปรแกรมต้นฉบับ ส่วนที่เป็นคำอธิบายเพิ่มเติม (comment) ทั้งหลายจะถูกลบทิ้งไป ในเวลาเดียวกัน Lexer จะทำการสร้างตารางข้อมูล (Symbol Table) ขึ้นมาด้วย ในระหว่างที่ Lexer ทำการสแกนไปนั้นก็จะทราบว่าในโปรแกรมไปนั้นก็จะทราบว่าในโปรแกรมต้นฉบับนั้นมีอะไรอยู่บ้าง แต่ Lexer ยังไม่สามารถบอกได้ว่า สัญลักษณ์นั้นเป็นอะไร ซึ่งสามารถรายงานความผิดพลาดออกมาได้ด้วย

- **Syntax analyzer** ส่วนของคอมไพลเลอร์ตรงนี้จะบอกให้ทราบชื่ออ้างอิง (identifier) ต่างๆ นั้นนำมาใช้งานอย่างถูกต้องหรือไม่และโทเคน (token) มีอยู่ครบหรือไม่ ซึ่งเป็นการตรวจสอบรูปแบบทั่วไป (syntax) ของโปรแกรมนั่นเอง

- **Semantic analyzer** ส่วนวิเคราะห์ความหมายจะดูความหมายของประโยคคำสั่งเช่น a=x; ในส่วนของ Lexical analyzer จำทำการ scan “a”, “=”, “x”, และ “;” ในส่วนของ syntax

analyzer ก็จะดูว่า syntax นั้นถูกต้องหรือไม่ (a = x;) มาถึงส่วนวิเคราะห์ความหมาย จะทำการดูความหมายซึ่ง "a" และ "x" จะต้องมี type ที่เหมือนกันหรือเข้ากันได้ (compatible) ซึ่งเป็นหน้าที่ของ semantic analyzer ทำการดูลักษณะของความหมาย

- **Intermediate Code Generator** เป็นขั้นตอนในการสร้างโค้ดในระดับเบื้องต้น ซึ่งในขั้นตอนนี้อาจมีการทำ Optimization หรือการปรับแต่งลดทอนโค้ดให้ดูดีขึ้นอีกด้วย

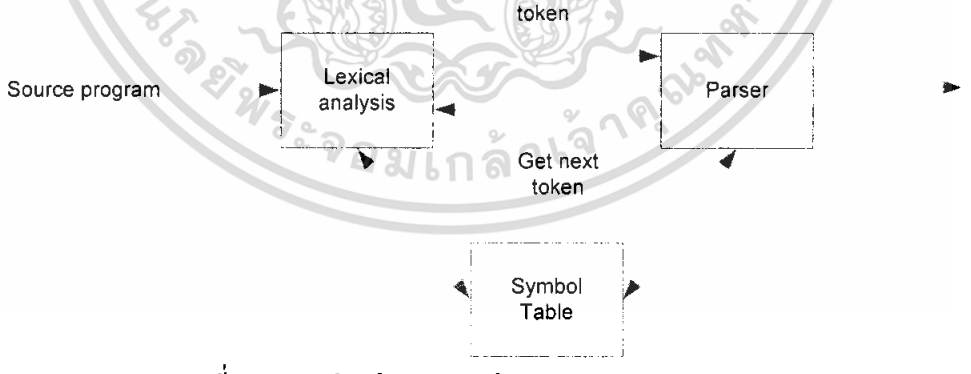
- **Code Generation** ทำการสร้างโค้ดซึ่งเป็นภาษาระดับต่ำให้เสร็จสมบูรณ์

นอกจากนี้ยังมีส่วนที่เป็นตัวจัดการความผิดพลาดที่เกิดขึ้นซึ่งจะเชื่อมกับทุกๆขั้นที่กล่าวมาข้างต้นและยังมีส่วนที่เรียกว่า Symbol Table ซึ่งเป็นตารางที่ใช้เก็บสัญลักษณ์หรือชื่อของตัวแปรหรือที่เรียกว่า identifier นั่นเอง

ซึ่งโดยปกติแล้วหากเป็นการสร้างตัวแปลภาษาแบบ Interpreter มักจะอยู่ที่ขั้นที่ 4 ซึ่งอาจจะทำการแปลงเป็น Intermediate Code ก่อนแล้วทำงานหรือ อาจจะหยุดที่ขั้นที่ 3 คือพาร์สทรี (Parse Tree) ซึ่งจะทำมาทำการประมวลผลตามลำกับขั้นที่ได้จากพาสทรีไปเรื่อยๆ ผลลัพธ์จะถูกส่งออกมาเริ่มจากลีฟโหนด (leaf node) ไปจนถึงรูทโหนด (root node)

2.1.2 Lexical Analyzer (Scanner)

หน้าที่หลักของ Lexical Analyzer (หรือสแกนเนอร์) ก็คือการจัดการประมวลผลกลุ่มก้อนของอักษรที่ป้อนเข้าไป จากนั้นจึงทำการตัดออกมาเป็นกลุ่มก้อนของอักษรหลายๆกลุ่ม ซึ่งกลุ่มก้อนของอักษรเหล่านั้นเรียกว่า "โทเคน"



ภาพที่ 2.1 การติดต่อกันระหว่าง Lexical Analyzer กับ Parser

จากแผนภาพจะเห็นได้ว่าการทำงานของ Lexical Analyzer นั้นมีความสัมพันธ์ใกล้ชิดกับพาร์สเซอร์ (Parser) โดย พาร์สเซอร์ จะเป็นตัวร้องขอโทเคน จาก Lexical Analyzer เพื่อนำไปจัดการประมวลผลต่อไป เมื่อ พาร์สเซอร์ ทำการประมวลผลโทเคน ที่ได้จาก Lexical Analyzer เสร็จเรียบร้อยแล้ว ก็จะร้องขอ โทเคน ตัวถัดไปจาก Lexical Analyzer เพื่อนำไปประมวลผลต่อไป

ในการทำงานของ Lexical Analyzer สามารถแบ่งเป็นขั้นตอนหลักๆ ได้ 2 ขั้นตอนก็คือ

1. Scanning เป็นขั้นตอนที่ใช้จัดการอ่านข้อมูลจากอินพุต (input) เข้ามา นอกจากนี้ยังใช้จัดการกับงานง่ายๆอื่นๆ เช่น การกำจัดช่องว่างที่เกิดจากการเว้นวรรคในภาษา FORTRAN เป็นต้น

2. Lexical Analysis เป็นขั้นตอนที่ใช้ในการจัดการประมวลผลกลุ่มของอินพุต ที่ได้จากขั้นตอน scanner ซึ่งในขั้นตอนนี้ประกอบไปด้วยวิธีการที่ซับซ้อนกว่าสแกนเนอร์มาก เช่นการวิเคราะห์โทเคนด้วย state machine

ปัจจัยที่ทำการแยก Lexical Analyzer ออกเป็นหนึ่งงานของตัวแปลภาษา

1. ง่ายต่อการออกแบบและสร้าง การแยกงานในส่วนของ Lexical Analyzer ออกมาทำให้งานบางอย่างง่ายขึ้นมาก เช่น หากเรารวม Lexical Analyzer เข้ากับ พาร์สเซอร์ การจัดการกับช่องว่างที่เกิดจากการเว้นวรรคจะเป็นเรื่องที่ยุ่งยากซับซ้อนมาก การแยก Lexical Analyzer ออกมาช่วยลบช่องว่างเหล่านั้นทิ้งไปนับว่าเป็นวิธีการแก้ปัญหาที่ดี

2. เป็นการเพิ่มประสิทธิภาพของตัวแปลภาษา การแบ่งแยกงานออกเป็นส่วนๆ เช่นนี้ทำให้เราสามารถเน้นความสามารถในการจัดการของหน่วยประมวลผลในงานที่ต้องใช้เวลานาน หรือ ต้องใช้ทรัพยากรของระบบมากได้

3. ทำให้จัดการปรับแต่งเคลื่อนย้ายตัวแปลภาษาได้ง่าย เงื่อนไขพิเศษบางประการของแต่ละภาษาสามารถถูกขจัดออกไปในขั้นของ Lexical Analyzer ได้ จึงง่ายต่อการเปลี่ยนแปลงอินพุต

หน่วยของ Lexical Analyzer

ในการทำงานของ Lexical Analysis จะมีการจัดการประมวลผลกับกลุ่มของอักขระที่เข้ามาเสมอๆ เราเรียกกลุ่มของอักขระจากอินพุต ที่อยู่ในรูปแบบที่มีความหมายว่า Lexeme ซึ่ง Lexeme หลายชนิด จะถูกจัดเข้าเป็นหมวดหมู่ด้วยชนิดของ โทเคน

Token	Sample Lexemes	Informal Description of Pattern
const	const	const
if	if	if
relation	<, <=, =, <>, >, >=	< or <= or = or <> or >= or >
id	pi,count,D2	Letter followed by letters and digits
num	3.1416, 0, 6.02E23	Any numeric constant
lifetime	“core dumped”	Any characters between “ and “ expect “

ตารางที่ 2.1 ตัวอย่างของโทเคน

จากรูปข้างต้น เมื่อใดก็ตามที่อ่านข้อมูลเข้ามารวมกันได้เป็นคำว่า pi จะเกิด โทเคน ชื่อ id (ย่อมาจาก identifier แปลว่าตัวแปร) ส่งผ่านไปยัง พาร์สเซอร์ ทันที

นอกจากตัวอย่างข้างต้นแล้ว ยังมี Lexeme ที่มีความซับซ้อนขึ้นไปอีก เช่น id อาจประกอบไปด้วยอักขระกี่ตัวก็ได้ ซึ่งเราจะใช้ regular grammar เข้ามาช่วยในการบ่งชี้ Lexeme ประเภทนี้

Regular expression

บางครั้งในการบ่งชี้ Lexeme นั้นมีความซับซ้อนมาก เช่น identifier (ตัวแปร) ในภาษา Pascal มีหลักการในการประกาศคือ ตัวอักษรนำหน้าตามด้วยตัวอักษรหรือตัวเลขกี่ตัวก็ได้ ในการประกาศ Lexeme ดังกล่าว เราจะนำ Regular Expression มาช่วย ดังตัวอย่าง

letter (letter | digit)*

- สัญลักษณ์ | ในที่นี้มีความหมายว่า "หรือ"
- สัญลักษณ์วงเล็บนั้นใช้จัดกลุ่ม
- สัญลักษณ์ * หมายถึง "0 หรือมากกว่านั้น"

กฎที่ใช้ในการกำหนดขอบเขตของ Regular expression บน Σ มีดังนี้

1. ε เป็น regular expression ที่แสดงถึง $\{\varepsilon\}$ นั่นคือเซตของอักขระว่าง
2. ถ้า a เป็นสัญลักษณ์ใน Σ แล้ว a เป็น regular expression ที่เป็นตัวแทนของ {a}
3. กำหนดให้ r และ s เป็น regular expression ที่เป็นตัวแทนของภาษา L(r) และ L(s) จะได้ว่า
   - 3.1 (r) | (s) เป็น regular expression ที่เป็นตัวแทนของ $L(r) \cup L(s)$
   - 3.2 (r)(s) เป็น regular expression ที่เป็นตัวแทนของ L(r)L(s)
   - 3.3 (r)* เป็น regular expression ที่เป็นตัวแทนของ (L(r))*
   - 3.4 (r) เป็น regular expression ที่เป็นตัวแทนของ $L(r)^2$

เราสามารถเรียกภาษาที่เป็นตัวแทนของ Regular expression ได้ว่า regular set

นอกจากนี้ Regular expression ยังมีคุณสมบัติบางประการที่น่าสนใจ ดังจะเห็นได้จากตารางด้านล่าง

Axiom	Description
r|s = s|r	| is commutative
r|(s|t) = (r|s)|t	| is associative
(rs)t = r(st)	*Concatenation is associative*
r|(s|t) = rs|rt	Concatenation distributes over |

(s|t)r = sr|tr	
ϵr = r rϵ = r	ϵ is identity element for concatenation
r* = (r|ϵ)*	Relation between * and ϵ
r** = r*	* is idempotent

ตารางที่ 2.2 คุณสมบัติเชิงคณิตศาสตร์ของ regular expression

เพื่อให้ง่ายต่อการใช้งาน เราอาจจะกำหนด regular expression โดยใช้ชื่อพิเศษ เช่น

letter → A | B | ... | Z | a | b | ... | z

digit → 0 | 1 | ... | 9

id → letter (letter | digit)*

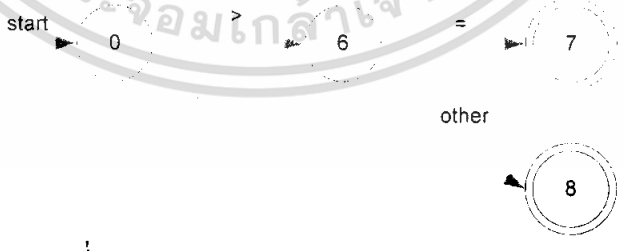
จากตัวอย่างข้างต้น

- letter เป็นชื่อพิเศษแทนตัวอักษรภาษาอังกฤษ
- digit เป็นชื่อพิเศษแทนตัวเลข
- id เป็นชื่อพิเศษแทนกลุ่มของตัวอักษรและตัวเลขที่ขึ้นต้นด้วยตัวอักษร

การรู้จำ โทเคน

ในการรู้จำ โทเคน นั้นเราจะใช้หลักการของ Transition diagram มาอธิบาย

สมมติให้มี Transition Diagram ดังนี้



ภาพที่ 2.2 Transition Diagram สำหรับ >=

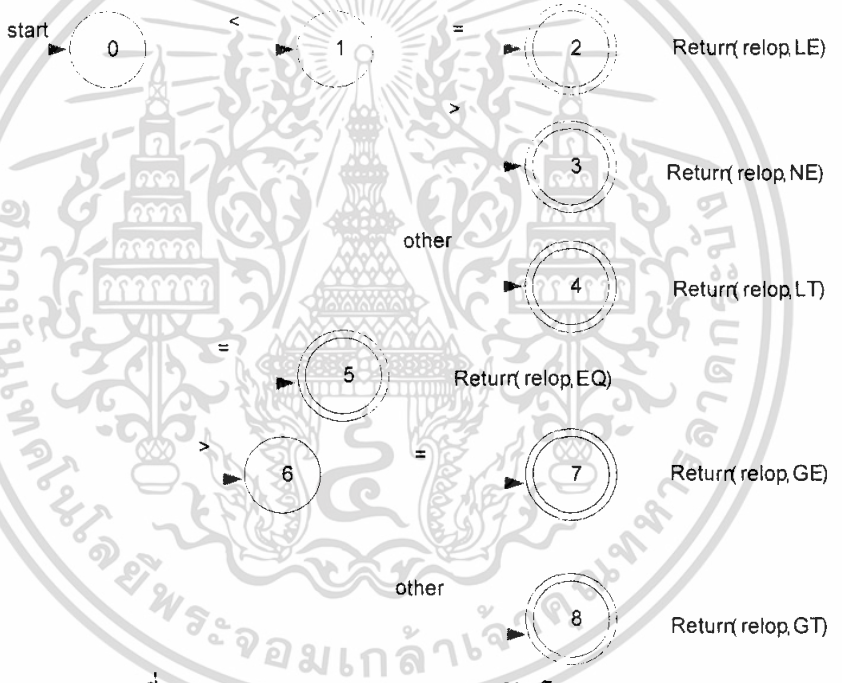
ในการทำงานจะเริ่มจาก state 0 หากอินพุต ที่อ่านเข้ามาตัวแรกคือ > จะทำให้เกิดการเปลี่ยน state ไปยัง state 6 ถ้าอินพุต ตัวที่ 2 เป็น = ก็จะเปลี่ยนไปยัง state 7 และทำการ install_id() >= แต่ถ้าอินพุต ที่อ่านมาตัวที่ 2 เป็นอย่างอื่น ก็จะทำการ install_id() > แทน

ตารางการกระทำในการรู้จำ โทเคน

Procedure	ทำการตรวจสอบใน symbol table	ค่าที่ส่งกลับ
Install_id( )	ถ้ามีอยู่ใน symbol table แล้ว	"0" ถ้าเป็น keywords
		มิฉะนั้นก็ส่ง "pointer" ไปยัง "id"ตัวนั้น
	ถ้าไม่มีอยู่ใน symbol table	"pointer" ไปยัง "id" ตัวนั้น(ตัวใหม่)
Get_token( )	ถ้าเป็น keywords	"token" สำหรับ keyword นั้น
	มิฉะนั้น	"token" สำหรับ "id"

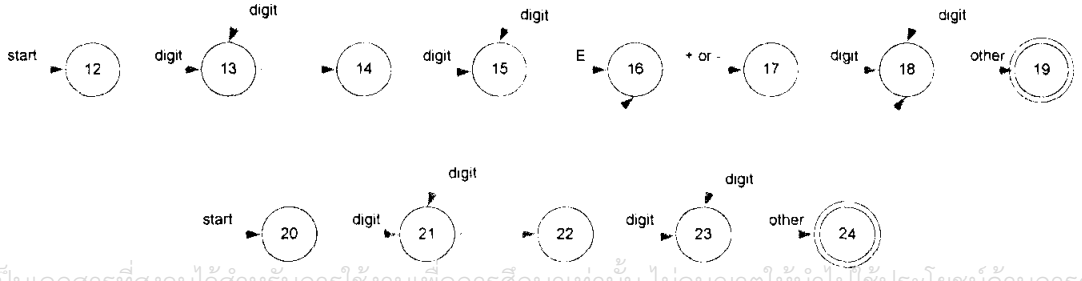
ตารางที่ 2.3 แสดงการกระทำในการรู้จำโทเคน

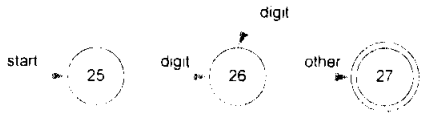
ตัวอย่าง Transition diagram สำหรับ โทเคน relop



ภาพที่ 2.3 transition diagram สำหรับ โทเคน relop

ตัวอย่าง Transition diagram สำหรับ unsigned numbers ในภาษาปาสคาล





ภาพที่ 2.4 transition diagram สำหรับ unsigned number ของภาษาปาสคาล

Context-free grammar

Context-free grammar (CFG) เป็นแกรมม่า ซึ่งในทุกๆกฎของการแปลง จะอยู่ในรูปของ

$$w\,V \rightarrow w$$

โดย V จะเป็น Nonterminal symbol และ w จะแทนชุดของ Terminal Symbol หรือ Nonterminal Symbol (ซึ่งอาจไม่มีตัวอะไรเลยก็ได้) รูปแบบของ context-free นั้นจะเป็นตัวพิสูจน์ว่า Nonterminal นั้นสามารถที่จะเขียนออกมาได้โดยไม่ต้องคำนึงถึงรูปแบบ context ของมันเอง Context-free grammar ยังเป็นตัวหลักในการนำไปใช้ในการออกแบบ ตัวโปรแกรมภาษาและในการออกแบบcompiler และยังใช้ในการ วิเคราะห์ syntax ของ natural language ต่างๆอีกด้วย รูปแบบของ Context-free grammar นั้นจะอยู่ในรูปแบบของ

$$G = (V, \Sigma, R, S)$$

โดยที่

1. V เป็นเซตจำกัดของตัวอักขระหรือตัวแปร *non-terminal* ซึ่งจะมีรูปแบบต่างๆกันตามแต่ละประโยค
2. Σ เป็นเซตจำกัดของ Terminals ที่ไม่ได้เชื่อมต่อกับ V ซึ่งทำให้รูปประโยคสมบูรณ์ขึ้น
3. S เป็นตัวแปรเริ่มต้น, ใช้ในการตรวจหาประโยคของตัวโปรแกรม โดยจะเป็นสมาชิกของ V
4. R เป็นความสัมพันธ์จาก V ไปยัง $(V \cap \Sigma)^*$

อย่างเช่น $\exists\ \omega \in (V \cup \Sigma)^* : (S, \omega) \in R$

สมาชิกของ R จะถูกเรียกว่า *rules* หรือ *productions* ของแกรมม่า

ข้อกำหนดเพิ่มเติม 1

สำหรับทุกๆชุดของ $u,v \in (V \cup \Sigma)^*$, จะมีความหมายว่า u ไปแทนที่ v, ซึ่งเขียนได้ว่า $u \Rightarrow v$ ถ้า $\exists\ (\alpha, \beta) \in R, u_1, u_2 \in (V \cup \Sigma)^*$ แล้ว $u = u_1 \alpha u_2$ และ $v = u_1 \beta u_2$ ดังนั้น v จะเป็นผลที่ได้จาก rule ของ (α, β) ไปยัง u

ข้อกำหนดเพิ่มเติม 2

สำหรับทุกๆ $u,v \in (V \cup \Sigma)^*$, $u \overset{*}{\Rightarrow} v$ (หรือ $u \Rightarrow\Rightarrow v$ ในหนังสือบางเล่ม)

ถ้า $\exists u_1,u_2,\ldots,u_k \in (V \cup \Sigma)^*$, $k \geq 0$ แล้ว $u \Rightarrow u_1 \Rightarrow u_2 \Rightarrow \ldots \Rightarrow u_k \Rightarrow v$

ข้อกำหนดเพิ่มเติม 3

ภาษาของแกรมม่า $G = (V, \Sigma, R, S)$ เป็นเซตของ $L(G) = \{\omega \in \Sigma^* : S \overset{*}{\Rightarrow} \omega\}$

ข้อกำหนดเพิ่มเติม 4

กำหนดให้ภาษา L เป็น context-free language (CFL) ถ้า L นั้นมี CFG, G แล้ว L = L(G)

ตัวอย่างที่ 1

ยกตัวอย่าง context-free grammar ง่ายๆ เช่น $S \longrightarrow aSb \mid ab$

ซึ่งตัว | ซึ่งจะใช้เป็นทางเลือกว่า S จะแปลงไปเป็นตัวใดต่อ ซึ่งจะมีความหมายเหมือนกับ

$S \longrightarrow aSb$

$S \longrightarrow ab$

โดยจะมี Terminals คือ *a* และ *b*, และมีเพียงS ที่เป็น non-terminal ซึ่งแกรมม่า นี้จะสร้างภาษาของ $\{a^n b^n : n \geq 1\}$ ซึ่งไม่ได้เป็นภาษา regular

ตัว $\longrightarrow$ จะหมายถึง เป็นตัวว่างไม่มีตัวอักขระ . หากเปลี่ยนแกรมม่า ด้านบนเป็น

$S \longrightarrow aSb \mid \varepsilon$ เราจะได้แกรมม่า ของภาษา $\{a^n b^n : n \geq 0\}$ขึ้นมาแทน นี่คือความแตกต่างที่เกิดขึ้นจากการใส่ ε เข้าไปแทน

ตัวอย่างที่ 2

Context-free grammar สำหรับภาษาที่ใช้ชุดตัวอักษร {a,b} ซึ่งมีจำนวนของ a และ b ในจำนวนต่างๆคือ

$S \longrightarrow U \mid V$

$U \longrightarrow TaU \mid TaT$

$V \longrightarrow TbV \mid TbT$

$T \longrightarrow aTbT \mid bTaT \mid \varepsilon$

จะเห็นว่า Nonterminal T จะสร้างชุดของตัวอักษรที่มีจำนวนaและb เท่ากัน Nonterminal U จะสร้างชุดของตัวอักษรที่มีจำนวน a มากกว่า b และ Nonterminal V จะสร้างชุดของตัวอักษรที่มีจำนวน a น้อยกว่า b

BNF

BNF หรือ Backus-Naur Form ในตัวภาษาต่างๆนั้นจะประกอบไปด้วย Syntax และSemantic ซึ่งจะเป็นตัวกำหนดรูปแบบของภาษา โดยที่ Syntax จะเป็นรูปแบบหรือโครงสร้างของ expression, statement หรือหน่วยต่างๆของprogram และ Semantic จะเป็นตัวบ่งบอกความหมายของ expression, statement หรือหน่วยต่างๆของ program

BNF เป็น meta syntax ที่ใช้เพื่อแสดง context-free grammar มาให้เห็นได้ง่ายขึ้น BNF นั้นถูกใช้กันอย่างกว้างขวาง ไม่ว่าจะใช้เพื่อให้เห็นถึงแกรมม่า ของภาษาโปรแกรม ชุดของโครงสร้าง ตัว protocol หรือแม้แต่จะใช้เพื่อแสดงแกรมม่า ต่างๆของ natural language ก็ตาม

ประวัติของ BNF

BNF ถูกสร้างขึ้นโดย John Backus เพื่อใช้ในการ แสดงแกรมม่า ของภาษา ALGOL ในงาน World Computer Congress ซึ่งขึ้นจัดในปารีสปี 1959

ข้อกำหนดของ BNF จะเป็นเซตของกฎในการแปลงไวยากรณ์ซึ่งจะเขียนอยู่ในรูปของ

<symbol> ::= <expression with symbols>

โดยใน BNF จะมีทั้งแบบ Right-hand side(RHS) และ Left-hand side(LHS) ซึ่งจะประกอบไปด้วย

- Non-terminal: ยกตัวอย่างเช่น <symbol>, <expr> เป็นต้น
- Terminal : เป็นสัญลักษณ์หรือตัวอักษรต่างๆ เช่น (0,1,2,3,4,5,6,7,8,9)เป็นต้น
- Grammar : เป็นกฎต่างๆที่เป็นเซตจำกัดและไม่เป็นเซตว่าง ซึ่งจะเขียนอยู่ในรูป

<ident_list> ⟶ identifier|identifier, <ident_list>

<if_stmt> ⟶ if <logic_expr> then <stmt>

Derivation

Derivation เป็นการทำให้แกรมม่า นั้นดูง่ายขึ้นโดยการแตกแกรมม่า ออกมาเป็นหลายๆ บรรทัดย่อยตัวอย่างเช่น สมมติให้แกรมม่า เป็น

<Sum> ::= <Sum> + <Sum>

<Sum> ::= (<Sum>)

<Sum >::= 1

<Sum> ::= 0

ซึ่งจะสามารถเขียนออกมาใหม่ได้เป็น

<Sum> ⟶ <Sum> + <Sum >

⟶ (<Sum>) + <Sum>

⟶ (<Sum> + <Sum>) + <Sum>

⟶ (<Sum> + 1) + <Sum>

⟶ (<Sum> + 1) + 0

⟶ (0 + 1) + 0

โดยชุดของ symbol ใน derivation จะต้องอยู่ลักษณะดังนี้

- ในตัวประโยคจะต้องมีเฉพาะ terminal เท่านั้น
- leftmost derivation จะเป็นรูปแบบประโยคที่มี nonterminal อยู่ทางซ้ายสุดเสมอ
- ตัว derivation จะไม่เป็นทั้ง leftmost หรือ rightmost ก็ได้

ความกำกวมในแกรมม่า

ความกำกวมในแกรมม่า จะสามารถมองเห็นได้ชัดใน parse tree คือจะสามารถเขียน parse tree ออกมาจากแกรมม่า นั้นๆได้มากกว่า1รูปแบบ ยกตัวอย่างเช่น

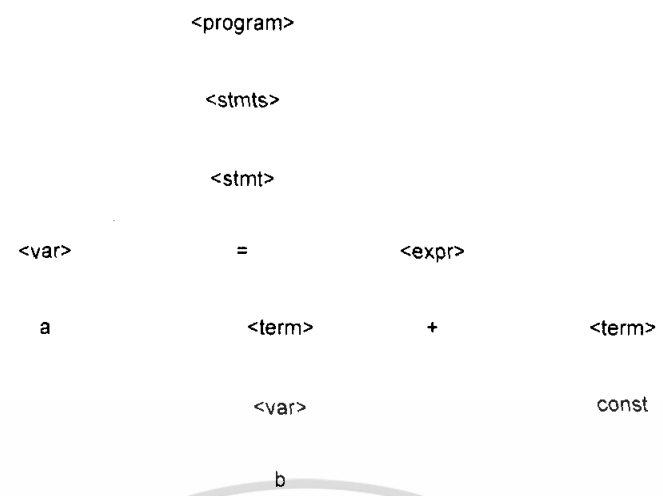
A ⟶ A + A | A - A | a

สามารถที่จะเขียนออกมาเป็น leftmost derivation ได้ 2 แบบ เพื่อใช้สำหรับชุด string a + a + a ดังนี้

A ⟶ A + A	A ⟶ A + A
⟶ a + A	⟶ A + A + A
⟶ a + A + A	⟶ a + A + A
⟶ a + a + A	⟶ a + a + A
⟶ a + a + a	⟶ a + a + a

Parse tree

Parse tree เป็นตัวที่จะช่วยให้มองเห็นโครงสร้างภาษานั้นชัดเจนมากขึ้น โดยที่จะเขียนโครงสร้างของภาษาออกมาในรูปของ tree โดยจะมีรูปแบบดังนี้



ภาพที่ 2.5 รูปแบบของต้นไม้

2.1.3 Syntax Analyzer

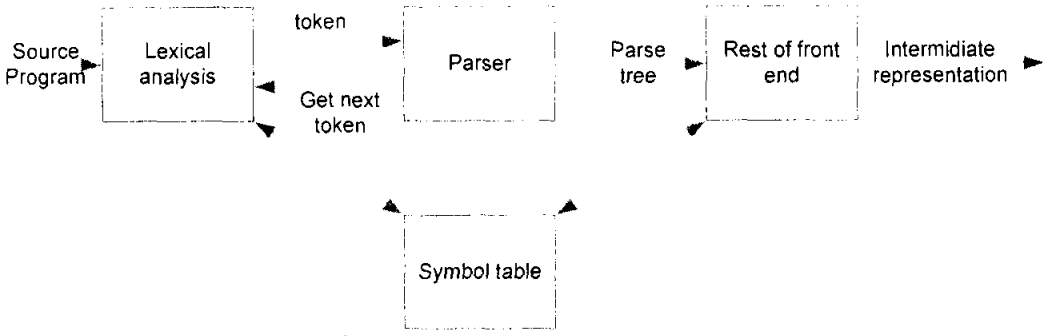
ในขั้นตอนของ Syntax Analyzer โดยรวมก็จะเป็นการตีความโครงสร้างของไวยากรณ์ตาม โทเคน ที่ได้มาจาก Lexical Analyzer ซึ่งผลลัพธ์ที่ได้จะอยู่ในรูปของ parse tree

ในการเขียนรูปแบบไวยากรณ์ของแต่ละภาษานั้น เราจะใช้ Context-free Grammar หรือ BNF (Backus Naur Form) เข้ามาช่วยในการเขียน ซึ่งการเขียนรูปแบบไวยากรณ์ในลักษณะนี้มีข้อดีหลายอย่างด้วยกัน คือ

- รูปแบบไวยากรณ์ที่อยู่ในลักษณะนี้สามารถอ่านทำความเข้าใจได้ง่าย
- ถ้าไวยากรณ์ที่เราสร้างขึ้นมาเขียนได้ดี จะสามารถสร้าง พาร์สเซอร์ ได้ง่าย และจะตรวจจับความผิดพลาดของประโยคที่อินพุต เข้ามาได้ดีอีกด้วย
- มีประโยชน์ช่วยในการสร้าง object code ในภายหลัง

พาร์สเซอร์ (Parser)

พาร์สเซอร์ มีหน้าที่คือ รับ โทเคน จาก Lexical Analyzer แล้วนำมาตรวจสอบว่า โทเคน หรือ string นั้นๆ เข้ากันได้กับไวยากรณ์ที่กำลังทำอยู่หรือเปล่า จากนั้นทำการสร้าง parse tree ออกมาดังรูป



ภาพที่ 2.6 ส่วนการทำงานของพาร์สเซอร์

ปัจจุบันวิธีการสร้าง parse tree มีอยู่ 2 วิธีก็คือ แบบ Top-Down กับแบบ Bottom-Up ซึ่งทั้งสองวิธีดังกล่าวจะใช้หลักการเดียวกันก็คือ อ่านข้อมูลอินพุต จาก ซ้ายไปขวา เพียงแต่จะต่างกันตรงที่วิธีการแบบ Top-Down จะทำการสร้าง parse tree จาก root node ส่วนวิธีการแบบ Bottom-Up จะทำการสร้าง parse tree จาก leave node

Parse Tree and Derivation

ในการสร้าง parse tree ขึ้นมาเราอาจมีวิธีการดังนี้

ภาพที่ 2.7 แสดงถึงการสร้าง parse tree ของอินพุต –(id + id)

จากภาพข้างต้น เป็นวิธีการง่ายๆ ในการสร้าง parse tree ของ –(id + id) ซึ่งประกอบไปด้วยกฎในการแปลงหลายๆกฎคือ

$E \longrightarrow - E$

$E \longrightarrow (E)$

$E \rightarrow E + E$

$E \rightarrow E * E$

$E \rightarrow id$

เราเรียก E ว่าเป็น Nonterminal Symbol ก็คือ เป็นสัญลักษณ์ที่สามารถทำการแปลงต่อไปได้ ส่วน id , (,) , - นั้น ไม่สามารถทำการแปลงต่อไปได้ซึ่งเราเรียกสัญลักษณ์จำพวกนี้ว่า Terminal Symbol

การทำพาร์สซิงแบ่งเป็นสองลักษณะคือ Top-down parsing และ Bottom-up parsing

Top-Down parsing

Top down parsing เป็นการแปลงไวยากรณ์ ให้กลายเป็นอินพุตสตริง (input string) โดยเริ่มทำการแปลงจากทางด้านซ้าย หรือเราอาจมองได้ว่า เป็นการสร้าง parse tree โดยเริ่มจาก root node ซึ่งในการทำ top down parsing นั้นมีทั้งแบบที่ทำแล้วย้อนขั้นตอนได้ (Recursive-descent parsing) กับแบบที่ไม่มีการย้อนขั้นตอน (Nonrecursive predictive parsing) ซึ่งหากไวยากรณ์มีการกำจัด ความกำกวมของไวยากรณ์ มาอย่างดีแล้ว การย้อนขั้นตอนก็จะไม่เกิด ซึ่งการประยุกต์ใช้งานจะแบ่งได้เป็นสองลักษณะคือ

- Recursive descent parser เป็น top-down พาร์สเซอร์ที่สร้างจากเซ็ตของโปรซีเยอร์ (procedures) ที่ยอมให้มีการเรียกซ้ำได้ (หรือไม่มีการเรียกซ้ำที่เทียบเท่ากัน) โดยส่วนใหญ่นั้นหนึ่งโปรซีเยอร์นั้นจะถูกนำไปใช้กับหนึ่ง production rules ของไวยากรณ์

- LL parser เป็น top-down พาร์สเซอร์สำหรับ context-free grammars โดยจะทำการ พาร์สจากซ้ายไปขวาและทำการสร้าง leftmost derivation ของรูปประโยค ซึ่งคลาสของไวยกรณ์ที่มีการพาร์สในลักษณะนี้รู้จักกันอีกชื่อว่า LLgrammars

Bottom-up parsing

Bottom-up parsing เป็นรูปแบบวิธีการที่ใช้ในการวิเคราะห์ข้อมูลที่เราไม่ทราบความสัมพันธ์ ซึ่งจะพยายามที่จะระบุส่วนพื้นฐานที่สุดของยูนิตก่อนและจึงทำส่วนที่อยู่สูงขึ้นต่อไปจากฐานที่ได้ โดยเราพยายามที่จะสร้างต้นไม้ขึ้นไปยังตัว symbol เริ่มต้นในวิทยาการคอมพิวเตอร์นั้น Bottom-up parsing นั้นรู้จักกันในชื่อว่า "shift-reduce parsing"

LR parser

LR parser นั้นเป็นพาร์สเซอร์สำหรับ context-free grammars ซึ่งจะอ่านอินพุตจากซ้ายไปขวาและทำการสร้าง Rightmost derivation ในส่วนของ LR(k) พาร์สเซอร์ก็ใช้ด้วยเช่นกัน ซึ่ง k จะอ้างถึงจำนวนของอินพุต symbol ที่จะอ่านล่วงหน้าที่จะใช้ในการสร้างตัวตัดสินใจในการพาร์สซึ่ง ส่วน

ใหญ่ค่า k จะเป็น 1 และบ่อยครั้งที่จะถูกละไว้ context-free grammar จะเรียกใช้ LR(k) ถ้ามี LR(k) พาร์สเซอร์ที่รองรับ

LR(0) คือพาร์สเซอร์ที่ไม่มีการมอง symbol ล่วงหน้า

SLR(1) คือพาร์สเซอร์แบบง่ายๆที่มีการมอง symbol ล่วงหน้าหนึ่งตัว

LALR(1) เป็นพาร์สเซอร์ชนิดพิเศษของ LR parser ที่สามารถจัดการกับ context-free grammar ได้มากกว่า SLR พาร์สเซอร์ ซึ่งพาร์สเซอร์ชนิดนี้เป็นนิยมที่สุดเพราะว่าพาร์สเซอร์ชนิดนี้แลกมาด้วยจำนวนในการจัดการกับไวยกรณ์ที่มากกว่าและต้องการขนาดของ parsing table พาร์สเซอร์ชนิดนี้ส่วนใหญ่ถูกสร้างโดยคอมไพลเลอร์ที่ใช้สร้างคอมไพลเลอร์ อย่างเช่น yacc หรือ CUP ภาษาที่ทั่วไปมากที่สุด แต่มีความซับซ้อนมากที่สุดในการนำไปใช้งานจริง

2.2 ตัวแปลภาษา (Interpreter)

อินเตอร์พรีเตอร์แบ่งได้เป็นสองแบบ คือ Iterative interpreter และ Recursive interpreter ซึ่ง Iterative interpreter เหมาะสำหรับประเภทคำสั่งที่ไม่ซับซ้อนหรือลักษณะที่เรียกว่า primitive ซึ่งมักจะเป็น real machine codes หรือ abstract machine codes หรือภาษาที่มีลักษณะง่ายๆ ส่วน Recursive interpreter เหมาะสำหรับภาษาที่มีความซับซ้อนของคำสั่ง เช่น การเรียกฟังก์ชั่นซ้อนๆ กัน หรือ การคำนวณหลายๆชั้น ซึ่งเหมาะสำหรับภาษาที่เป็นภาษาระดับสูง ซึ่งในโปรแกรมที่จะพัฒนาจะเป็นลักษณะ Recursive interpreter ซึ่งรูปแบบมีขั้นต้อนดังนี้

1. fetch and analyze the program
2. execute the program

ซึ่งขั้น fetch and analyze the program นั้นจะเป็นขั้นตอน Lexical analyzer, Syntax Analyzer ซึ่งจะทำการสร้างสิ่งที่เรียกว่า พาร์สทรี (Parse Tree) ขึ้นมาเพื่อนำไปแปลความหมายเป็นภาษาเครื่อง และนำไปใช้ในขั้นที่เรียกว่า Semantic analyzer แล้วทำการประมวลผลโปรแกรมแล้วส่งผลลัพธ์ออกมา

จากที่กล่าวมาจะเห็นว่าส่วนใหญ่จะหมายถึงการแปลในระดับที่แปลเป็น Intermediate Code มาแล้วซึ่งไปการประยุกต์ใช้งานเราสามารถที่จะทำให้ตัวแปลภาษาสามารถประมวลผลคำสั่งได้ทันทีหลังจากทำการวิเคราะห์รูปประโยคคำสั่งแล้ว ซึ่งจากขั้นตอนการทำคอมไพล์เลอร์จะมีเพียงแค่สามส่วนคือ Lexical Analysis, Syntax Analysis, Semantics Analyzer ซึ่งส่วนที่เพิ่มเข้ามาคือการ Execute โดยที่เราสามารถกระทำไปพร้อมๆกับการทำ Semantics Analyzer ก็ได้ส่วนที่เพิ่มเข้ามานี้เป็นการท่องไปในพาสทรีที่ได้จากขั้นตอนของ Syntax Analysis

2.2.1 การสร้างตัวแปลภาษา

ขั้นตอนการสร้างประกอบไปด้วย

1. ออกแบบส่วนของไวยากรณ์ (Grammar) ของภาษา ซึ่งจะแบ่งเป็นสองส่วนคือ ส่วนของกฎการวิเคราะห์คำ (Lexical rules) สำหรับตัววิเคราะห์คำ (Lexical Analysis หรือ Scanner) และส่วนของการวิเคราะห์รูปประโยค (Syntax Analysis หรือ Parser) ในส่วนของตัววิเคราะห์คำเราสามารถเขียนขึ้นมาเองโดยง่ายหรืออาจใช้เครื่องมือในการสร้างก็ได้ แต่ในการออกแบบก็ยังต้องใช้ความรู้พื้นฐานด้านตัวแปลภาษาอยู่ถึงจะทำให้โปรแกรมทำงานได้อย่างถูกต้อง ในส่วนของตัววิเคราะห์ประโยคนั้นขึ้นอยู่กับอัลกอริทึมที่จะใช้ซึ่งอัลกอริทึมที่มีประสิทธิภาพส่วนใหญ่นั้นเมื่อนำไปเขียนโปรแกรมโดยตรงตามไวยากรณ์ของภาษามักจะทำได้ยากและซับซ้อน ดังนั้นโดยปกติแล้วส่วนของการวิเคราะห์ไวยากรณ์นั่นจะไม่เขียนขึ้นเอง แต่จะใช้เครื่องมือในการสร้างแทนซึ่งรูปแบบของการเขียนจะเป็น Context-Free Grammar ซึ่งผลลัพธ์ส่วนท้ายที่ได้จากการวิเคราะห์ประโยคคือ พาสทรี (Parse Tree)

2. จากนั้นจะเป็นการออกแบบและสร้างส่วนประกอบสำหรับรองรับการทำงานของส่วนวิเคราะห์ไวยากรณ์และส่วนวิเคราะห์รูปประโยค ซึ่งประกอบด้วย ตารางข้อมูล (Symbol Table) และ ส่วน วิเคราะห์ความหมาย (Semantic Analyzer) โดยในส่วนนี้จะเป็นการทำการท่องไปในพาสทรีเพื่อทำการตรวจสอบความหมายความถูกต้องและประมวลผลเพื่อหาผลลัพธ์ในทันที ซึ่งส่วนนี้จะมีความซับซ้อนสูงมาก

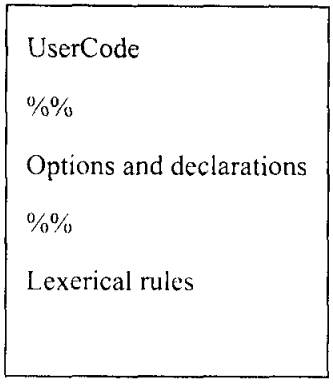
3. นอกนั้นจะเป็นการทำคุณสมบัติพิเศษเสริมความสามารถต่างๆของโปรแกรมซึ่งจะต้องออกแบบให้สามารถเพิ่มเข้าไปในตัวแปลภาษาให้ได้ง่ายโดยที่ไม่ต้องเข้าไปแก้ไขตัวแปลภาษาเลย

2.2.2 หลักการทำงานของเครื่องมือที่นำมาใช้ในการสร้างอินเตอร์พรีเตอร์

Lexical Analyzer Generator: JFlex

JFlex เป็นโปรแกรมที่ใช้ในการสร้างตัววิเคราะห์คำสำหรับภาษาจาวา ซึ่งเราทราบแล้วว่าตัววิเคราะห์คำเป็นตัวที่ใช้หาสิ่งที่เรียกว่า โทเคน สำหรับส่งไปให้ตัว parser จัดการวิเคราะห์รูปแบบประโยคซึ่งเหตุผลที่เราใช้เครื่องมือในการช่วยสร้างเพราะจะมีประสิทธิภาพดีกว่าการเขียนเข้าไปโดยตรงเพราะเครื่องมือนี้จะทำการวิเคราะห์รูปแบบของการอ่านเพื่อปรับปรุงประสิทธิภาพให้ด้วย ในการใช้งานนั้นข้อมูลเข้าของโปรแกรมนี้จะเป็นลักษณะของ Regular expression ซึ่งต้องเป็นไปตามกำหนดของโปรแกรมที่จะได้กล่าวต่อไป ส่วนผลลัพธ์จะเป็นซอร์สโค้ดโปรแกรมของแต่ละภาษาของเครื่องมือ สำหรับ JFlex ก็เป็น Lexical Analyzer สำหรับภาษาจาวานั่นเอง ซึ่งในขั้นแรกเราต้องทำการกำหนดรูปของการวิเคราะห์คำ(Lexical Specification) โดยเขียนเท็กซ์ไฟล์ในรูปตามที่ JFlex กำหนด แล้วนำเข้าสู่โปรแกรม JFlex เพื่อทำการประมวลผลแล้วสร้าง Lex ออกมาให้

ข้อกำหนดรูปแบบของการทำ Lexical ใน JFlex ประกอบไปด้วยสามส่วนซึ่งถูกแบ่งโดยบรรทัดที่เริ่มด้วยสัญลักษณ์ %% ดังนี้



ซึ่งสามารถใส่คอมเมนต์ได้ในรูปแบบคอมเมนต์บรรทัดเดียว "//" หรือแบบหลายบรรทัด "/*" ตามด้วย "*/"

User code

ส่วนแรกจะเป็นส่วนของ User code ที่จะถูกก็อปปี้เข้าไปเป็นส่วนหนึ่งของซอร์สไฟล์ที่จะเจนเนอเรทโดยตรง ในส่วนก่อนการประกาศคลาสของแสกนเนอร์หรือส่วนบนสุดของคลาสนั่นเอง ในส่วนนี้สามารถกำหนดเพกเกจของจาวาและการอิมพอร์ตต่างๆได้เช่นกันแต่ไม่เกี่ยวข้องกับการกำหนด Lexical Specifications แต่ไม่แนะนำให้ใส่โค้ดที่เป็นคลาสที่ช่วยในการทำงานสำคัญๆ หรือ เป็นส่วนที่ใช้ทำงานเฉพาะด้านเช่นคลาสโทเค้น ซึ่งควรจะแยกออกไปเป็นไฟล์ของมันเอง

Options and declarations

ส่วนที่สองของ Lexical specification ประกอบไปด้วยออปชั่นต่างๆสำหรับปรับแก้ตัว Lexer ที่จะสร้างขึ้น (ประกอบด้วย JFlex directives และ โค้ดภาษาจาวาที่จะแทรกลงไปในคลาสของ แสกนเนอร์) การประกาศ สถานะ (States) และการกำหนดมาโคร สำหรับใช้งานในส่วนที่สามคือ "Lexical rules" ของ Lexical specification ไฟล์ แต่ละ JFlexer directive ต้องเริ่มต้นด้วยเครื่องหมาย % ในแต่ละบรรทัด แต่ละไดเรคทีฟจะต้องมีพารามิเตอร์ตั้งแต่หนึ่งขึ้นไป ออปชั่นเหล่านี้จะส่งผลต่างๆถึง สแกนเนอร์คลาสที่จะสร้างขึ้นได้แก่

%class "classname"	ให้ JFlex สร้างคลาสเป็น "classname" และ เขียนโค้ดที่เจนเนอเรทขึ้นไว้ในไฟล์ "classname.java" หากไม่กำหนดจะถูกสร้างเป็น คลาสชื่อ "Yylex" และมีชื่อไฟล์เป็น "Yylex.java" โดยค่าเริ่มต้น
%implements "interface 1"[, "interface 2", ..]	ทำให้คลาสที่เจนเนอเรทขึ้นทำการอิมพลีเมนต์อินเตอร์เฟสที่กำหนดให้ ถ้าหากมีมากกว่าหนึ่ง %implements ทุกอินเตอร์เฟสที่กำหนดจะถูกอิมพลีเมนต์ด้วย

%extends "classname"	ทำให้คลาสที่สร้างขึ้นทำการสืบทอดหรือเป็นซับคลาสของคลาส "classname" ซึ่งต้องมีเพียงแค่คลาสเดียวหรือ %extends ตัวเดียวเท่านั้น
%public	ทำให้คลาสที่สร้างขึ้นเป็น public
%final	ทำให้คลาสที่สร้างขึ้นเป็นชนิด final คือไม่สามารถนำไปสืบทอดได้อีก
%abstract	ทำให้คลาสที่สร้างขึ้นเป็นชนิด abstract คือต้องนำไปสืบทอดเท่านั้นในการใช้งาน
%apiprivate	ทำให้ทุกๆเมธอดที่สร้างขึ้นและตัวแปรของคลาสเป็นรูปแบบ private ยกเว้นสำหรับคอนสครัคเตอร์ และ User code และถ้าหากมีการกำหนด %cup เมธอด next token
%{ ... %}	คำสั่งที่อยู่ระหว่าง %{ และ %} จะเป็นการคัดลอกคำสั่งไปยังคลาสที่สร้างขึ้น ซึ่งจะสามารถประกาศตัวแปรและฟังชั่นได้ และถ้ามีมากกว่าหนึ่งคลาส ก็จะนำตัวที่อยู่ระหว่าง %{ และ %} มาต่อกัน
• %init{ ... %init}	คำสั่งที่อยู่ระหว่าง %{ และ %} จะเป็นการคัดลอกคำสั่งไปยังคอนสครัคเตอร์ ของคลาสที่สร้างขึ้น ตัวแปรที่ถูกประกาศภายในจะสามารถนำไปใช้ได้และถ้ามีมากกว่า 1 ตัวก็ได้

%cup	คำสั่ง %cup สามารถใช้ในโหมด CUP compatibility เหมือนกับคำสั่งอื่นๆซึ่งเราไม่ต้องกำหนดเองทั้งหมด ดังนี้
	1. %implements java_cup.runtime.Scanner
	2. %function next_token
	3. %type java_cup.runtime.Symbol
	4. %eofval{
	return new java_cup.runtime.Symbol(<CUPSYM>.EOF);
	%eofval}
	5. %eofclose
	ค่าเริ่มต้นของ <CUPSYM> คือ sym และสามารถเปลี่ยนได้ด้วยคำสั่ง %cupsym

%cupsym "classname"	เป็นการตั้งชื่อให้กับคลาส CUP generate จะประกอบไปด้วยชื่อของ token ค่าพื้นฐานคือ sym คำสั่งนี้ควรอยู่ก่อนคำสั่ง %cup
%ignorecase %caseless	คำสั่งนี้จะทำให้ JFlex นั้นจะดูและอักขระและตัวอักษรพิเศษ เช่น ตัวอักษรใหญ่หรือเล็ก โดยระบบนี้จะง่ายต่อภาษาที่ไม่สนใจในขนาดของตัวอักษร ตัวอักษร "break" ถ้าเรามาดูภาษาจะได้

	([bB][rR][eE][aA][kK]) คำสั่ง %caseless จะไม่ส่งผลกระทบต่อการค้นหาหรืออักขระ แค่ทำให้ a สามารถจับคู่กับ A ได้ อักขระตัวใหญ่หรือตัวเล็กนั้นถูกนิยามโดย Unicode และตัดสินใจด้วยเมธอด Character.toUpperCase และ Character.toLowerCase ของ Java และ JFlex

ตารางที่ 2.4 แสดงออฟชันชนิดต่างๆสำหรับ JFlex

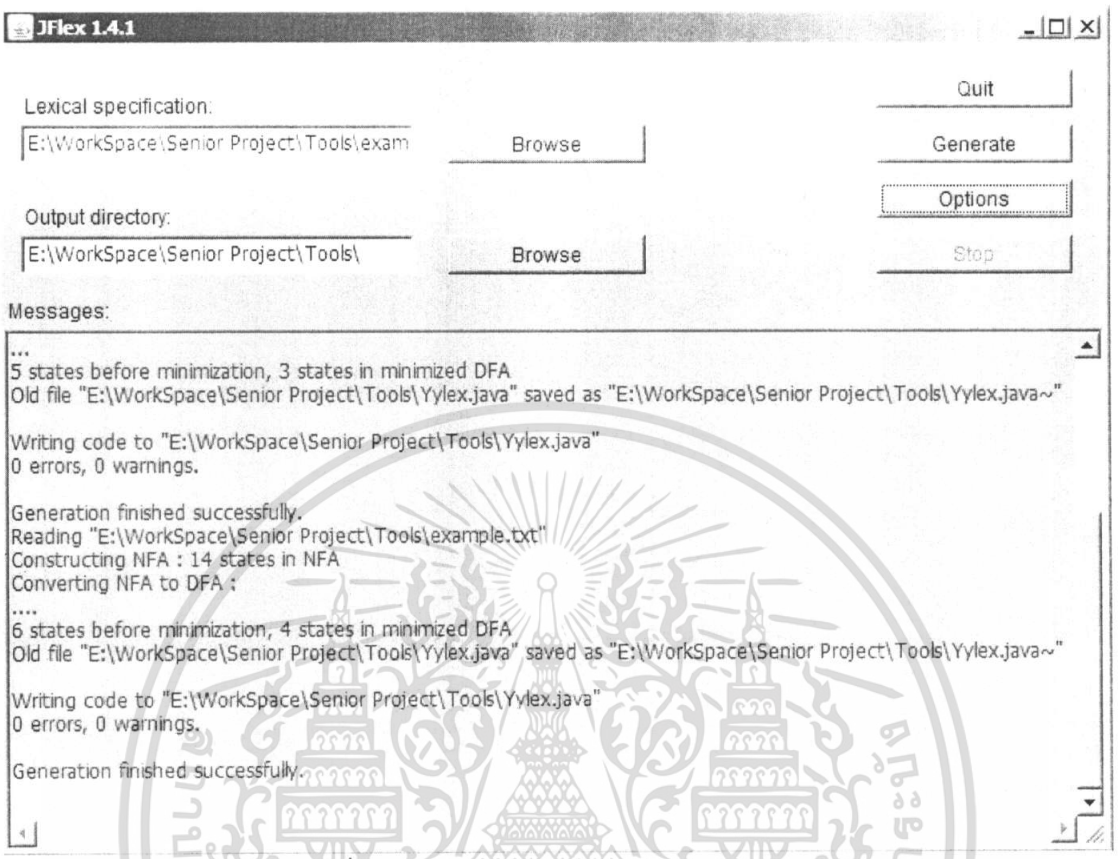
Lexical rules

ในการกำหนดกฎของการวิเคราะห์คำใน JFLex จะประกอบไปด้วยเซ็ตของ regular expression และส่วนของการทำงานซึ่งเขียนเป็นโค้ดภาษาจาวา ซึ่งจะทำงานหลังจากที่ตัวสแกนเนอร์สามารถจับคู่คำที่สัมพันธ์กับ regular expression ที่เรากำหนดขึ้นมาได้ ในส่วนของ Lexical rules รูปแบบกฎของส่วนของการวิเคราะห์คำจะถูกกำหนดโดย BNF แกรมม่า (โดยที่ terminal symbol จะถูกครอบด้วยเครื่องหมาย 'quotes') ซึ่งเราสามารถใช้คุณสมบัติต่างๆ ของ regular expression และ BNF แกรมม่าเข้ามาช่วยได้

ตัวอย่างเช่นหากเราต้องการตัวอ่านเลขฐานสอง โดยที่ถ้ามีการเว้นวรรคก็จะส่งชนิดและค่าของเลขออกไป จะได้ Lexical rules ดังนี้

```
%%

%public                 //กำหนดให้คลาสที่ถูกสร้างขึ้นเป็น public
%unicode //รองรับการทำงานตัวอักษีแบบ Unicode
/* User Code เพื่อกำหนดคลาสที่ชื่อว่า Token ซึ่งคลาสนี้ถ้าเราไม่ระบุชนิดของการคืนค่า
   JFLexer จะกำหนดเป็นคลาสที่ชื่อว่า Token ให้ซึ่งเราต้องทำการเขียนโค้ดขึ้นมารองรับส่วนนี้ */

%{
 public class Yytoken {
  public int type;
  public String value;
  Yytoken (int t, String v) {
   type = t;
   value = v;
  }
 }
%}
/* ใช้หลักการพื้นฐานของ regular expression ในการกำหนด */
/* กำหนดมาโครสำหรับตัวเลขฐานสอง */
Binary = 0 | 1 ( 0 | 1 )*          //หมายความว่า มีเลข 1 ตัวเดียว หรือ เลข 1 ตามด้วย 0 หรือ 1 กี่ตัวก็ได้
/* กำหนดมาโครสำหรับกำหนดเซ็ตของตัวอักษร */
Char = [a-z]                       //หมายความว่าจะมีตัวอักขระตั้งแต่ a ถึง z เพียงตัวใดตัวหนึ่งเท่านั้น
%%
/* Keywords */
/* หากพบตัวเลขฐานสองให้ทำการคืนเป็นออปเจกต์ของคลาส Yytoken
   เก็บค่า 1 และเลขฐานสองที่อ่านได้ */
{Binary}                { return new Yytoken (1, yytext()); }
/* หากพบตัวอักขระให้ทำการคืนเป็นออปเจกต์ของคลาส Yytoken
  เก็บค่า 2 และเลขฐานสองที่อ่านได้ */
{Char}                  { return new Yytoken (2, yytext()); }

"\ "                    { // ไม่ต้องทำอะไรหากเจอ white space }

/* หากสิ้นสุดไฟล์ให้คืนค่า null */
<<EOF>>                 { return null; }
```

จากนั้นเราก็ทำการเปิดโปรแกรม JFLex แล้วทำการกดปุ่ม Generate จะได้ดังรูป

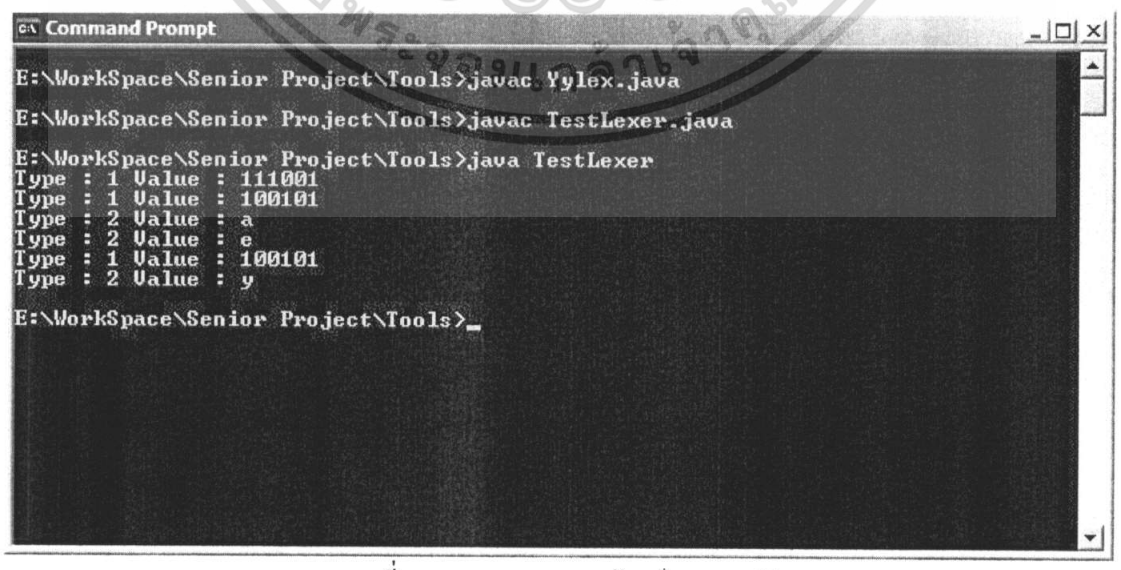


ภาพที่ 2.8 ตัวอย่างหน้าจอของโปรแกรม JFLex

หลังจากนั้นโปรแกรมจะสร้างไฟล์ที่ชื่อว่า YyLex.java ขึ้นมาให้ จากนั้นเราก็สามารถนำไปใช้งานได้โดยการเขียนโค้ดดังนี้

```
import java.io.*;


public class  TestLexerer {
        public static void main(String[] args) throws IOException {
                //สายอักขระสำหรับอินพุต
                String input = "111001 100101a e100101y";
                //ออปเจกต์สำหรับอ่านข้อมูล
                Reader reader = new StringReader(input);
                //สร้างตัว Lexerer จากคลาส YyLexer ที่ได้สร้างขึ้น
                YyLexer scanner = new YyLexer(reader);
                YyLexer.Yytoken token = null;
                // ทำการสแกนไปจนกว่าจะสิ้นสุดสายอักขระ
                while ( (token=scanner.yyLexer()) != null) {
                        //แสดงหมายเลขชนิดและค่าที่อ่านได้
                        System.out.println("Type : " + token.type +
                                " Value : " + token.value);
                }
        }
```

หลังจากนั้นเมื่อนำโปรแกรมที่ได้มาคอมไพล์และรันจะได้ดังรูป



ภาพที่ 2.9 แสดงการคอมไพล์และการรัน

จากโปรแกรมและผลลัพธ์แสดงให้เห็นว่าจากโปรแกรมที่ JFLex สร้างขึ้นมาตามข้อกำหนดของเราซึ่งต้องการแบ่งเลขฐานสองและตัวอักขระออกจากกันซึ่งอินพุตที่เราใส่เข้าไปเป็น "111001 100101a e100101y" จากนั้นเมื่อรันโปรแกรมจะเห็นได้ว่าเลขฐานสองและตัวอักขระถูกแบ่งออกมา

CUP (Constructor of Useful Parsers)

CUP คือโปรแกรมสำหรับสร้าง พาสเซอร์แบบ LALR จากรูปแบบข้อกำหนดอย่างง่าย ซึ่งทำงานเช่นเดียวกับโปรแกรม YACC ที่นิยมใช้กันสำหรับภาษาซี รวมไปถึงคุณสมบัติต่างๆด้วย ส่วน CUP นั้นเขียนด้วยภาษาจาวาซึ่งจะเน้นในการระบุข้อกำหนดโดยอิง
ภาษาจาวารวมไปถึงโค้ดที่จะสร้างขึ้นตามข้อกำหนด

การใช้ CUP ต้องสร้างข้อกำหนดบนพื้นฐานของไวยากรณ์ของภาษาสำหรับพาร์สเซอร์ที่ต้องการสร้างขึ้น ซึ่งต้องอาศัยตัว scanner ในการแยกคำแต่ละคำให้เป็นโทเค้นที่มีความหมายอย่างเช่นคีย์เวิร์ด ตัวเลข มาทำงานร่วมด้วย

ตัวอย่างง่ายๆที่จะแสดงให้เห็นการออกแบบและการสร้างดังนี้
สมมติว่าเราต้องการสร้างตัวคำนวณตัวเลขจำนวนเต็มขึ้นมา โดยที่รับอินพุตมาจากคีย์บอร์ดและจะถูกแบ่งโดยเครื่องหมาย ";" ทำการคำนวณหาผลลัพธ์ออกทางหน้าจอ จะได้ไวยากรณ์ของภาษาดังนี้

```
expr_list ::= expr_list expr_part | expr_part
expr_part ::= expr ';'
expr      ::= expr '+' expr | expr '-' expr | expr '*' expr
            | expr '/' expr | expr '%' expr | '(' expr ')'
            | '-' expr | number
```

เพื่อที่จะสร้างพาร์สเซอร์บนพื้นฐานของแกรมม่าดังกล่าว ในขั้นแรกเราจะตองระบุชื่อและเซ็ตของ terminal symbol ที่จะปรากฏบนอินพุต และเซ็ตของ non-terminal symbols ในกรณีนี้จะได้ว่ามี non-terminal symbol เป็น

expr_list, expr_part และ expr

ส่วน terminal symbol จะได้ดังนี้

SEMI, PLUS, MINUS, TIMES, DIVIDE, MOD, NUMBER, LPAREN, และ RPAREN

จากการออกแบบข้างต้นเราจะรู้ว่าไวยากรณ์ดังกล่าวนั้นยังมีปัญหา ซึ่งก็คือเกิดความกำกวมของไวยากรณ์ ทำให้เวลาทำการแตกออกเป็นพาสทรีนั้นทำได้สองแบบสองความหมาย ซึ่งใน CUP จะให้เรากำหนด precedence และ associativities สำหรับ terminal ดังนั้นเราจึงสามารถเขียนไวยากรณ์แบบกำกวมได้ ในการกำหนดดูได้จากตัวอย่าง ซึ่งเราจะได้ข้อกำหนดเบื้องต้นของไวยากรณ์ที่เรากำหนดดังนี้

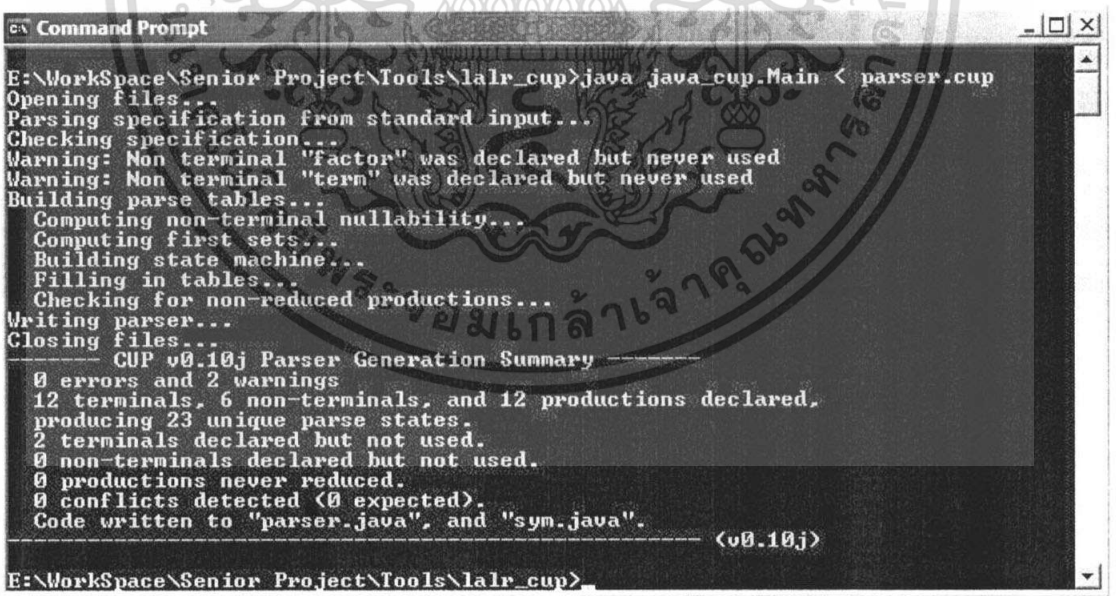
```
// ข้อกำหนดของ CUP สำหรับการคำนวณทางคณิตศาสตร์อย่างง่าย (ยังไม่ระบุการทำงาน)
import java_cup.runtime.*;
/* กำหนดส่วนเริ่มต้นของการใช้งานและการทำงานต่างๆ. */
init   with          {: scanner.init();                :};  //กำหนดเมธอดที่ทำงานครั้งแรก
scan with            {: return getScanner.next_token(); :};      //กำหนดเมธอดที่ทำการสแกนคำ
/* Terminals (โทเค้นที่คืนค่าออกมาจากสแกนเนอร์) */
terminal          SEMI, PLUS, MINUS, TIMES, DIVIDE, MOD;
terminal          UMINUS, LPAREN, RPAREN;
terminal Integer NUMBER;
/* Non terminals */
non terminal          expr_list, expr_part;
non terminal Integer    expr, term, factor;
/* Precedences (กำหนดลำดับความสำคัญ โดยที่สำคัญมากกว่าอยู่ด้านล่าง) */
precedence left PLUS, MINUS;
precedence left TIMES, DIVIDE, MOD;
precedence left UMINUS;
/* ไวยากรณ์ของภาษา */
expr_list ::= expr_list expr_part |
              expr_part
          ;
expr_part ::= expr SEMI
          ;
expr      ::= expr PLUS expr
          | expr MINUS expr
          | expr TIMES expr
          | expr DIVIDE expr
          | expr MOD expr
          | MINUS expr %prec UMINUS //กำหนด precedences ในส่วนไวยากรณ์ผ่านตัวอ้างอิง
          | LPAREN expr RPAREN
          | NUMBER
          ;
```

เราจะดูในแต่ละส่วนของข้อกำหนดต่างๆซึ่งรายละเอียดจะเอาไว้ทีหลัง หากมองเร็วๆเราจะเห็นว่าส่วนแรกจะเป็นข้อกำหนดพื้นฐานต่างๆทั่วๆไปในการสร้างตัวพาร์สเซอร์ขึ้นมาและส่วนรันไทม์โค้ด ในกรณีนี้เรากำหนดให้อิมพอร์ตคลาส java_cup.runtime ซึ่งเป็นส่วนที่สำคัญ จากนั้นก็เป็นการระบุส่วนเริ่มต้นโค้ดการทำงาน และโค้ดสำหรับการเรียกใช้สแกนเนอร์เพื่อรับข้อมูลถัดๆไป ส่วนที่สองนั้นเป็นส่วนที่ทำการประกาศ terminals, non-terminals และความสัมพันธ์ในแต่ละคลาส ในกรณีนี้ terminals สามารถที่จะประกาศชนิดของข้อมูลเป็น Integer หรือจะไม่ประกาศก็ได้ รายละเอียดของชนิดข้อมูลนั้นจะเป็นตัวบอกชนิดของค่าที่อยู่ใน terminals และ non-terminals เหล่านั้น ซึ่งถ้าไม่ได้ประกาศเอาไว้ก็หมายความว่า terminals หรือ non-terminals นั้นไม่ได้เก็บค่าไว้ ซึ่งก็คือการที่ไม่มีชนิดข้อมูลนั้นก็แสดงให้เห็นว่า terminals หรือ non-terminals นั้นหยุดลงที่การไม่มีค่า ส่วนที่สามจะเป็นรายละเอียดของ precedences และ associativities ของ terminals โดย terminals ที่ถูกกำหนด precedences ท้ายสุดนั้นแสดงว่า terminals นั้นมี precedences ที่สูงที่สุด ในส่วนสุดท้ายนั้นจะเป็นส่วนของไวยากรณ์

ในการสร้างพาร์สเซอร์จากข้อกำหนดนี้เราจะใช้ CUP generator สมมติว่าเราเก็บข้อกำหนดไว้ในไฟล์ชื่อว่า parser.cup ดังนั้นเราจะเรียกใช้งาน CUP โดยใช้คำสั่งดังนี้:

```
java java_cup.Main < parser.cup
```

ซึ่งจะได้ผลลัพธ์ดังรูป



ภาพที่ 2.10 แสดงการทำงานของ CUP

ในกรณีนี้ระบบจะสร้างไฟล์จาวาขึ้นมาสองตัวคือ sym.java และ parser.java เพื่อเก็บส่วนของตัวสร้างพาร์สเซอร์ ซึ่งก็ตามที่คาดไว้ไฟล์สองตัวนี้จะทำการเก็บคำสั่งสำหรับคลาส sym และ parser ใน sym class นั้นจะประกอบไปด้วยลำดับสำหรับคำสั่งที่คงที่สำหรับแต่ละ terminal symbol ซึ่งนี่เป็นรูปแบบในการอ้างถึงไปยัง symbol ของตัว scanner (ตามตัวอย่างโค้ดเช่น "return new Symbol(sym.SEMI);") ส่วน parser class นั้นจะทำการ implement parser ด้วยตัวเอง

จากข้อกำหนดข้างต้นในระหว่างที่ทำการสร้างพาร์สเซอร์ตัวเต็มนั้นไม่ได้มีการทำ semantic actions ใดๆ ซึ่งมันบอกเพียงแค่ว่าการพาร์สนั้นผ่านหรือไม่ผ่านเท่านั้น เพื่อที่จะทำการคำนวณและพิมพ์ค่าของแต่ละ expression นั้น เราจำเป็นต้องเพิ่มโค้ดเข้าไปตามจุดต่างๆเอง ใน CUP นั้น action จะถูกเก็บอยู่ใน code string ที่ ถูกครอบด้วยข้อจำกัดในรูป {: และ :} โดยทั่วไปนั้นระบบจะทำการเก็บบันทึกทุกตัวอักษรไว้ด้วยข้อจำกัด แต่ไม่ได้พยายามที่จะตรวจสอบว่ามีการเก็บ java code ต่อไปเราจะแสดงถึงข้อกำหนดของ CUP ที่มีความสมบูรณ์มากยิ่งขึ้น ดังตัวอย่างด้านล่าง

```
// ข้อกำหนดของ CUP สำหรับการคำนวณทางคณิตศาสตร์อย่างง่าย (ระบุการทำงาน)

import java_cup.runtime.*;

/* กำหนดส่วนเริ่มต้นของการใช้งานและการทำงานต่างๆ.  */
init   with {: scanner.init();                :};        //กำหนดเมธอดที่ทำงานครั้งแรก
scan with {: return getScanner.next_token(); :};    //กำหนดเมธอดที่ทำการสแกนค่า

/* Terminals (โทเค้นที่คืนค่าออกมาจากสแกนเนอร์) */
terminal          SEMI, PLUS, MINUS, TIMES, DIVIDE, MOD;
terminal          UMINUS, LPAREN, RPAREN;
terminal Integer NUMBER;

/* Non terminals */
non terminal             expr_list, expr_part;
non terminal Integer    expr, term, factor;

/* Precedences (กำหนดลำดับความสำคัญ โดยที่สำคัญมากกว่าอยู่ด้านล่าง) */
precedence left PLUS, MINUS;
precedence left TIMES, DIVIDE, MOD;
precedence left UMINUS;

/* ไวยากรณ์ของภาษา */
expr_list ::= expr_list expr_part
            |
            expr_part
            ;
expr_part ::= expr:e
            {: System.out.println("= " + e); :}
            SEMI
            ;
expr ::=  expr:e1 PLUS expr:e2          // e1 + e2
            {: RESULT = new Integer(e1.intValue() + e2.intValue()); :}
            |
            expr:e1 MINUS expr:e2          // e1 – e2
            {: RESULT = new Integer(e1.intValue() - e2.intValue()); :}
            |
            expr:e1 TIMES expr:e2          // e1 * e2
            {: RESULT = new Integer(e1.intValue() * e2.intValue()); :}
            |
            expr:e1 DIVIDE expr:e2          // e1 / e2
            {: RESULT = new Integer(e1.intValue() / e2.intValue()); :}
            |
            expr:e1 MOD expr:e2          // e1 % e2
            {: RESULT = new Integer(e1.intValue() % e2.intValue()); :}
            |
            NUMBER:n
            {: RESULT = n; :}
            |
            MINUS expr:e                    // -e
            {: RESULT = new Integer(0 - e.intValue()); :}  %prec UMINUS
            |
            LPAREN expr:e RPAREN          // ( e )
            {: RESULT = e; :}
            ;
```

จะเห็นว่ามีหลายอย่างที่เปลี่ยนไป ที่สำคัญที่สุดคือโค้ดที่ถูกทำงานแต่ละจุดในพาร์สจะถูกรวมเข้าไปในตัวจำกัดโค้ดสตริง {: และ :} นอกจากนั้นป้ายชื่อ (labels) ได้ถูกวางไว้บน Symbols ต่างๆใน Production ทางฝั่งขวา ตัวอย่างเช่น

```
expr:e1 PLUS expr:e2
  {: RESULT = new Integer (e1.intValue () + e2.intValue ()); :}
```

ใน non-terminals expr ตัวแรกนั้นได้ถูกกำหนดชื่อด้วย e1 และตัวถัดไปกำหนดด้วย e2 ส่วนทางฝั่งซ้ายของแต่ละ productions จะถูกกำหนดชื่ออย่างเป็นนัยคือ RESULT อยู่เสมอ

แต่ละ Symbol ที่ปรากฏอยู่ใน production จะเป็นตัวอธิบายใน runtime โดยออบเจ็กต์ของ symbol type นั้นบน parse stack ป้ายชื่อจะอ้างถึงตัวแปรในออบเจ็กต์เหล่านั้น ใน expression expr:e1 PLUS expr:e2 นั้น e1 และ e2 นั้นจะอ้างถึงออบเจ็กต์ของตัวแปรชนิด Integer ซึ่งออบเจ็กต์นี้อยู่ใน value fields ของออบเจ็กต์ type Symbol ซึ่งเป็นตัวแทนของ non-terminals บน parse stack

RESULT นั้นเป็น Integer เช่นเดียวกันเนื่องจากว่า non-terminal expr นั้นได้ถูกประกาศไว้เป็นชนิด Integer ซึ่งออบเจ็กต์นี้จึงกลายมาเป็นค่าของตัวแปรใน Symbol ออบเจ็กต์ตัวใหม่
ในชื่อแต่ละตัวนั้นได้ถูกประกาศการเข้าถึงของตัวแปรแก่ผู้ใช้งานไว้มากกว่า 2 ตัว ชื่อของค่าทางซ้ายและขวาจะถูกส่งผ่านไปยังโค้ดสตริงซึ่งนั่นก็คือผู้ใช้งานสามารถรู้ได้ว่า terminals หรือ non-terminals ทางซ้ายและขวาใดอยู่ในอินพุตสตรีม(input stream) ชื่อของตัวแปรเหล่านี้จะเป็นชื่อที่เพิ่ม left หรือ right เข้าไป ตัวอย่างเช่น ให้ Production ทางฝั่งขวาเป็น expr:e1 PLUS expr:e2 ผู้ใช้งานไม่ได้เพียงแต่เข้าถึงตัวแปรโดยใช้ e1 และ e2 เท่านั้นแต่สามารถใช้ e1left, e1right, e2left and e2right ได้เช่นกัน ซึ่งตัวแปรเหล่านี้เป็นประเภท int

ลำดับต่อไปในการสร้างพาร์สเซอร์คือการสร้างสแกนเนอร์ (ที่รู้จักกันในชื่อ Lexical analyzer หรือเรียกสั้นๆว่า Lexer) ซึ่งมีหน้าที่รับผิดชอบในการ พิจารณา characters, ทำการถอดช่องว่าง และคอมเมนท์ออก, แสดงให้เห็นว่า terminal symbol นี้มาจากไวยกรณ์กลุ่มไหนของตัวแทน character, หลังจากนั้นจะทำการคืนค่าตัวแทนออบเจ็กต์ของ symbol นั้นไปยังพาร์สเซอร์ และ Terminal จะทำการรับค่าคืนด้วยการเรียกไปยังฟังก์ชันของสแกนเนอร์ ดังตัวอย่าง เช่น พาร์สเซอร์จะเรียก scanner.next_token() สแกนเนอร์จะต้องทำการคืนค่าออบเจ็กต์ประเภทข้อมูล java_cup.runtime.Symbol ซึ่งประเภทของตัวแปรนี้ต่างไปจาก CUP's java_cup.runtime.symbol เวอร์ชันก่อนๆมาก ซึ่ง Symbol ออบเจ็กต์นี้จะประกอบด้วยค่าinstance ตัวแปรของออบเจ็กต์ประเภทข้อมูลซึ่งควรจะถูกกำหนดโดย Lexer ตัวนี้นี้จะอ้างถึงค่าของ symbol นั้นและประเภทข้อมูลของออบเจ็กต์ในค่านั้นควรจะเหมือนกับที่ประกาศไว้ในส่วนของ terminals และ non-terminals ในตัวอย่างข้างต้นถ้า Lexer ต้องการที่จะส่ง NUMBER โทเค้น มันก็ควรจะสร้าง symbol

ที่มีค่าของตัวแปรในออบเจ็กต์ของประเภทข้อมูลเป็น Integer ออบเจ็กต์ของ Symbol ที่สอดคล้องกับ terminals และ non-terminals ที่ไม่มีค่าจะมีค่า null ในฟิลด์

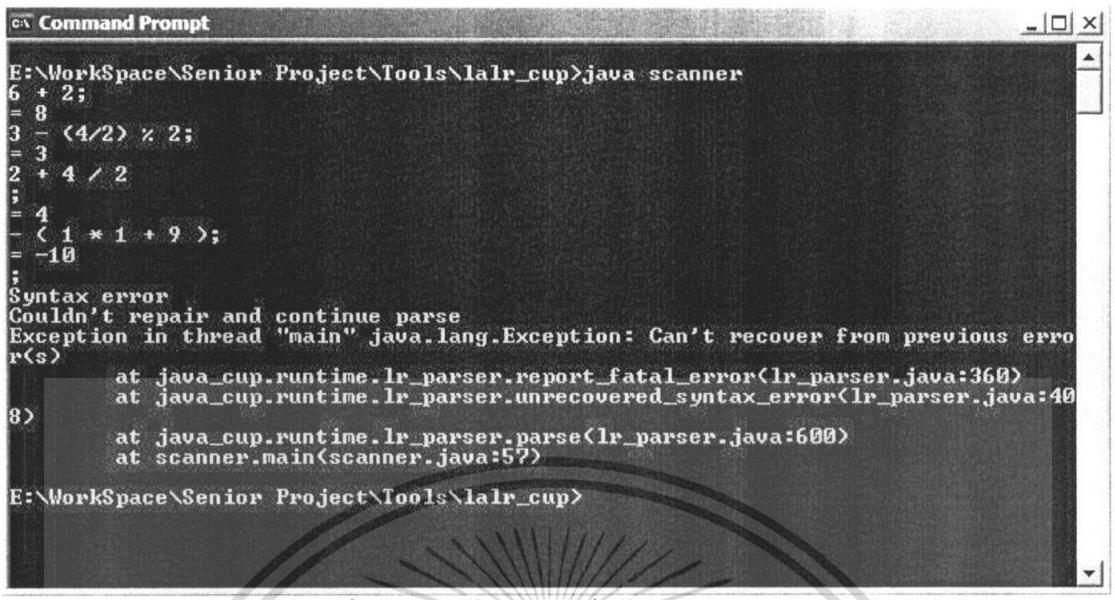
โค้ดที่อยู่ใน init พร้อมกับข้อกำหนดจะถูก executed ก่อนที่โทเคนใดๆจะร้องขอโทเคนแต่ละตัวจะร้องขอถ้าโค้ดที่ถูกสแกนพบนั้นตรงกับข้อกำหนด ซึ่งมาถึงจุดนี้ความถูกต้องของสแกนเนอร์นั้นขึ้นอยู่กับผู้เขียนเอง อย่างไรก็ตามต้องจำเอาไว้ว่าในการเรียกฟังก์ชันสแกนเนอร์แต่ละครั้งควรจะคืนค่า instance ของ java_cup.runtime.Symbol (หรือ subclass) ตัวใหม่ ซึ่งออบเจ็กต์เหล่านี้จะเป็นตัวบันทึกเหตุการณ์พร้อมกับข้อมูลของพาร์สเซอร์ลงไปยัง stack การนำออบเจ็กต์มาใช้ใหม่นั้นจะส่งผลให้ตัวบันทึกเหตุการณ์มันไปขัดกันเอง ใน CUP 0.10j symbol reuse ควรจะถูกตรวจพบและถ้ามันเกิดขึ้น พาร์สเซอร์ก็จะแจ้งข้อผิดพลาดเพื่อให้เราทำการแก้ไขสแกนเนอร์

ตัวอย่างของการเขียนสแกนเนอร์ (scanner.java)

```
// Simple Example Scanner Class
import java_cup.runtime.*;
public class scanner implements java_cup.runtime.Scanner {
  /* single lookahead character */
  protected static int next_char;

  /* advance input by one character */
  protected static void advance()   throws java.io.IOException
    { next_char = System.in.read(); }

  /* initialize the scanner */
  public static void init()   throws java.io.IOException
    { advance(); }
```

```
/* recognize and return the next complete token */
public Symbol next_token()   throws java.io.IOException    {
   for (;;) { switch (next_char)  {
            case '0': case '1': case '2': case '3': case '4':
            case '5': case '6': case '7': case '8': case '9':
             /* parse a decimal integer */
             int i_val = 0;
             do {
              i_val = i_val * 10 + (next_char - '0'); advance();
             } while (next_char >= '0' && next_char <= '9');
            return new Symbol(sym.NUMBER, new Integer(i_val));
            case ';': advance(); return new Symbol(sym.SEMI);
            case '+': advance(); return new Symbol(sym.PLUS);
            case '-': advance(); return new Symbol(sym.MINUS);
            case '*': advance(); return new Symbol(sym.TIMES);
            case '/': advance(); return new Symbol(sym.DIVIDE);
            case '%': advance(); return new Symbol(sym.MOD);
            case '(': advance(); return new Symbol(sym.LPAREN);
            case ')': advance(); return new Symbol(sym.RPAREN);
            case -1: return new Symbol(sym.EOF);
            default:   /* in this simple scanner we just ignore everything else */
              advance();
            break;
           }
  } }
        public static void main (String args[]) throws Exception {
                scanner s = new scanner(); // สร้างออบเจกต์สแกนเนอร์
                parser p = new parser(s);    // สร้างออบเจกต์พาร์สเซอร์
                p.parse();                          // เริ่มทำการรับอินพุตและทำการประมวลผล
        }
};
```

จากนั้นทำการคอมไพล์ไฟล์ parser.java และ scanner.java และรันจะได้ผลดังรูป



ภาพที่ 2.11 แสดงผลลัพธ์ที่ได้หลังการคอมไพล์

จากตัวอย่างจะเห็นว่าเราสามารถทำการป้อนตัวเลขและเครื่องหมายต่างเพื่อทำการคำนวณค่าออกมาได้โดยเมื่อต้องการคำนวณก็พิมพ์ ";" แล้วกด enter ผลลัพธ์ก็จะปรากฏขึ้นมา หากเราป้อนผิดตามข้อกำหนดของไวยากรณ์ก็จะแสดงความผิดพลาดและหยุดทำงาน

2.3 พื้นฐานหลักการของคอมพิวเตอร์กราฟิกส์ OpenGL และ JOGL

การสร้างภาพสามมิติโดยใช้ OpenGL

OpenGL เป็นซอร์ฟแวร์ไลบรารีที่ใช้ติดต่อกับฮาร์ดแวร์แสดงผล โดยมีคำสั่งพื้นฐานประมาณ 120 คำสั่ง ที่สามารถใช้กำหนดคุณลักษณะ และควบคุมการทำงานของแอ๊พพลิเคชันสามมิติ ผู้พัฒนาโปรแกรมสามารถใช้ไลบรารี OpenGL ได้โดยไม่มีค่าลิขสิทธิ์

OpenGL ถูกออกแบบมาโดยไม่ยึดติดกับระบบ สามารถทำงานได้บนทุกๆแพล็ตฟอร์ม (Portability) เพื่อที่จะบรรลุเป้าหมายนี้ OpenGL จะไม่มีคำสั่งที่จัดการกับระบบปฏิบัติการเลย อีกทั้งยังไม่มีคำสั่งเพื่อรับ อินพุทจากผู้ใช้อีกด้วย หน้าที่ทั้งสองอย่างนี้จึงตกอยู่กับผู้เขียนโปรแกรม อย่างไรก็ตามยังมีซอร์ฟแวร์ เพิ่มเติมที่ช่วยจัดการงานทั้งสองนี้หากพัฒนาโปรแกรมบนระบบปฏิบัติการแบบ Windows (ดู GLUT: OpenGL Utility Toolkit) นอกจากนี้ OpenGL ยังไม่มีคำสั่งระดับสูงที่จะใช้วาดวัตถุสามมิติแบบซับซ้อน อย่างเช่นรถยนต์ อวัยวะ หรือโมเลกุล สิ่งที่ OpenGL เตรียมไว้ให้สำหรับสร้างรูปจำลองสามมิติคือรูปทรง เรขาคณิตอย่างง่ายได้แก่ จุด เส้น และรูปหลายเหลี่ยมซึ่งผู้ใช้งานจะต้องนำรูปทรงเหล่านี้มาประกอบกัน เพื่อให้เกิดรูปทรงสามมิติที่ซับซ้อน (ไลบรารี Open Inventor ถูกสร้างขึ้นจากชุดคำสั่งของ OpenGL เพื่อช่วย ให้ผู้ใช้สามารถกำหนดสร้างรูปทรงที่ซับซ้อนได้โดยง่าย)

คณะกรรมการที่ร่วมกันกำหนดมาตรฐานของ OpenGL หรือ OpenGL Architecture Review Board (ARB) ประกอบด้วยบริษัท 3DLabs, Apple, ATI, Dell, Evans & Sutherland, HP, IBM, Intel, Matrox, NVIDIA, SGI, และ Sun Microsystems ปัจจุบัน OpenGL ได้ถูกพัฒนามาจนถึงเวอร์ชัน 3.0

OpenGL ถูกควบคุมโดยคณะกรรมการ OpenGL Architecture Review Board ตัวไลบรารีของ OpenGL ต่างจาก DirectX เนื่องจากมันสนับสนุนการทำงานเฉพาะกราฟฟิกสองมิติและสามมิติ ผู้เขียนเกมต้องใช้ไลบรารีตัวอื่นช่วยเพื่อทำให้เกมสมบูรณ์ ทำให้มีเกมเพียงไม่มากที่เขียนด้วย OpenGL อย่างเช่น Quake, DOOM, Half Life, Unreal และ Call of Duty แต่ชุดคำสั่งของ OpenGL สามารถทำงานได้บนทุกแพลตฟอร์ม การทำงานค่อนข้างเสถียรและการเรียนรู้ค่อนข้างง่ายโดยมีหนังสือประกอบมากมาย สำหรับงานด้านวิจัยและพัฒนาแล้ว OpenGL ได้รับความนิยมอย่างสูง

คำสั่งและชนิดของข้อมูลของ OpenGL

OpenGL Command Number of Input Input Data Type Vector

ภาพที่ 2.12 แสดงองค์ประกอบของคำสั่ง OpenGL

การใช้งานคำสั่ง OpenGL จะต้อง include ไฟล์ header ชื่อ gl.h คำสั่งของ OpenGL จะขึ้นต้นด้วย gl อย่างเช่น glClearColor() ค่าคงที่จะขึ้นต้นด้วย GL_ เช่น GL_COLOR_BUFFER_BIT สำหรับตัวลงท้ายหรือ suffix ของบางคำสั่งจะประกอบด้วยตัวเลขและตัวอักษรเช่น glColor3f ซึ่งเลข 3 บ่งบอกถึงจำนวนตัวแปรอินพุทคือสี RGB หากเป็น glColor4f จะหมายถึง RGBA ส่วน f หมายถึงอินพุทจะต้องเป็นข้อมูลชนิดเลขทศนิยม บางคำสั่งอาจรับข้อมูลอินพุทที่แตกต่างกันได้ถึง 8 ชนิด ตารางที่ 1 แสดง suffix ที่ใช้กำหนดชนิดของข้อมูลเทียบกับ ANSI C และ OpenGL Type Definition

Suffix	Data Type	ANSI C Type Def.	OpenGL Type Def.
b	8-bit integer	signed char	GLbyte
s	16-bit integer	short	GLshort
i	32-bit integer	long	Glint, GLsizei

f	32-bit floating-point	float	GLfloat,GLclampd
d	64-bit floating-point	double	GLdouble, GLclampd
ub	8-bit unsigned integer	unsigned char	GLubyte, GLboolean
us	16-bit unsigned integer	unsigned short	GLushort
ui	32-bit unsigned integer	unsigned long	GLuint, GLenum
-	-	void	GLvoid

ตารางที่ 2.5 แสดง Suffix ของคำสั่งและชนิดของข้อมูล

ดังนั้นคำสั่ง glVertex2i (1, 3) และ glVertex2f (1.0, 3.0) จึงทำหน้าที่เหมือนกันต่างกันที่คำสั่งแรกจะต้องกำหนดข้อมูลอินพุทเป็นเลขจำนวนเต็มและคำสั่งหลังกำหนดเป็นเลขทศนิยม

คำสั่งบางคำสั่งใน OpenGL มี suffix ตัวสุดท้ายเป็น v ซึ่งหมายความว่าคำสั่งนั้นรับข้อมูลอินพุทเป็นพอยน์เตอร์ที่ชี้ไปยังอาร์เรย์ของค่าแทนที่จะเป็นตัวเลข ตัวอย่างคำสั่งที่เหมือนกันแต่รับอินพุทต่างกัน

```
glColor3f(1.0, 0.0, 0.0);
float color_array[] = (1.0, 0.0, 0.0);
glColor3fv(color_array);
```

การทำงานของ OpenGL จะเป็นลักษณะ State Machine คือเมื่อสั่งให้โปรแกรมอยู่ในสถานะใดสถานะนั้นจะคงอยู่ตลอดจนกว่าจะมีการเปลี่ยนแปลง อย่างเช่นการตั้งค่าสีซึ่งเป็น State Variable เมื่อตั้งค่าสีไปครั้งหนึ่ง วัตถุที่ถูกวาดหลังจากนั้นจะมีสีนั้นตลอดจนกว่าจะมีคำสั่งเปลี่ยนสี

GLU: OpenGL Utility Library

GLU เป็นการเอาชุดคำสั่งของ OpenGL ที่ถูกใช้บ่อยๆมารวมกันเป็นคำสั่งใหม่ซึ่งทำให้การเขียนโปรแกรมสะดวกขึ้นอย่างเช่นการทำ Mipmapping สำหรับ Texture Mapping, การแปลงโคออร์ดิเนตแบบ Orthogonal และ Perspective, การแบ่งซอยรูปเหลี่ยมหรือ Polygon Tessellation, การเรนเดอร์รูปทรงพื้นฐานเช่นทรงกลม ทรงกระบอก และแผ่นจาน, และการวาดเส้นโค้งและพื้นผิวโค้งโดยใช้ Non-UniformRational B-Spline (NURBS) คำสั่งของ GLU จะขึ้นต้นด้วยคำว่า glu เสมอ การเรียกใช้จะต้อง include ไฟล์ header ชื่อ glu.h ข้อมูลเพิ่มเติมเกี่ยวกับคำสั่งของ GLU ที่

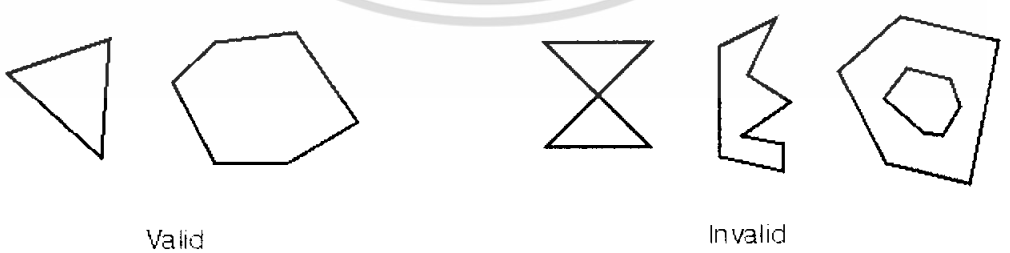
GLUT: OpenGL Utility Toolkit

GLUT เขียนขึ้นโดย Mark Kilgard เป็นเครื่องมือช่วยติดต่อระหว่างแอ๊พพลิเคชันของ OpenGL กับระบบปฏิบัติการแบบ Windows หรือเรียกว่าเป็น API (Application Programming Interface) ทำให้การเขียนโปรแกรมโดยใช้ OpenGL มีความซับซ้อนน้อยลง เหมาะสำหรับการพัฒนาโปรแกรมขนาดเล็กถึงขนาดกลาง คำสั่งของ GLUT จะขึ้นต้นด้วยคำว่า glut

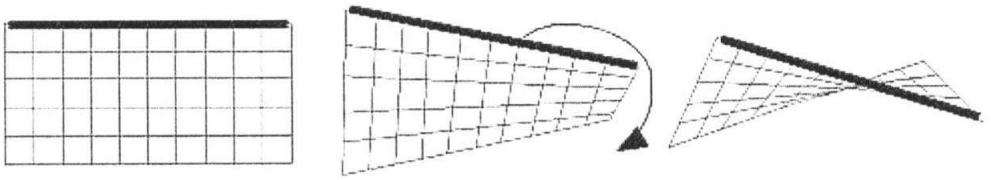
รูปทรงพื้นฐาน (Geometric Primitives)

รูปทรงพื้นฐานของ OpenGL ได้แก่ จุด, เส้น, และรูปเหลี่ยม โดยรูปทรงทั้งสามนี้ถูกกำหนดตำแหน่งโดยคู่ลำดับสามมิติ *(x, y, z)* หรือคู่ลำดับโฮโมจีเนียส *(x, y, z, w)* (โดยปกติแล้ว $w = 1$) ที่เรียกว่า vertex ซึ่งใช้ 1 vertex สำหรับจุด 2 vertex สำหรับเส้น และมากกว่า 2 vertex สำหรับรูปเหลี่ยม OpenGL ได้กำหนดลักษณะของรูปเหลี่ยมที่เหมาะสมโดยใช้ข้อบังคับสามข้อคือ ขอบของรูปเหลี่ยมจะต้องไม่ตัดกันเอง รูปทรงของรูปเหลี่ยมจะต้องไม่แหว่งเว้าเข้าไป และจะต้องไม่มีรูภายในรูปเหลี่ยม (ดูภาพที่ 1 ประกอบ) แต่ไม่มีข้อกำหนดสำหรับจำนวนมุมของรูปเหลี่ยม ข้อบังคับเหล่านี้ทำให้การแสดงผลมีความรวดเร็ว และหากใช้รูปเหลี่ยมที่ไม่เหมาะสมก็อาจทำให้การเรนเดอร์ภาพผิดพลาดได้ อย่างไรก็ตามรูปเหลี่ยมที่ไม่เหมาะสมสามารถแบ่งเป็นรูปเหลี่ยมย่อยๆได้โดยผู้ใช้หรือใช้ไลบรารี GLU ซึ่งมีคำสั่งที่ใช้ซอยรูปเหลี่ยมที่ซับซ้อนด้วย (Tessellation)

ข้อควรระวังสำหรับการเรนเดอร์รูปเหลี่ยมคือเมื่อ vertex ของรูปเหลี่ยมไม่ได้วางอยู่บนระนาบเดียวกัน การหมุนและการเปลี่ยนมุมมองอาจทำให้รูปเหลี่ยมนั้นกลายเป็นรูปเหลี่ยมที่ไม่เหมาะสม ได้ตัวอย่างในภาพที่ 2 เมื่อรูปสี่เหลี่ยมมี vertex ที่ไม่ได้อยู่บนระนาบเดียวกันโดยวางเหลื่อมกันเล็กน้อย เมื่อเกิดการหมุนจะทำให้รูปสี่เหลี่ยมนั้นมีลักษณะที่ไม่เหมาะสมคือเว้าเข้าไป ซึ่งจะทำให้การเรนเดอร์ผิดพลาดปัญหานี้สามารถหลีกเลี่ยงได้หากใช้เฉพาะรูปสามเหลี่ยมเพราะจุดทั้งสามจะวางตัวอยู่ในระนาบเดียวกันแน่นอน



ภาพที่ 2.13 รูปเหลี่ยมที่เหมาะสมและไม่เหมาะสม



ภาพที่ 2.14 รูปเหลี่ยมที่ไม่ได้ระนาบเมื่อหมุนเปลี่ยนมุมมองจะทำให้เกิดส่วนเว้า

เส้นโค้งและพื้นผิวโค้งสำหรับ OpenGL แล้วไม่ถือว่าเป็นรูปทรงพื้นฐาน แต่สามารถประมาณได้โดยใช้เส้นตรงหรือพื้นผิวย่อยๆมาประกอบกัน ดังภาพที่ 3 และ 4

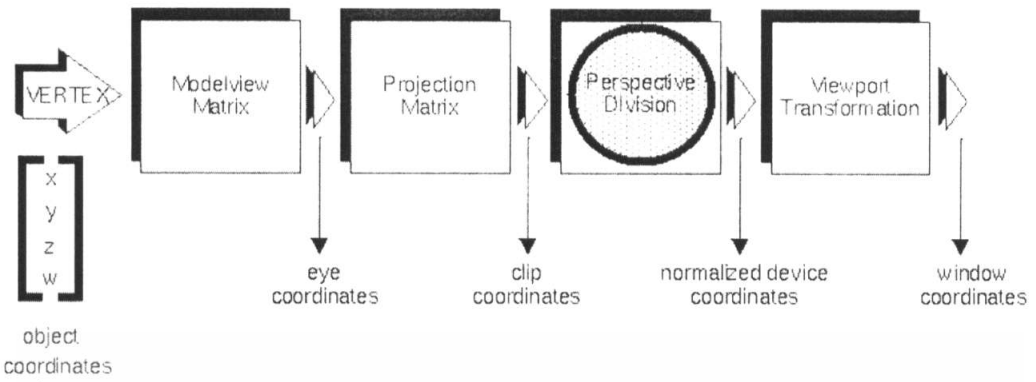
ภาพที่ 2.15 เส้นโค้งที่เกิดจากการประกอบกันของเส้นตรง

ภาพที่ 2.16 ทรงกลมที่เกิดจากการประกอบกันของพื้นผิวเรียบ

การจัดวางตำแหน่งวัตถุและการฉายภาพลงบนจอ

หากมี Vertex v ซึ่งเป็นโคออร์ดิเนต (x, y, z, w) และเมตริกการแปลง (Transformation Matrix) M ขนาด 4×4 การแปลงโคออร์ดิเนตหรือ Transformation จะเขียนในรูปเมตริกได้คือ

$$v' = Mv \tag{1}$$



ภาพที่ 2.17 กระบวนการแปลงโคออร์ดิเนตสามมิติของวัตถุไปเป็นพิกเซลบนจอภาพ

จากรูปแสดงกระบวนการแปลงโคออร์ดิเนตสามมิติของวัตถุไปเป็นพิกเซลบนจอภาพโดยแต่ละขั้นตอนสามารถเทียบได้กับขั้นตอนการถ่ายภาพ ขั้นตอนทั้ง 4 สามารถอธิบายได้ดังนี้

- *Modelview Transformation* – รวมสองขั้นตอนคือ Viewing Transformation เป็นการเปลี่ยนมุมมองซึ่งเทียบได้กับการจัดตำแหน่งกล้องและ Modeling Transformation ประกอบด้วยการเลื่อน (Translation), การหมุน (Rotation), และการสเกลวัตถุ (Scaling) ซึ่งเทียบได้กับการจัดตำแหน่งวัตถุ การแปลงทั้งสองจะเปลี่ยน Object Coordinates เป็น Eye Coordinates

ภาพที่ 2.18 ลำดับที่แตกต่างกันเมื่อทำการหมุนและเลื่อนวัตถุ ให้ผลลัพธ์ที่แตกต่างกันมาก

การทำ Modelview Transformation นั้นค่อนข้างซับซ้อน สมมติว่ามีวัตถุวางอยู่บนจุดกำเนิดและต้องการหมุนวัตถุรอบแกน z ทวนเข็มนาฬิกาเป็นมุม 45 องศา ร่วมกับการเลื่อนวัตถุไปตามแกน $+x$ หากวัตถุถูกหมุน 45 องศาก่อนที่จะเลื่อนไปตามแกน $+x$ วัตถุจะวางตัวอยู่บนแกน x (ภาพ

ที่ 6.a) แต่ถ้าหากเลื่อนวัตถุไปตามแกน $+x$ ก่อนที่จะหมุนวัตถุ วัตถุจะวางตัวอยู่บนเส้นตรง $y=x$ (ภาพที่ 6.b) จะเห็นว่าลำดับของการ Transform มีความสำคัญมาก

- *Projection Transformation* – เป็นการกำหนดรูปร่างของปริมาตรการมองหรือ Viewing Volume เป็น orthogonal หรือ Perspective และกำหนดขอบเขตของปริมาตรซึ่งมีผลทำให้บางส่วนของวัตถุถูกตัดออกจากฉาก เทียบได้กับการเลือกเลนส์กล้องและการปรับระยะวัตถุ ขั้นตอนนี้จะให้ Clip Coordinates

- *Perspective Division* – จะทำการหารค่าของโคออร์ดิเนต (x, y, z, w) ด้วยค่าของ w ทำให้ได้ Normalized Device Coordinates

- *Viewport Transformation* – เป็นการเปลี่ยนโคออร์ดิเนตของวัตถุไปเป็นโคออร์ดิเนตของจอภาพซึ่งสามารถทำให้ภาพขยาย ยืดหรือหดได้ เทียบได้กับการปรับขนาดของภาพในขั้นตอนการอัดรูป ขั้นตอนนี้จะให้ Window Coordinates

หลังจากการแปลงโคออร์ดิเนตจนได้ Window Coordinates แล้ว การเรนเดอร์ภาพลงบนหน้าจอจะใช้ทั้งโคออร์ดิเนต (x, y, z) โดย (x, y) กำหนดจุดพิกเซลบนจอที่จะวาด และ z เป็นตัวเก็บค่าความลึกของ vertex โดยวัดระยะจากจอภาพ ค่าความลึกนี้มีประโยชน์ในการตัดส่วนที่ไม่จำเป็นออก เช่น ถ้ามี vertex สองจุดที่มีค่าพิกัด x และ y เดียวกันแต่มีค่า z ต่างกัน ค่า z จะถูกใช้ในการเปรียบเทียบเพื่อตัดสินใจว่าพื้นผิวไหนถูกบังอยู่ และจะไม่วาดพื้นผิวส่วนนั้น โดยเรียกเทคนิคนี้ว่า Hidden-Surface Removal

แสง

แสงมีความสำคัญมาก ในการสร้างภาพสามมิติ วัตถุส่วนใหญ่จะไม่ดูเหมือนสามมิติจนกว่าจะมีการให้แสงจะเห็นว่าทรงกลมที่ยังไม่ให้แสงจะมีลักษณะเหมือนแผ่นจาน 2 มิติ

ภาพที่ 2.19 ทรงกลมที่ไม่มีแสงตกกระทบและทรงกลมที่มีแสงตกกระทบ

เมื่อแสงตกกระทบวัตถุ ตาเราจะสามารถรับสีที่แตกต่างกันของวัตถุขึ้นอยู่กับพลังงานของโฟตอนที่ตกกระทบเซลล์ประสาทรูปกรวย โฟตอนเหล่านั้นมาจากแหล่งกำเนิดแสง บางส่วนของโฟตอนจะถูกดูดกลืนและส่วนที่เหลือจะถูกสะท้อนโดยพื้นผิวของวัตถุ พื้นผิวที่แตกต่างกันจะมีคุณสมบัติการสะท้อนแสงที่แตกต่างกัน พื้นผิวที่มันเงาจะสะท้อนแสงไปยังทิศทางที่เจาะจง

พื้นผิวหยาบจะสะท้อนแสงให้กระจายไปในทุกทิศทาง แต่โดยปกติแล้วพื้นผิวส่วนใหญ่จะมีคุณสมบัติอยู่ระหว่างพื้นผิวสองแบบนี้

OpenGL จะคำนวณแสงโดยแยกแสงเป็น 3 องค์ประกอบคือแดง เขียวและน้ำเงิน (RGB) ดังนั้นสีของแสงจากแหล่งกำเนิดจะถูกกำหนดโดยปริมาณขององค์ประกอบเหล่านี้ และสีของพื้นผิววัตถุจะถูกกำหนดโดยเปอร์เซ็นต์ของแสงสีแดงเขียวและน้ำเงินที่ตกกระทบและถูกสะท้อนออกไป ตัวอย่างเช่นลูกบอลสีแดงจะดูดกลืนแสงสีเขียวและแสงสีน้ำเงินแต่จะสะท้อนแสงสีแดง เมื่ออยู่ในแสงขาวจะมองเห็นเป็นสีแดง แต่ถ้าอยู่ในแสงสีเขียวมันจะกลายเป็นสีดำ สมการคำนวณแสงของ OpenGL นั้นเป็นแค่การประมาณการแต่ให้ผลลัพธ์ที่ดีในเวลาคำนวณที่สั้น

แหล่งกำเนิดแสงใน OpenGL สามารถกำหนดให้มาได้จากหลายทิศทาง โดยสามารถกำหนดให้มาจากทิศทางที่แน่นอนได้หรือจะให้เป็นแสงที่สะท้อนไปมารอบๆฉาก ตัวอย่างเช่นเมื่อเราเปิดหลอดไฟในห้อง แสงส่วนใหญ่จะมาจากหลอดไฟนั้นโดยตรง แต่แสงบางส่วนจะสะท้อนไปมาในผนังห้องหลายรอบก่อนที่จะถึงตาเรา แสงที่สะท้อนไปมาในห้องนี้เรียกว่า Ambient Light ซึ่งถูกสมมติว่าไม่ทราบแหล่งกำเนิดที่แน่นอนแต่จะหายไปเมื่อเราปิดหลอดไฟนั้น

โมเดลของแหล่งกำเนิดแสงใน OpenGL จะถูกแบ่งเป็น 4 แบบใหญ่คือ Emitted, Ambient, Diffuse, และ Specular Light โดยโมเดลทั้ง 4 นี้จะถูกกำหนดองค์ประกอบสี RGB ที่แตกต่างกันและถูกคำนวณแยกกันแล้วนำมารวมกันภายหลัง

- *Emitted Light* – เป็นแสงที่ส่งออกมาจากตัววัตถุเอง

- *Ambient Light* – เป็นแสงที่กระจัดกระจายจากการสะท้อนจนไม่รู้แหล่งที่มา ทำให้ดูเหมือนกับมาจากทุกทิศทาง เมื่อแสงแบบนี้ตกกระทบวัตถุจะสะท้อนเท่าๆกันในทุกทิศทาง แสงจากหลอดไฟแบ็คไลท์ในห้องจะมีองค์ประกอบนี้อยู่มากเนื่องจากถูกสะท้อนหลายครั้งก่อนที่จะมาถึงตา ส่วนแสงจากสปอทไลท์จะมีองค์ประกอบนี้น้อยมากเนื่องจากแสงจะพุ่งไปในทิศทางเดียว

- *Diffuse Light* – เป็นแสงที่มาจากแหล่งกำเนิดโดยตรงแต่เมื่อตกกระทบวัตถุจะสะท้อนออกไปเท่ากันทุกทิศทาง

- *Specular Light* – มาจากแหล่งกำเนิดที่เจาะจงเช่นกันแต่จะสะท้อนไปในทิศทางที่เจาะจง คือมุมสะท้อนเท่ากับมุมตกกระทบ แสงจากเลเซอร์ที่ตกกระทบกระจกที่มีคุณภาพจะให้การสะท้อนเป็น Specular เกือบ 100 เปอร์เซ็นต์ วัตถุที่มันเงาเช่นพลาสติกจะมีคุณสมบัติของ Specular มากกว่าวัตถุที่หยาบเช่นชอล์ค

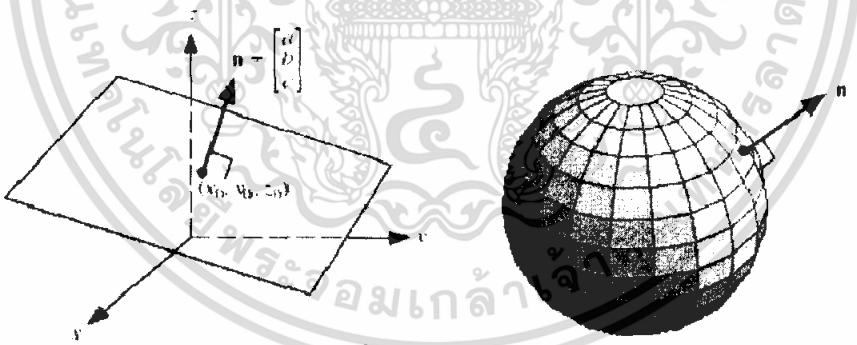
วัตถุมีโมเดลการสะท้อนแสงสามแบบคือ Ambient, Diffuse, และ Specular Reflectance ที่ถูกกำหนดสีไว้แตกต่างกัน Ambient Reflectance ของวัตถุจะใช้สำหรับคำนวณสีของวัตถุที่แสดงออกมาเมื่อถูก Ambient Light ตกกระทบ เช่นเดียวกับ Diffuse และ Specular Reflectance ซึ่งคำนวณคู่กับ Diffuse และ Specular Light ตามลำดับ โดยปกติแล้วสีของวัตถุที่ถูกกำหนดโดย Ambient และ Diffuse Reflectance จะเหมือนหรือคล้ายคลึงกัน ส่วน Specular Reflectance จะ

กำหนดเป็นสีขาวหรือเทา ซึ่งทำให้จุดที่สะท้อนเป็นสีของแหล่งกำเนิดแสง อย่างเช่นลูกบอลมันเงาสีแดงเมื่อถูกฉายด้วยแสงสีขาวพื้นที่ส่วนใหญ่จะเป็นสีแดงแต่จะมีส่วนหนึ่งที่มีมุมตกกระทบเท่ากับมุมสะท้อนซึ่งจะเป็นสีขาว

การกำหนดปริมาณสีสำหรับแสงจะกำหนดเป็นเปอร์เซ็นต์ของความเข้มสูงสุดของแต่ละสี ถ้าค่า RGB เป็น 1.0 ทั้งหมดจะได้แสงสีขาว ถ้าเป็น 0.5 ทั้งหมด จะได้สีขาวที่ความเข้มต่ำลงครึ่งหนึ่ง แต่ถ้า R=G=1.0, B=0.0 จะเป็นแสงสีเหลือง ส่วนการกำหนดปริมาณสีที่สะท้อนออกมาจากวัตถุจะกำหนดเป็นสัดส่วนของสีที่สะท้อนออกมา อย่างเช่นถ้า R=1.0, G=0.5, B=0.0 วัตถุจะสะท้อนแสงสีแดงทั้งหมด สะท้อนแสงสีเขียวเพียงครึ่งหนึ่งของแสงสีเขียวที่ตกกระทบและดูดกลืนแสงสีน้ำเงินทั้งหมด สมมติว่าแสงมีค่าความเข้มเป็น(LR, LG, LB) และวัตถุมีค่าการสะท้อนแสงเป็น (MR, MG, MB) แสงที่เข้าสู่ตาจะเป็น (LR*MR, LG*MG, LB*MB) ถ้าหากมีแสงจากสองแหล่งเข้าสู่ตาเป็น (R1, G1, B1) และ (R2, G2, B2) แสงรวมจะเป็น (R1+R2, G1+G2, B1+B2) หากค่าที่รวมได้มีค่ามากกว่า 1.0 ซึ่งสว่างเกินกว่าที่จอภาพจะแสดงผลได้ ค่านั้นจะถูกปัดลงเป็น 1.0

Normal Vector

Normal Vector เป็นเวคเตอร์ที่ชี้ไปในทิศทางตั้งฉากกับพื้นผิวของวัตถุ มีไว้สำหรับกำหนดลักษณะการวางตัวของพื้นผิว มันถูกใช้ในการคำนวณปริมาณแสงที่ตกกระทบพื้นผิวบริเวณนั้น OpenGL อนุญาตให้มีการกำหนด Normal Vector เฉพาะที่จุด Vertex ของพื้นผิวเท่านั้น

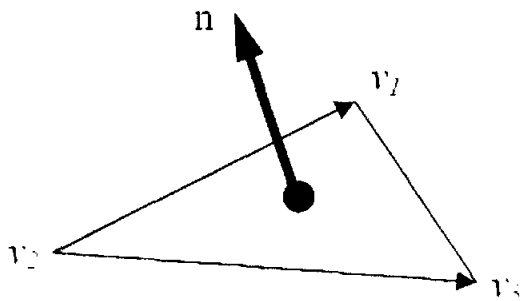


ภาพที่ 2.20 Normal Vector

หากมีสมการทางคณิตศาสตร์ที่ใช้อธิบายพื้นผิว เราสามารถหาค่า Normal Vector ที่แต่ละจุดบนพื้นผิวได้สมมติสมการพื้นผิวเป็น $z = f(x, y)$ จะหา Normal Vector ที่จุด $(x0, y0)$ ได้จาก

$$n = \begin{bmatrix} f_x(x_0, y_0) \\ f_y(x_0, y_0) \\ -1 \end{bmatrix}$$

เมื่อ $fx = \partial_f / \partial_x$ และ $fy = \partial_f / \partial_y$ เป็นอนุพันธ์ย่อยของฟังก์ชัน $f(x, y)$ [6]



ภาพที่ 2.21 Normal Vector ของรูปสามเหลี่ยมที่มีจุดยอดที่ *v1*, *v2*, และ *v3*

แต่หากพื้นผิวประกอบด้วยรูปเหลี่ยมและไม่มีสมการอธิบายแล้วจะมีสูตรการคำนวณ Normal Vector คือ

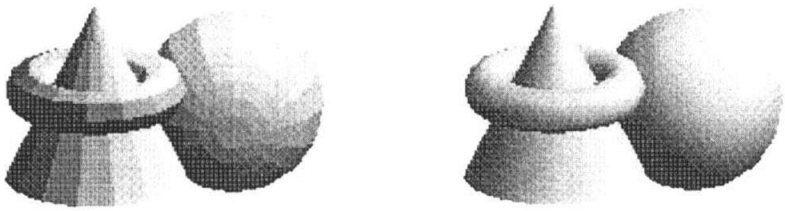
$$n = [v_1\text{-}v_2]\text{x}[v_2\text{-}v_3] = \begin{matrix} x_1 - x_2 \\ y_1 - y_2 \\ z_1 - z_2 \end{matrix} \times \begin{matrix} x_2 - x_3 \\ y_2 - y_3 \\ z_2 - z_3 \end{matrix} = \begin{matrix} (y_1 - y_2)(z_2 - z_3) - (z_1 - z_2)(y_2 - y_3) \\ (z_1 - z_2)(x_2 - x_3) - (x_1 - x_2)(z_2 - z_3) \\ (x_1 - x_2)(y_2 - y_3) - (y_1 - y_2)(x_2 - x_3) \end{matrix}$$

เมื่อ $v1=(x1, y1, z1)$, $v2=(x2, y2, z2)$, และ $v3=(x3, y3, z3)$ เป็นจุดใดๆบนรูปเหลี่ยมที่ไม่วางอยู่บนเส้นตรงเดียวกัน Normal Vector ที่ได้จากการคำนวณทั้งสองวิธีจะต้องถูก Normalize ให้มีขนาดเป็น 1 ก่อนโดยใช้สูตร

$$n = \begin{matrix} \dfrac{x}{\sqrt{x^2 + y^2 + z^2}} \\ \dfrac{y}{\sqrt{x^2 + y^2 + z^2}} \\ \dfrac{z}{\sqrt{x^2 + y^2 + z^2}} \end{matrix}$$

สำหรับพื้นผิวเรียบแล้วทุกๆจุดจะมี Normal Vector ชี้ไปในทิศทางเดียวกันหมด แต่สำหรับพื้นผิวโค้งซึ่งประกอบด้วยพื้นผิวเรียบย่อยๆหากกำหนดให้ Normal Vector ตั้งฉากกับพื้นผิวเรียบย่อยๆแล้ว เมื่อแสดงผลจะมองเห็นรอยต่อของพื้นผิวได้ชัดเจนเนื่องจากทิศทางของ Normal Vector ไม่ต่อเนื่องที่บริเวณรอยต่อของพื้นผิวดังภาพที่ 2.21 ซึ่งปัญหานี้สามารถแก้ได้โดยใช้การเฉลี่ยค่า

Normal Vector ที่รอยต่อ ตัวอย่างในภาพที่ 2.22 ที่จุด P ค่า Normal Vector ควรเป็นค่าเฉลี่ยจาก Normal Vector สี่จุดรอบๆหรือ $np = n1+n2+n3+n4$



ภาพที่ 2.22 เรนเดอร์โดยใช้Normal Vectorของรูปเหลี่ยม(ซ้าย) และใช้ Normal Vector จริง(ขวา)

ภาพที่ 2.23 การเฉลี่ยค่า Normal Vector ที่จุด Vertex

Java Bindings for OpenGL

ในหลายปีมานี้ โปรแกรมเมอร์ที่ต้องการเขียนโปรแกรมกราฟฟิกขั้นสูงที่จะสามารถขายให้กับผู้ใช้ทั่วไปที่ใช้ ระบบปฏิบัติการณ์ที่แตกต่างกันไปนั้น ส่วนใหญ่จะใช้ OpenGL โดยที่ OpenGL นั้นถูกจดทะเบียนโดย SGI OpenGL นั้นแสดงให้เห็นถึงการทำงานข้ามรูปแบบ API ของภาษา C แต่ในความเป็นจริงแล้วมันเป็นฮาร์ดแวร์ที่ทำงานเป็นเอกเทศสำหรับหน้าจอของโปรแกรม

OpenGL นี้มีไว้สำหรับทำกราฟฟิก โดยมันจะเร็วมาก และเวลาส่วนใหญ่ที่ใช้ก็คือการทำงานของฮาร์ดแวร์ มันเหมือนกับว่า OpenGL สามารถทำอะไรก็ได้ตามแต่ที่ต้องการ

น่าเสียดายที่ OpenGL นั้นเขียนมาสำหรับภาษา C แต่ว่าภาษา C นั้นไม่ใช่ภาษาที่โปรแกรมเมอร์ส่วนใหญ่นิยมจะใช้เขียนโปรแกรมที่ยุ่งยาก และข้อเสียที่ใหญ่ที่สุดก็คือ OpenGL จะทำอะไรไม่ได้เลยหากไม่มีหน้าต่างให้ใส่กราฟฟิก แต่ OpenGL ก็ไม่ได้จัดการเรื่องสร้างหน้าต่างให้ ทำให้ยากแก่ผู้ที่เริ่มต้นทางด้าน OpenGL

แต่ยังดีที่ GLUT (อุปกรณ์เสริมของ OpenGL) ได้ทำการให้มีการติดต่อกับหน้าต่าง ปุ่มและเหตุการณ์ต่างๆที่ถูกสร้างขึ้นมาจากผู้ใช้งานได้ง่ายขึ้น ยังไงก็ตามการเรียน OpenGL ในภาษา C หรือ C++ นั้นก็ยังยากสำหรับโปรแกรมเมอร์มือใหม่ หรือโปรแกรมเมอร์ที่ต้องการเขียนโปรแกรมเชิงวัตถุ

Java นั้นเป็นการเขียนโปรแกรมเชิงวัตถุ มีการพยายามอย่างมากที่จะเอา OpenGL มาผสมร่วมกับ Java และคนที่ทำให้ทุกคนตระหนักถึงการผสม Java เข้ากับ OpenGL หรือเรียกรวมกันว่า JOGL ก็คือบริษัท Sun Microsystems (ผู้สร้าง Java) และ SGI (ผู้สร้าง OpenGL)

JOGL เป็น Open source ที่ถูกเริ่มพัฒนาขึ้นมาโดยกลุ่มเกมเทคโนโลยีใน Sun Microsystems เมื่อปี 2003

JOGL ได้จัดเตรียม API ที่สามารถที่สามารถใช้ OpenGL 2.0 ตามข้อกำหนดได้อย่างเต็มรูปแบบเหมือนกับที่ขายตามท้องตลอดและยังสามารถใช้งานร่วมกับ AWT และ Swing ได้ และ JOGL นั้นยังสนับสนุนภาษาในการ shader ของทั้ง GLSL และ Nvidia

JOGL นั้นมุ่งความสนใจไปที่ 2D และ 3D เรนเดอร์เช่นเดียวกับ OpenGL ซึ่งไม่ได้รวมส่วนที่สนับสนุนสำหรับเกมอื่นๆเช่น เสียง หรือเครื่องมืออินพุตต่างๆ ซึ่งทั้งคู่ที่กล่าวมานั้นสามารถไปใช้ในส่วนของ JOAL และ JInput ได้

คุณสมบัติส่วนใหญ่ที่อยู่ใน OpenGL GLU และ GLUT ไลบรารีนั้นได้ถูกบรรจุอยู่ใน JOGL แล้ว

GLU (เป็น Utility Library ของ OpenGL) ได้รวมความสามารถในการสนับสนุนสำหรับการเรนเดอร์ทรงกลม, ทรงกระบอก, ดิสก์, ตำแหน่งของกล้อง, การทำพื้น และการทำ texture mipmaps ใน JOGL เวอร์ชันของ GLUT (Utility Toolkit ของ OpenGL) นั้นไม่ได้รวมความสามารถการทำงานภายใต้ การทำงานของวินโดวส์ซึ่งสามารถจัดการโดย JAVA แต่ว่าจะเสนอเป็นข้อมูลเชิงเรขาคณิตเป็นหลัก (ทั้งในโหมด solid และ wireframe) ใน JOGL นั้นยังรวมถึงส่วน การทำ frame-based animation, texture loading, file IO, and ความสามารถในการทำสกรีนช็อต

JOGL นั้นค่อยๆพัฒนาเพื่อนำไปใช้อ้างอิงสำหรับข้อกำหนดของ JSR-231 เพื่อการรวม OpenGL เข้ากับ Java JOGL 1.1.1 ถูกเข้ามาแทนที่โดย JSR-231 ในเดือนตุลาคมปี 2005 และในปัจจุบันได้ออกมาเป็นเวอร์ชัน 1.1.0-rc2 ในเดือนมกราคมปี 2007

คลาส GLDrawable และ GLContext ที่เพิ่มเข้ามานั้นมีความสำคัญในการทำส่วนที่จะกล่าวถึงอย่างมากเพราะสามารถที่จะทำให้เข้าถึงการวาดพื้นผิวของ OpenGL และข้อมูลของสถานะได้โดยตรง

Javadocs สำหรับ JOGL จะได้มาจากที่เดียวกันกับ the binary distribution of JOGL โดย javadocs นั้นจะมีชื่อที่คล้ายๆกับ jogl-1.1.0-docs.zip

ถ้าได้ทำการดูแพ็คเกจของ javax.media.opengl และ com.sun.opengl แล้วจะรู้ว่าคลาสนั้นมีขนาดใหญ่มาก คุณจะสามารถนำความรู้ที่ได้จาก JOGL นั้นไปประยุกต์กับการทำงานจริงๆได้อย่างมากมาย และคลาสที่เราควรจะศึกษา คือ

GLDrawable, GLCanvas, GLJPanel, GLCapabilities, GLDrawableFactory

ในนี้จะมีพื้นฐานของการจัดการทางด้านหน้าจอทางด้านกราฟฟิกอยู่ จากข้างต้นที่ได้กล่าวถึงจุดด้อยของผู้ที่เริ่มเรียนก็คือ OpenGL ขาดมาตรฐานหรือความรู้ของจัดการระบบวินโดวส์หรือการสร้างหน้าต่างสำหรับแสดงผล GLUT ก็ได้ทำการช่วยเหลือในการจัดการอย่างอัตโนมัติโดยที่ที่ใช้งานไม่ต้องเข้าไปจัดการให้วุ่นวาย แต่ว่า Swing และ AWT (Abstract Window Toolkit) นั้นใช้งานง่ายอยู่แล้วพร้อมทั้งวิธีการจัดการก็ไม่แตกต่างจากการใช้ GLUT

ควรตระหนักว่า OpenGL นั้นเขียนมาสำหรับภาษา C หมายความว่าหาก Java ต้องการที่จะใช้งานนั่นคือจะต้องมี interface ที่เป็นการเชื่อมภาษาภายนอกซึ่งทำได้โดยใช้ JNI (Java Native Interface) ซึ่งไม่แปลกเลยที่ควรจะมีการเขียนการเชื่อมต่อขึ้นมาให้ใช้งานง่าย OpenGL นั้นค่อนข้างจะใหญ่ การเขียนการเชื่อมต่อทั้งหมดนั้นจะกินเวลามาก และทำให้เกิดความยุ่งยากขึ้น มันมีการปรับเปลี่ยนและพัฒนาในแต่ละเวนเดอร์รวมไปถึง OpenGL ด้วย ซึ่งในอีกไม่นาน ก็จะเป็นการยากที่จะตามการพัฒนาของ OpenGL ได้ทันเพื่อที่จะเขียน Java ให้เป็น native interface สำหรับแต่ละรุ่น

ในกลุ่ม JOGL นั้นได้ตัดสินใจที่จะทำให้มีข้อได้เปรียบของไฟล์ C header โดยเขียนโค้ดที่จะทำงานให้แก่ JNI ซึ่งทั้งหมดนี้เรียกว่า GlueGen โดยที่ GlueGen จะพาร์ซไฟล์ C header ออกมาเพื่อที่จะปรับปรุงโค้ด Java และ JNI ที่ต้องการเชื่อมต่อกับไลบรารีได้ด้วย จะหมายความว่าการอัพเดท OpenGL จะสามารถเพิ่มเข้าไปใน JOGL ได้อย่างรวดเร็ว

จุดเด่นของ JOGL ที่แตกต่างไปจากไลบรารี่ที่เป็น wrapper ตัวอื่นคือการแยกคำสั่งต่างๆของ OpenGL API ไปเป็นเมธอดในคลาสต่างๆแทนที่จะบังคับให้เขียนเป็นโปรแกรมเชิงวัตถุทำให้เราสามารถใช้งานคำสั่งของ Opengl ได้แทบไม่แตกต่างจากการเขียนโดยภาษาซีเพียงแค่มีคลาสมาห่อหุ้มคำสั่งไว้เท่านั้น เช่นคำสั่งบนเฮดเดอร์ gl.h จะถูกเก็บไว้ในคลาสเราสามารถใช้งานคำสั่งต่างๆได้จากออบเจกต์ของคลาส GL ดังตัวอย่าง

```
glVertex2f(1.0f, 0.0f) ;
gl.glVertex2f(1.0f, 0.0f);
```

ด้านบนจะเป็นของภาษาซีส่วนด้านล่างจะเป็นของจาวาซึ่งจะเห็นได้ว่าแทบไม่มีอะไรเปลี่ยนแปลงเลยแม้แต่น้อยเพียงแค่มีการเรียกเมธอดจากคลาสเท่านั้นเอง

วิธีการติดตั้งพื้นฐาน

JOGL สามารถใช้งานร่วมกับ J2SE 1.4.2 ขึ้นไป ซึ่งในที่นี้จะใช้ Java 1.6.0 และ ดาวน์โหลด JSR-231 1.1.0 release candidate 2 ของ JOGL

ใน คู่มือการใช้ของ JOGL แนะนำว่า JARS และ DLLs ควรจะถูกติดตั้งในไดเรกทอรี่ของตนเองแทนที่จะไปอยู่ใน JRE ไดเรกทอรี่ ซึ่งตัว JARs และ DLLS สามารถใช้โดยการกำหนด classpath และ ค่า java.library.path บน command line

The Callback Framework

JOGL มีคลาสของ GUI หลักๆ 2 คลาสคือ GLCanvas และ GLJPanel ซึ่งนำมาใช้กับ GLAutoDrawable อินเตอร์เฟสซึ่งสามารถใช้งานได้เหมือนกับ *drawing surfaces* สำหรับคำสั่งใน OpenGL

GLCanvas นั้นถูกนำมาบรรจุไว้สำหรับนำไปใช้งานเหมือนกับ AWT's Cnavas คลาส และมันยังเป็น คอมโพเนนท์ขนาดใหญ่ตัวหนึ่งดังนั้นจึงต้องคำนึงเป็นอย่างมากในการนำไปใส่รวมกับ Swing อย่างไรก็ตาม GLCanvas นั้นสามารถทำงาน OpenGL ได้อย่างรวดเร็วมากเมื่อใช้คู่กับอุปกรณ์เร่งความเร็ว

GLJPanel นั้นเป็นเครื่องมือเล็กๆ ซึ่งใช้เพื่อแก้ปัญหาของ Swing ซึ่งในอดีตนั้นมีกิตติศัพท์ในเรื่องความช้าเนื่องจากมันจะทำการคัดลอก OpenGL frame buffer ลงไปยัง BufferedImage ก่อนที่จะทำการแสดงผล อย่างไรก็ตามใน Java SE 6 ก็ได้ถูกพัฒนาให้มีความเร็วขึ้นเป็นอย่างมาก สิ่งที่ทำให้ GLJPanel นั้นเหนือกว่า GLCanvas คือการยอมให้ ภาพ 3D ใน OpenGL กับ องค์ประกอบ 2D ใน Swing สามารถรวมกันได้ซึ่งเป็นหนทางใหม่ที่น่าตื่นตาตื่นใจ

การนำ GLCanvas ไปใช้

ออฟเจ็กต์ของ GLCanvas นั้นจะคู่กับ GLEventListener listener ซึ่งจะทำการตอบสนองการเปลี่ยนแปลงใน canvas และทำการวาดตามคำร้องขอ เมื่อ canvas ถูกสร้างในครั้งแรก GLEventListener's init() เมธอดจะถูกเรียกขึ้นมาซึ่งเมธอดนี้ใช้ในการใส่ค่าเริ่มต้นให้แก่สถานะของ OpenGL ไม่ว่า canvas จะถูกเปลี่ยนขนาดรวมถึงตอนที่วาดครั้งแรก GLEventListener's reshape() จะทำงานขึ้นมาซึ่งมันสามารถที่จะทำการเขียนทับค่าเริ่มต้นของ viewport และ projection matrix ใน OpenGL ซึ่ง reshape() นั้นจะถูกเรียกใช้เมื่อ canvas นั้นมีการนำไปใช้กับ parent คอมโพเนนท์ด้วย ไม่ว่าเมื่อไหร่ก็ตามที่ display() เมธอด ถูกเรียกใช้ display() เมธอดใน GLEventListener จะทำงาน โค้ดในการวาดฉาก 3D นั้นควรจะถูกวางอยู่ในเมธอดนี้ รองไปจาก canvas และ listener เกมส่วนมากต้องการกรรมวิธีสำหรับการทริกเกอร์การอัพเดต canvas ซึ่งความสามารถนี้มีใน JOGL's FPSAnimator utility class ซึ่งสามารถกำหนดตารางเวลาไปยัง canvas สำหรับการเรียก display() เมธอดตามความถี่ที่ได้ตั้งเอาไว้

2.4 ทฤษฎีและหลักการทางคณิตศาสตร์ที่เกี่ยวข้องกับโปรแกรม

2.4.1 LU Decomposition

กระบวนการในการแบ่งเมทริกA ที่มีขนาด NxN ให้มี 2 ส่วนคือ เมทริกสามเหลี่ยมล่าง L และเมทริกสามเหลี่ยมบน U จะได้ว่า

$$LU = A$$

เขียนเมทริกขนาด 3x3 โดยจะแบ่งได้ดังนี้

$$\begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix}\begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

$$\begin{bmatrix} l_{11}u_{11} & l_{11}u_{12} & l_{11}u_{13} \\ l_{21}u_{11} & l_{21}u_{12}+l_{22}u_{22} & l_{21}u_{13}+l_{22}u_{23} \\ l_{31}u_{11} & l_{31}u_{12}+l_{32}u_{22} & l_{31}u_{13}+l_{32}u_{23}+l_{33}u_{33} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

โดยจะได้สมการ 3 สมการคือ

$$i < j \; l_{i1}u_{1j} + l_{i2}u_{2j} + \ldots + l_{ii}u_{ij} = a_{ij}$$
$$i = j \; l_{i1}u_{1j} + l_{i2}u_{2j} + \ldots + l_{ii}u_{ij} = a_{ij}$$
$$i > j \; l_{i1}u_{1j} + l_{i2}u_{2j} + \ldots + l_{ii}u_{ij} = a_{ij}$$

วิธีในการแก้สมการ

$$Ax = (LU)x = L(Ux) = b,$$

เริ่มจากการแก้สมการ Ly = b สำหรับ y ทำได้โดยขั้นตอนดังนี้

$$y_1 = \frac{b_1}{l_{11}}$$

$$y_i = \frac{1}{l_{ii}}\left[b_i - \sum_{j=1}^{i-1} l_{ij}y_j \right]$$

ให้ $i = 2,\ldots N$. จากนั้นแก้สมการ $Ux = y$ สำหรับทุกๆ x ทำได้โดยขั้นตอนดังนี้

$$X_N = \frac{y_N}{u_{NN}}$$

$$x_i = \frac{1}{u_{ii}}\left[yi - \sum_{i=i+1}^{N} u_{ii} x_j \right]$$

สำหรับ $i = N-1,\ldots,1$

2.4.2 Gaussian Elimination

วิธีกำจัดแบบเกาส์นั้นจะใช้ในการแก้ปัญหาเมทริกที่อยู่ในรูป

$$Ax = b$$

โดยเริ่มต้นนั้นจะตั้งสมการดังนี้

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1k} \\ a_{21} & a_{22} & \cdots & a_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ a_{k1} & a_{k2} & \cdots & a_{kk} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_k \end{bmatrix}$$

แล้วมาทำให้เป็น Augmented matrix equation

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1k} \\ a_{21} & a_{22} & \cdots & a_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ a_{k1} & a_{k2} & \cdots & a_{kk} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_k \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{bmatrix}$$

หลังจากนั้นคอลัมน์ที่มีค่า x ก็ให้ทำการให้เป็นเมทริกสามเหลี่ยมบน

$$\left[\begin{array}{cccc|c} a'_{11} & a'_{12} & \cdots & a'_{1k} & b'_1 \\ 0 & a'_{22} & \cdots & a'_{2k} & b'_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & a'_{kk} & b'_k \end{array}\right]$$

แก้สมการ k แถวเพื่อหาค่า x_k ต่อมาก็ทำทำการแทนที่ในแถวที่ k-q เพื่อหาค่า x_{k-1} ไปเรื่อยๆ ก็จะได้ดังสูตรข้างล่าง

$$x_i = \frac{1}{a'_{11}}\left(b'_i - \sum_{j=i+1}^{k} a'_{ij}\, x_j\right)$$

2.4.3 Bisection

Bisection จะทำกาแบ่งกราฟออกเป็นสองส่วนเท่าๆกัน หลังจากนั้นก็จะทำการหาจุดกึ่งกลางของกราฟด้วย x = (a+b)/2 โดยจะดูในช่วงของ [a,(a+b)/2] หรือ [(a+b)/2,b] ที่คำตอบนั้นผิด หลังจากนั้นก็จะทำซ้ำไปเรื่อยๆ ตามที่ต้องการจนกว่าจะได้คำตอบตามที่ต้องการ

ให้ a_n และ b_n เป็นจุดสิ้นสุดในรอบที่ n และให้ r_n เป็นคำตอบของการประมาณรอบที่ n หลังจากนั้นจำนวนของการทำซ้ำนั้นให้ค่าที่น้อยกว่าค่า $\in$

$$b_n - a_n = \frac{(b-a)}{2^{n-1}}$$

และกำหนดให้ r_n เป็นดังนี้

$$r_n \equiv \frac{1}{2}(a_n + b_n)$$

และค่าผิดพลาดก็ควรจะน้อยกว่า $\in$ ตามสมการ

$$|r_n - r| \le \frac{1}{2}(b_n - a_n) = 2^{-n}(b-a) < \in$$

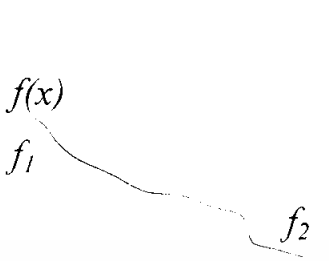
ทำการใส่ Natural logarithm ทั้งสองข้างของสมการ จะได้ดังนี้

$$-n\ln 2 < \ln \in - \ln(b-a)$$

ดังนั้น

$$n > \frac{\ln(b\ a) - \ln \in}{\ln 2}$$

2.4.4 Trapezoidal Rule



ภาพที่ 2.24 ภาพตัวอย่างของ Trapezoidal Rule

จากสมการ $$\int_{x_2}^{x_1} f(x)dx = \frac{1}{2}h(f_1 + f_2) - \frac{1}{12}h^3 f''(\xi)$$

โดยที่ $f_i \equiv f(x_i)$ และ h เป็นค่าความต่างของจุด และ $\in$ เป็นจุดที่เราพอใจโดยที่ $x_1 \le \xi \le x_2$ เลือกค่า ξ ที่ทำให้ $f''(\xi)$ มีค่ามากที่สุดและให้เป็นค่าขอบเขตบนสำหรับค่าผิดพลาดในการประมาณแบบ trapezoidal

บทที่ 3
ขั้นตอนการดำเนินงานและหลักการของโปรแกรม

ขั้นตอนในการดำเนินการแบ่งออกเป็นขั้นๆดังนี้

1. กำหนดรูปแบบของโปรแกรมเบื้องต้น
2. ออกแบบไวยากรณ์ของภาษา
3. ออกแบบข้อกำหนดของตัววิเคราะห์คำ (สแกนเนอร์) และข้อกำหนดของตัววิเคราะห์ไวยากรณ์ (พาร์สเซอร์) และสร้างคลาสจากข้อกำหนดทั้งสอง
4. สร้างตัวแปลภาษาจากข้อกำหนดและสร้างคลาสเพื่อรองรับการทำงานตัวแปลภาษา
5. สร้างส่วน GUI เพื่อรับคำสั่งเข้ามาให้ตัวแปลภาษาทำงาน
6. สร้างส่วน GUI ที่ทำการติดต่อกับไลบรารี่ของ JOGL เพื่อใช้ในการแสดงกราฟ
7. ทำให้ส่วนการแสดงผลกราฟสามารถแสดงผลออกทางเครื่องพริ้นเตอร์ได้

3.1 กำหนดรูปแบบของโปรแกรมเบื้องต้น

รูปแบบของโปรแกรมจะเป็นโปรแกรมที่ใช้ในการคำนวณทางคณิตศาสตร์ที่มีความซับซ้อนและแสดงผลออกเป็นแบบกราฟิก(Graphic) นอกจากนี้ยังสามารถใช้การแสดงกราฟของฟังก์ชั่นทางคณิตศาสตร์ที่เรากำหนดขึ้นมาได้

ลักษณะของโปรแกรมจะเป็นแบบส่วนเชื่อมต่อผู้ใช้งานแบบกราฟิก โดยแบ่งเป็นสองส่วนที่ใช้งานหลักคือ ส่วนแสดงผลลัพธ์ และส่วนของการรับคำสั่ง ใช้งานจากการป้อนคำสั่งต่างๆเข้ามา จากนั้นเมื่อผู้ใช้งานต้องการให้โปรแกรมประมวลผลคำสั่งก็จะกระทำได้โดยการกดปุ่ม Execute หรือ กดปุ่ม Enter ของคีย์บอร์ด จากนั้นโปรแกรมก็จะนำคำสั่งไปประมวลผลและแสดงผลลัพธ์ออกมา นอกจากนี้ผู้ใช้งานยังสามารถเลือกพิมพ์แบบบรรทัดเดียวหรือหลายๆบรรทัดก็ได้ แต่จะถือว่าเป็นเพียงแค่หนึ่งคำสั่งเท่านั้น

รูปแบบของคำสั่งจะคล้ายกับภาษาเบสิก (Basic) การกำหนดชนิดของตัวแปรจะเป็นแบบไดนามิกไทป์คือสามารถใส่ค่าลงในตัวแปรได้ทันทีโดยไม่ต้องมีการกำหนดชนิดของตัวแปร ยกเว้นในเรื่องของการทำฟังก์ชั่นบางตัวที่มีการกำหนดชนิดของอาร์กิวเมนต์ไว้แล้วซึ่งหากใส่ค่าไม่ถูกกับชนิดก็จะแสดงข้อความผิดพลาดออกมา นอกจากนี้ยังสามารถกำหนดตัวแปรแบบคอลเลคชั่นได้ซึ่งประกอบไปด้วยอะเรย์ เมตริกซ์และคิวบ์ (อะเรย์สามมิติ) ซึ่งจะสามารถใช้เก็บข้อมูลที่เป็นตัวเลขเท่านั้น

ลักษณะทั่วไปของภาษาเป็นดังนี้

3.1.1 ตัวดำเนินการพื้นฐาน

โปรแกรมเข้าใจตัวดำเนินการพื้นฐานและมีระดับความสำคัญของตัวดำเนินการจากต่ำไปสูงดังนี้ "+", "-", "*", "/", "%" และ "^"

```
ตัวอย่างเช่น
> 2+3
5.0
> 2*3
6.0
> 2^3
8.0
> 2/3
0.6666666666666666
```

โดยเราสามารถกำหนดลำดับโดยการใช้เครื่องหมายวงเล็บช่วยได้เช่น

```
> 2+3*4
14.0
> (2+3)*4
20.0
> 2*3^2
18.0
> (2*3)^2
36.0
> 2^3!
40320.0
> 2^(3!)
64.0
> 2+3!
120.0
> 2+(3!)
8.0
```

3.1.2 ตัวแปร

การใส่ค่าให้ตัวแปรนั้นสามารถทำได้โดยการใช้ตัวดำเนินการ = (equal) เช่น

```
> x=5
5.0
> x*2
10.0
> x^2
25.0
```

3.1.3 จำนวนเชิงซ้อน

รูปแบบการใช้จำนวนเชิงซ้อนนั้นสามารถใช้ได้โดยการใส่ตัวอักษร i ลงไป หากเป็น ค่า i ให้ใส่เป็น 1i ซึ่งรากที่สองของ i เป็น -1 เช่น

```
> 1+i
1.0 + i
> 1i^2
-1.0
> i^3
-1.0i
> 1i^4
1.0
```

3.1.4 คอลเลคชั่น

มีสามประเภทคือ อะเรย์, เมตริซ์, และคิวบ์รูปแบบการใช้งานจะใช้เครื่องหมาย “[“ และ “]” เป็นตัวบอกมิติ และ เครื่องหมาย “,” กำหนดตำแหน่งของข้อมูลข้อจำกัดคือเก็บข้อมูลได้เฉพาะตัวเลขเท่านั้น

เมตริกและคิวป์จะทำให้ขนาดในแต่ละมิติเท่ากันเสมอ เช่น

```
> a[1] = 5
5.0
> b[1,2] = 3
3.0
> c[2,3,1] = 1
1.0
```

3.1.5 ตัวเลข

การใช้งาน

ใช้ในการเก็บค่าทุกชนิดที่เป็นตัวเลขรวมถึงจำนวนจริงและจำนวนเชิงซ้อนรองรับการทำงานบนเครื่องหมายทางคณิตศาสตร์ เช่น

```
1
1 + i
```

ทั้งหมดนี้เป็นค่าเชิงตัวเลขทั้งสิ้น

3.1.6 สายอักขระ (สตริง)

การใช้งาน

การกำหนดค่าที่เป็นสตริงนั้นสามารถกำหนดโดยใช้เครื่องหมาย " (double quote) อยู่ที่หัวและท้ายของค่านั้น โดยค่าที่เป็นสตริงนั้นต้องไม่มีเครื่องหมายอักขระพิเศษอื่นๆนอกจากตัวอักษร ตัวเลข หรือการเว้นวรรค

```
"Hello World"
"123"
```

3.1.7 คำสั่งที่เป็นโครงสร้างเงื่อนไข

If

สามารถเขียนได้ 2 รูปแบบดังนี้

```
if boolean-expression then
        statement-sequence;
end if
```

หรือ

```
if boolean-expression then
        statement-sequence1;
else
        statement-sequence2;
end if
```

For

สามารถเขียนได้สองรูปแบบดังนี้

```
for initialization-expression to boolean-
expression do
        statement-sequence;
loop;
```

หรือ

```
For initialization-expression to boolean-
expression step increment-number do
        statement-sequence;
loop;
```

While

สามารถเขียนได้ดังนี้

```
while boolean-expression do
        statement-sequence;
loop
```

3.2 ออกแบบไวยากรณ์ของภาษา

จากข้อกำหนดเราจะได้รูปแบบของไวยากรณ์ซึ่งเป็นไวยากรณ์แบบ Context-free ในรูปแบบของ BNF (พร้อมคำอธิบาย) ดังนี้

กำหนดคำสั่งแต่ละคำสั่งจะประกอบไปด้วยคำสั่งประเภท while statement, for statement, if statement, expression statement หรือ function declaration statement

```
statement ::= while_statement
            | for_statement
            | if_statement
            | expression_statement
            | function_declaration
```

กำหนดคำสั่งที่เป็นลำดับหลายๆคำสั่งจะประกอบไปด้วยคำสั่งเดียว หรือ หลายๆคำสั่ง แต่ละคำสั่งจะจบด้วยเครื่องหมาย ";"

```
statement_sequence ::= statement_sequence statement ';'
                     | statement ';'
```

กำหนด คำสั่งวนรอบแบบ while จะต้องเริ่มต้นด้วยคำว่า "while" ตามด้วยเงื่อนไขแบบตรรกะแล้วตามด้วยคำว่า "do" ตามด้วย คำสั่งที่เป็นลำดับหลายๆคำสั่ง และจบด้วย คำว่า "loop"

```
while_statement ::= 'while' boolean_expression 'do' statement_sequence 'loop'
```

กำหนดคำสั่งวนรอบแบบ for จะต้องเริ่มต้นด้วยคำว่า "for" ตามด้วย การใส่ค่าให้ตัวแปรเพื่อนับจำนวนวนรอบ แล้วตามด้วยคำว่า "to" แล้วตามด้วยจำนวนเพื่อทดสอบการวนลูป แล้วตามด้วยคำว่า "do" จากนั้นตามด้วยคำสั่งที่เป็นลำดับหลายๆคำสั่ง แล้วจบด้วยคำว่า "loop" นอกจากนี้

เราสามารถกำหนดจำนวนการเพิ่มหรือลดได้โดยเพิ่มคำว่า "step" เข้าไปหลังจำนวนที่ ใช้ทดสอบในการวนลูป

```
for_statement ::= 'for' identifier '=' expression_statement 'to' expression_statement
'do' statement_sequence 'loop'
                | 'for' identifier '=' expression_statement 'to' expression_statement
'step' expression_statement 'do' statement_sequence 'loop'
```

กำหนดคำสั่งเงื่อนไข if จะต้องเริ่มต้นด้วยคำว่า "if" ตามด้วย เงื่อนไขทางตรรกะแล้วตามด้วคำว่า "then" แล้วตามด้วยคำสั่งที่เป็นลำดับหลายๆคำสั่ง แล้วจบด้วยคำว่า "end if" หรือเราสามารถใช้คำว่า "else" เพื่อกำหนดการทำงานที่ไม่ตรงเงื่อนไขได้ ซึ่งวางต่อจากคำสั่งที่เป็นลำดับสำหรับเงื่อนไขถูกต้อง แล้วใส่คำสั่งที่เป็นลำดับหลายๆคำสั่งสำหรับเงื่อนไขที่ไม่ตรงเงื่อนไขลงไปแล้วจบด้วย "end if"

```
if_statement ::= 'if' boolean_expression 'then' statement_sequence 'end' 'if'
        | 'if' boolean_expression 'then' statement_sequence 'else'
statement_sequence 'end' 'if'
```

กำหนดคำสั่งสำหรับสร้างฟังก์ชั่นแบ่งเป็น 2 แบบคือ statement function และ expression function ซึ่ง statement function คือการกำหนดฟังก์ชั่นด้วยการให้ฟังก์ชั่นนั้นประกอบไปด้วย การทำงานของ คำสั่งย่อยๆหลายๆคำสั่งส่วน expression function คือฟังก์ชั่นที่ถูกกำหนดโดย expression statement ซึ่งเป็นประโยคทางคณิตศาสตร์นั้นเอง ในการกำหนดฟังก์ชั่นจะกำหนดได้โดยเริ่มจากคำว่า "function" ตามด้วยชื่อฟังก์ชั่น ตามด้วย เครื่องหมาย '(' และ ')' ซึ่งระหว่างเครื่องหมายทั้งสองจะมีการใส่ชื่อตัวแปรที่เป็นพารามิเตอร์ หรือไม่ก็ได้หากใส่ตัวแปรที่รับค่าหลายค่าก็คั่นด้วยเครื่องหมาย ',' หลังจากนั้นหากเป็นแบบ statement function จะตามด้วยคำสั่งย่อยๆหลายๆคำสั่งแล้วจบด้วยคำว่า "end function" แต่ถ้า หากเป็น expression function ก็จะต้องตามด้วยเครื่องหมาย "=" ก่อนแล้วตามด้วย expression statement แล้วค่อยจบด้วย "end function"

```
function_declaration ::= 'function' identifier '=' identifier '(' ')' statement_sequence
'end' 'function'
                | 'function' identifier '=' identifier '(' identifier_list ')'
statement_sequence 'end' 'function'
                | 'function' identifier '(' ')' statement_sequence 'end' 'function'
                | 'function' identifier '(' identifier_list ')' statement_sequence 'end'
'function'
                | 'function' identifier '(' identifier_list ')' '=' expression_statement
'end' 'function'
```

กำหนดรูปแบบของรายการของนิพจน์ให้เป็นนิพจน์เดียวหรือหลายนิพจน์โดยคั่นด้วยเครื่องหมาย ","

```
expression_list ::= expression_statement
        | expression_list ',' expression_statement
```

กำหนดรูปแบบของรายการของตัวเลขให้เป็นตัวเดียวหรือหลายๆตัวโดยคั่นด้วยเครื่องหมาย ","

```
number_list ::= number
                | number_list ',' number
```

กำหนดรูปแบบของรายการของชื่อตัวแปรให้เป็นชื่อเดียวหรือหลายชื่อโดยคั่นด้วยเครื่องหมาย ","

```
identifier_list ::= identifier
                | identifier_list ',' identifier
```

กำหนดรูปแบบของนิพจน์ซึ่งประกอบไปด้วยรูปแบบต่างๆดังนี้

- นิพจน์ที่เป็นตัวเลข หรือตัวแปร หรือ ข้อความ หรือการเรียกใช้งานฟังก์ชั่น
- นิพจน์ทางคณิตศาสตร์ได้แก่ การใช้เครื่องหมายวงเล็บเพื่อกำหนดลำดับการคำนวณ และ

- การคำนวณทางคณิตศาสตร์ต่างๆได้แก่ “+”, “-“, “*”, “/”, “%”, “^”,
- และนิพจน์ที่เป็นการใส่ค่าให้กับตัวแปร

```
expression_statement ::= number
                       | identifier
                       | string
                       | identifier '(' ')'
                       | identifier '(' expression_list ')'
                       | '(' expression_statement ')'
                       | '-' expression_statement
                       | expression_statement '+' expression_statement
                       | expression_statement '-' expression_statement
                       | expression_statement '*' expression_statement
                       | expression_statement '/' expression_statement
                       | expression_statement '%' expression_statement
                       | expression_statement '^' expression_statement
                       | identifier '=' expression_statement
                       | identifier '=' collection_initializer
```

กำหนดรูปแบบของนิพจน์ทางตรรกะประกอบไปด้วย

- Boolean logical operator ได้แก่ “&&” และ “||” หมายถึง และ, หรือ ตามลำดับ ซึ่งจะถูกใช้ร่วมกับนิพจน์ทางตรรกะด้วยกันเท่านั้นไม่สามารถนำไปใช้ร่วมกับข้อมูลชนิดอื่นได้
- Relational Operator ได้แก่ “==”, “!=”, “>”, “<”, “>=”, “<=” หมายถึง เท่ากับ ไม่เท่ากับ น้อยกว่า มากกว่า มากกว่าหรือเท่ากับ และ น้อยกว่าหรือเท่ากับ ตามลำดับ

```
boolean_expression ::= '(' boolean_expression ')'
                     | boolean_expression '&&' boolean_expression
                     | boolean  expression '||' boolean_expression
                     | expression_statement '==' expression_statement
```

```
| expression_statement '!=' expression_statement
| expression_statement '>' expression_statement
| expression_statement '<' expression_statement
| expression_statement '>=' expression_statement
| expression_statement '<=' expression_statement
```

กำหนดรูปแบบของการใส่ค่าคอลเลคชั่นแบบต่างๆในลักษณะเป็นโครงสร้างได้แก่ ข้อมูลแบบอะเรย์ ข้อมูลแบบเมตริกซ์และข้อมูลแบบคิวบ์

```
collection_initializer ::= array_initializer
                         | matrix_initializer
                         | cube_initializer
```

กำหนดรูปแบบของการใส่ค่าคอลเลคชั่นแบบอะเรย์โดยให้มีเครื่องหมาย “{“ และ “}” ครอบและตามด้วยรายการของตัวเลข

```
array_initializer ::= '{' number_list '}'
```

กำหนดรูปแบบของรายการของตัวเลขต้องเป็นตัวเลขที่มีการคั่นด้วยเครื่องหมาย “,”

```
array_initializer_list ::= array_initializer
                         | array_initializer_list ',' array_initializer
```

กำหนดรูปแบบของการใส่ค่าคอลเลคชั่นแบบเมตริกซ์ ซึ่งก็คือรายการของอะเรย์ที่ครอบด้วยเครื่องหมาย “{“ และ “}”

```
matrix_initializer ::= '{' array_initializer_list '}'
```

กำหนดรูปแบบของรายการเมตริกซ์ ซึ่งก็คือคอลเลคชั่นแบบเมตริกซ์ครอบเครื่องหมาย “{“ และ “}” หรือคอลเลคชั่นแบบเมตริกซ์หลายๆตัวขั้นด้วยเครื่องหมาย “,” และครอบด้วยเครื่องหมาย “{“ และ “}”

```
matrix_initializer_list ::= '{' matrix_initializer '}'
                          | '{' matrix_initializer_list ',' matrix_initializer '}'
```

กำหนดรูปแบบของการใส่ค่าคอลเลคชั่นแบบคิวบ์ ซึ่งก็คือรายการของเมตริกซ์ที่ครอบด้วยเครื่องหมาย “{“และ “}”

```
cube_initializer ::= '{' matrix_initializer_list '}'
```

กำหนดรูปแบบของชื่อตัวแปร (identifier) ซึ่งประก่อบไปด้วยการอ้างชื่อตัวแปรปกติ การอ้างชื่อตัวแปรที่เป็นชนิดข้อมูลอะเรย์ การอ้างชื่อตัวแปรที่เป็นชนิดข้อมูลเมตริกซ์และการอ้างชื่อตัวแปรที่เป็นชนิดข้อมูลคิวบ์

```
identifier ::= ID
             | ID '[' expression_statement ']'
             | ID '[' expression_statement ',' expression_statement ']'
             | ID '[' expression_statement ',' expression_statement ','
expression_statement ']'
```

กำหนดรูปแบบของชนิดของตัวเลขมีสอบแบบคือ จำนวนจริงและจำนวนเชิงซ้อน

```
number ::= REAL
         | COMPLEX
```

กำหนดรูปแบบของชนิดข้อมูลสายอักขระ

```
string ::= STRING
```

3.3 ออกแบบข้อกำหนดของตัววิเคราะห์คำ (สแกนเนอร์) และข้อกำหนดของตัววิเคราะห์ไวยากรณ์ (พาร์สเซอร์) และสร้างคลาสจากข้อกำหนดทั้งสอง

เมื่อได้ไวยากรณ์มาแล้วก็ทำการออกแบบข้อกำหนดของตัววิเคราะห์คำ (สแกนเนอร์) และและข้อกำหนดของตัววิเคราะห์ไวยากรณ์ (พาร์สเซอร์) และสร้างคลาสจากข้อกำหนดทั้งสอง

จากไวยากรณ์ที่ออกแบบข้างต้นเรากำหนด Terminal symbol ได้ดังนี้

Terminal	**Token**	**Terminal**	**Token**
WHILE	while	DO	do
LOOP	loop	END	end
IF	if	THEN	then
ELSE	else	FUNCTION	function
TO	to	FOR	for
STEP	step	SEMICOL	;
COMMA	,	LPAR	(
RPAR	)	LBRC	{
RBRC	}	LBRK	[
RBRK	]	ASSIGN	=
EQ	==	NEQ	!=
LE	<	LEQ	<=
GE	>	GEQ	>=
MINUS	-	PLUS	+
TIMES	*	DIV	/
MOD	%	POW	^
AND	&&	OR	||
ID	ชื่อของตัวแปร	REAL	ตัวเลขจำนวนจริง
COMPLEX	จำนวนเชิงซ้อน	STRING	ข้อมูลสายอักขระ

ตารางที่ 3.1 แสดง Terminal symbol ที่ได้จากการออกแบบไวยากรณ์ภาษา

และจะได้ Non-terminal และคลาสที่จะนำมารองรับการทำงานดังนี้

Non-terminal	Class ที่รองรับ	Non-terminal	Class ที่รองรับ
statement	TStatement	statement_sequence	TStatementSequence
while_statement	TWhileStatement	for_statement	TForStatement
if_statement	TIfStatement	function_declaration	TFunction
expression_list	TExpressionList	identifier_list	TIdentifierList
expression_statement	TExpresstion	boolean_expression	TBooleanExpression
collection_initializer	TCollectionInitializer	array_ initializer	TArrayInitializer
array_ initializer _list	TArrayInitializerList	matrix_ initializer	TMatrixInitializer
matrix_ initializer_list	TMatrixInitializerList	cube_initializer	TCubeInitializer
identifier	TIdentifer	number	TNumber
string	TString		

ตารางที่ 3.2 แสดง Non-terminal symbol ที่ได้จากการออกแบบไวยากรณ์ภาษา

สำหรับข้อกำหนดของตัววิเคราะห์คำสำหรับโปรแกรม JFlex จะเป็นดังนี้

```
import java_cup.runtime.Symbol;
%%
%public                 // กำหนดให้สแกนเนอร์ที่จะสร้างประกาศเป็น public
%cup                    // กำหนดให้สแกนเนอร์ที่จะสร้างทำงานร่วมกับ CUP
%implements sym         // กำหนดให้สแกนเนอร์ที่จะสร้างอิมพลีเมนต์คลาส sym ของ CUP
%unicode                // กำหนดให้สแกนเนอร์ที่จะสร้างรองรับตัวอักษรแบบ unicode

/* ในส่วนของ user code จะเป็นการใส่โคดเพื่อจัดการในเรื่องของ Symbol table */
%{
 SymbolTable symbolTable;        // ประกาศตัวอ้างอิงถึง Symbol table
```

ประกาศเมธอดสำหรับติดตั้ง Symbol table เข้ามา

```
public void setSymbolTable(SymbolTable symbolTable) {
  this.symbolTable = symbolTable;
}
```

ประกาศเมธอดสำหรับคืนค่าชนิดของโทเคนที่พบ

```
private Symbol symbol(int symbolID) {
  return new Symbol(symbolID);
}
```

ประกาศเมธอดสำหรับคืนค่าชนิดของโทเคนที่พบและค่าที่สแกนมาได้

```
private Symbol symbol(int symbolID, Object value) {
   return new Symbol(symbolID, value);
  }
%}

/* กำหนดรูปแบบของ Identifier หรือ ชื่อตัวแปร */
Identifier = [:jletter:][:jletterdigit:]*

/* กำหนดรูปแบบของจำนวนเต็ม */
IntegerLiteral = 0 | [1-9][0-9]*

/* กำหนดรูปแบบของจำนวนจริง */
RealLiteral = ({FloatLiteral1} | {FloatLiteral2}) {Exponent}?
FloatLiteral1 = {IntegerLiteral} \. [0-9]*
FloatLiteral2 = [0-9]+
Exponent = [eE] [+-]? [0-9]+

/* กำหนดรูปแบบของจำนวนเชิงซ้อน */
ComplexLiteral = {RealLiteral} i

/* กำหนดรูปแบบของสายอักขระ */
StringCharacter = [^\r\n\"\\]

%%
```

/* กำหนดการคืนค่าโทเคนคีย์เวิร์ดที่สแกนพบ */

Token	Symbol	Token	Symbol
"while"	WHILE	"do"	DO
"loop"	LOOP	"end"	END
"if"	IF	"then"	THEN
"else"	ELSE	"function"	FUNCTION
"for"	FOR	"to"	TO
"step"	STEP		

/* กำหนดการคืนค่าชื่อตัวแปรที่สแกนพบ */

```
{Identifier}            { symbolTable.enter(yytext(), new SymbolTableEntry(yytext()));
                          return symbol(ID, yytext()); }
```

/* กำหนดการคืนค่าโทเคนจำนวนจริงที่สแกนพบ */

```
{RealLiteral}           { return symbol(REAL, yytext()); }
```

/* กำหนดการคืนค่าโทเคนจำนวนเชิงซ้อนที่สแกนพบ */

```
{ComplexLiteral}        { return symbol(COMPLEX, yytext()); }
```

/* กำหนดการคืนค่าโทเคนสายอักขระที่สแกนพบ */

```
\" {StringCharacter}* \"  { return symbol(STRING, yytext()); }
```

/* กำหนดการคืนค่าโทเคนเครื่องหมายต่างๆที่สแกนพบ */

Token	Symbol	Token	Symbol
","	COMMA	"("	LPAR
")"	RPAR	"{"	LBRC
"}"	RBRC	"["	LBRK
"]"	RBRK	"="	ASSIGN
"=="	EQ	"!="	NEQ
"<"	LE	"<="	LEQ
">"	GE	">="	GEQ
"-"	MINUS	"+"	PLUS
"*"	TIMES	"/"	DIV
"%"	MOD	"&&"	AND
"^"	POW	"||"	OR
";"	SEMICOL		

(สามารถดูซอร์สโค้ดที่สมบูรณ์ในส่วนของกฎในการตัดคำได้ในส่วนของภาคผนวก ค.1) และสำหรับข้อกำหนดไวยากรณ์สำหรับโปรแกรม CUP จะได้ดังนี้

```
terminal WHILE, DO, LOOP, END, IF, THEN, ELSE, FUNCTION, TO, FOR, STEP;
terminal SEMICOL, COMMA;
terminal LPAR, RPAR, LBRC, RBRC, LBRK, RBRK;
terminal ASSIGN, EQ, NEQ, LE, LEQ, GE, GEQ;
terminal MINUS, PLUS, TIMES, DIV, MOD, POW;
terminal AND, OR;
terminal UMINUS;

terminal String ID, REAL, COMPLEX;
terminal String STRING;

non terminal TStatement statement;
non terminal TStatementSequence statement_sequence;
non terminal TWhileStatement while_statement;
non terminal TForStatement for_statement;
non terminal TIfStatement if_statement;
non terminal TFunction function_declaration;
non terminal TExpressionList expression_list;
non terminal TNumberList number_list;
non terminal TIdentifierList identifier_list;
non terminal TExpression expression_statement;
non terminal TBooleanExpression boolean_expression;
non terminal TCollectionInitializer collection_initializer;
non terminal TArrayInitializer array_initializer;
non terminal TArrayInitializerList array_initializer_list;
non terminal TMatrixInitializer matrix_initializer;
```

```
non terminal TMatrixInitializerList matrix_initializer_list;
non terminal TCubeInitializer cube_initializer;
non terminal TIdentifier identifier;
non terminal TNumber number;
non terminal TString string;
```

precedence, left associatively

```
precedence left ASSIGN;
precedence left AND, OR;
precedence left EQ, NEQ;
precedence left LE, LEQ, GE, GEQ;
precedence left MINUS, PLUS;
precedence left TIMES, DIV, MOD;
precedence left UMINUS;
precedence right POW;
```

Grammar rules

```
statement ::= while_statement:whileStmt
              {: RESULT = new TStatement(whileStmt); :}
              | for_statement:forStmt
              {: RESULT = new TStatement(forStmt); :}
              | if_statement:ifStmt
              {: RESULT = new TStatement(ifStmt); :}
              | expression_statement:expStmt
              {: RESULT = new TStatement(expStmt); :}
              | function_declaration:funcDecl
              {: RESULT = new TStatement(funcDecl); :}
              ;
```

```
statement_sequence ::= statement_sequence:stmtSeq statement:stmt SEMICOL
                {: RESULT = new TStatementSequence(stmtSeq, stmt); :}
                | statement:stmt SEMICOL
                {: RESULT = new TStatementSequence(stmt); :}
                ;

while_statement ::= WHILE boolean_expression:booleanExp DO statement_sequence:stmtSeq
LOOP
                {: RESULT = new TWhileStatement(booleanExp, stmtSeq); :}
                ;

for_statement ::= FOR identifier:id ASSIGN expression_statement:expStmt TO
expression_statement:expStmt2 DO statement_sequence:stmtSeq LOOP
                {: RESULT = new TForStatement(id, expStmt, expStmt2, stmtSeq); :}
                | FOR identifier:id ASSIGN expression_statement:expStmt TO
expression_statement:expStmt2 STEP expression_statement:expStmt3 DO
statement_sequence:stmtSeq LOOP
                {: RESULT = new TForStatement(id, expStmt, expStmt2, expStmt3, stmtSeq);
:}
                ;

if_statement ::= IF boolean_expression:booleanExp THEN statement_sequence:stmt END IF
                {: RESULT = new TIfStatement(booleanExp, stmt); :}
                | IF boolean_expression:booleanExp THEN statement_sequence:stmt1 ELSE
statement_sequence:stmt2 END IF
                {: RESULT = new TIfStatement(booleanExp, stmt1, stmt2); :}
                ;

function_declaration ::= FUNCTION identifier:returnName ASSIGN identifier:funcName
LPAR RPAR statement_sequence:stmtSeq END FUNCTION
                {: RESULT = new TFunction(returnName, funcName, null, stmtSeq); :}
```

```
                | FUNCTION identifier:returnName ASSIGN identifier:funcName LPAR
identifier_list:idList RPAR statement_sequence:stmtSeq END FUNCTION
                {: RESULT = new TFunction(returnName, funcName, idList, stmtSeq); :}

                | FUNCTION identifier:funcName LPAR RPAR ASSIGN
expression_statement:expStmt END FUNCTION
                {: RESULT = new TFunction(funcName, null, expStmt); :}

                | FUNCTION identifier:funcName LPAR identifier_list:idList RPAR ASSIGN
expression_statement:expStmt END FUNCTION
                {: RESULT = new TFunction(funcName, idList, expStmt); :}

                | FUNCTION LBRC identifier_list:idList RBRC ASSIGN identifier:funcName
LPAR RPAR statement_sequence:stmtSeq END FUNCTION
                {: RESULT = new TFunction(idList, funcName, null, stmtSeq); :}

                | FUNCTION LBRC identifier_list:idList RBRC ASSIGN identifier:funcName
LPAR identifier_list:idList2 RPAR statement_sequence:stmtSeq END FUNCTION
                {: RESULT = new TFunction(idList, funcName, idList2, stmtSeq); :}
                ;

expression_list ::= expression_statement:expStmt
                {: RESULT = new TExpressionList(expStmt); :}
            | expression_list:expList COMMA expression_statement:expStmt
                {: RESULT = new TExpressionList(expList, expStmt); :}
            ;
number_list ::= number:num
                {: RESULT = new TNumberList(num); :}
                | number_list:numList COMMA number:num
                {: RESULT = new TNumberList(numList, num); :}
                ;
```

```
identifier_list ::= identifier:id
                    {: RESULT = new TIdentifierList(id); :}
                    | identifier_list:idList COMMA identifier:id
                    {: RESULT = new TIdentifierList(idList, id); :}
                    ;

expression_statement ::= number:num
                    {: RESULT = num; :}
                    | identifier:id
                    {: RESULT = id; :}
                    | string:str
                    {: RESULT = str; :}
                    | identifier:id LPAR RPAR
                    {: RESULT = new TFunctionCall(id); :}
                    | identifier:id LPAR expression_list:expList RPAR
                    {: RESULT = new TFunctionCall(id, expList); :}
                    | LPAR expression_statement:expStmt RPAR
                    {: RESULT = expStmt; :}
                    | MINUS expression_statement:expStmt
                    {: RESULT = new TExpressionPrefix(expStmt, "-"); :} %prec UMINUS
                    | expression_statement:leftExp PLUS expression_statement:rightExp
                    {: RESULT = new TExpressionInfix(leftExp, "+", rightExp); :}
                    | expression_statement:leftExp MINUS expression_statement:rightExp
                    {: RESULT = new TExpressionInfix(leftExp, "-", rightExp); :}
                    | expression_statement:leftExp TIMES expression_statement:rightExp
                    {: RESULT = new TExpressionInfix(leftExp, "*", rightExp); :}
                    | expression_statement:leftExp DIV expression_statement:rightExp
                    {: RESULT = new TExpressionInfix(leftExp, "/", rightExp); :}
                    | expression_statement:leftExp MOD expression_statement:rightExp
                    {: RESULT = new TExpressionInfix(leftExp, "%", rightExp); :}
                    | expression_statement:leftExp POW expression_statement:rightExp
```

```
{: RESULT = new TExpressionInfix(leftExp, "^", rightExp); :}
                | identifier:id ASSIGN expression_statement:expStmt
                {: RESULT = new TAssignment(id, expStmt); :}
                | identifier:id ASSIGN collection_initializer:collectionInit
                {: RESULT = new TAssignment(id, collectionInit); :}
                ;

boolean_expression ::= LPAR boolean_expression:boolExp RPAR
                {: RESULT = boolExp; :}
                | boolean_expression:leftBoolExp AND boolean_expression:rightBoolExp
                {: RESULT = new TLogicalBoolean(leftBoolExp, "&&", rightBoolExp); :}
                | boolean_expression:leftBoolExp OR boolean_expression:rightBoolExp
                {: RESULT = new TLogicalBoolean(leftBoolExp, "||", rightBoolExp); :}
                | expression_statement:leftExp EQ expression_statement:rightExp
                {: RESULT = new TRelationalBoolean(leftExp, "==", rightExp); :}
                | expression_statement:leftExp NEQ expression_statement:rightExp
                {: RESULT = new TRelationalBoolean(leftExp, "!=", rightExp); :}
                | expression_statement:leftExp GE expression_statement:rightExp
                {: RESULT = new TRelationalBoolean(leftExp, ">", rightExp); :}
                | expression_statement:leftExp LE expression_statement:rightExp
                {: RESULT = new TRelationalBoolean(leftExp, "<", rightExp); :}
                | expression_statement:leftExp GEQ expression_statement:rightExp
                {: RESULT = new TRelationalBoolean(leftExp, ">=", rightExp); :}
                | expression_statement:leftExp LEQ expression_statement:rightExp
                {: RESULT = new TRelationalBoolean(leftExp, "<=", rightExp); :}
                ;

collection_initializer ::  array_initializer:arrayInit
                {: RESULT = arrayInit; :}
                | matrix_initializer:matInit
                {: RESULT = matInit; :}
                | cube_initializer:cubeInit
```

```
                {: RESULT = cubeInit; :}
                ;

array_initializer ::= LBRC number_list:numList RBRC
                {: RESULT = new TArrayInitializer(numList); :}
                ;

array_initializer_list ::= array_initializer:arrayInit
                {: RESULT = new TArrayInitializerList(arrayInit); :}
                | array_initializer_list:arrayInitList COMMA array_initializer:arrayInit
                {: RESULT = new TArrayInitializerList(arrayInitList, arrayInit); :}
                ;

matrix_initializer ::= LBRC array_initializer_list:arrayInitList RBRC
                {: RESULT = new TMatrixInitializer(arrayInitList); :}
                ;

matrix_initializer_list ::= LBRC matrix_initializer:matInit RBRC
                {: RESULT = new TMatrixInitializerList(matInit); :}
                | LBRC matrix_initializer_list:matInitList COMMA matrix_initializer:matInit
RBRC
                {: RESULT = new TMatrixInitializerList(matInitList, matInit); :}
                ;

cube_initializer ::= LBRC matrix_initializer_list:matInitList RBRC
                {: RESULT = new TCubeInitializer(matInitList); :}
                ;

identifier ::= ID:id
                {: RESULT = new TIdentifier(id); :}
                | ID:id LBRK expression_statement:expStmt RBRK
```

```
                {: RESULT = new TIdentifier(id, expStmt); :}
                | ID:id LBRK expression_statement:expStmt1 COMMA
expression_statement:expStmt2 RBRK
                {: RESULT = new TIdentifier(id, expStmt1, expStmt2); :}
                | ID:id LBRK expression_statement:expStmt1 COMMA
expression_statement:expStmt2 COMMA expression_statement:expStmt3 RBRK
                {: RESULT = new TIdentifier(id, expStmt1, expStmt2, expStmt3); :}
                ;

number ::= REAL:real
                {: RESULT = new TNumber(real, false); :}
                | COMPLEX:complex
                {: RESULT = new TNumber(complex, true); :}
                ;

string ::= STRING:str
                {: RESULT = new TString(str); :}
                ;
```

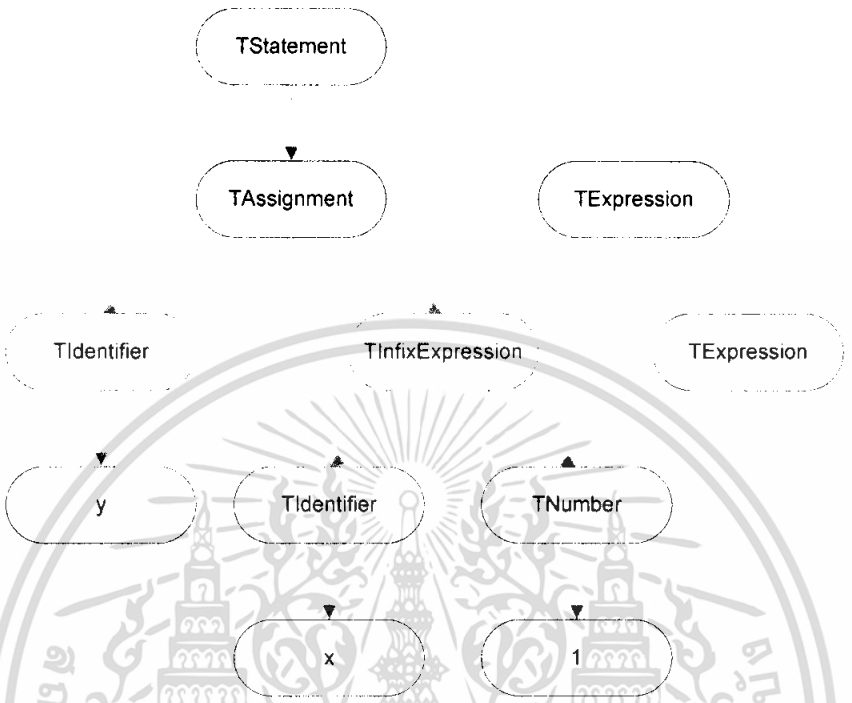
ซึ่งจะเห็นได้ว่าเราจะไม่ได้ทำการใส่ user code ในคำนวณโดยตรงแต่เราจะส่งแต่ละส่วนให้แก่คลาสที่กำหนดหน้าที่ไว้แล้วให้จัดการในแต่ละส่วนผลที่ได้จากการพาร์สของข้อกำหนดนี้ก็จะได้เป็นคลาสที่มีโครงสร้างเป็นต้นไม้เพื่อทำการแปลความหมายต่อไปซึ่งรายละเอียดการทำงานจะอธิบายในหัวข้อถัดไป (สามารถดูส่วนของซอร์สโค้ดของหลักไวยากรณ์โครงสร้างของภาษาที่สมบูรณ์ได้ในส่วนของภาคผนวก ค.2)

3.4 สร้างตัวแปลภาษาจากข้อกำหนดและสร้างคลาสเพื่อรองรับการทำงานตัวแปลภาษา

หลังจากที่เราทำการกำไวยากรณ์และข้อกำหนดต่างๆแล้วเราก็นำข้อกำหนดที่ได้ไปให้โปรแกรม JFlex สร้างสแกนเนอร์ให้และให้ CUP สร้างพาร์เซอร์ให้ซึ่งจะได้ไฟล์มาสามไฟล์ คือ Ylex.java ซึ่งเป็นซอร์สโคดไฟล์ของสแกนเนอร์ parser.java เป็นไฟล์ซอร์สโคดของพาร์เซอร์และไฟล์ sym.java ซึ่งเป็นไฟล์ที่เก็บ ชนิดของโทเคนต่างๆไว้จะถูกใช้งานทั้งใน สแกนเนอร์และพาร์

เซอร์สิ่งถัดไปที่จะต้องทำคือต้องสร้างคลาสเพื่อรองรับการทำงานสำหรับ แต่ละโหนดของทรีที่ได้จากการพาร์ส

ตัวอย่างพาร์สทรีที่ได้จากการพาร์สประโยค y = x + 1 เป็นดังนี้



ภาพที่ 3.1 แสดงพาร์สทรีที่ได้จากการพาร์สประโยค y = x + 1

เมื่อเราได้พาร์สทรีแล้วเราจะนำทรีนี้ไปทำงานเพื่อแสดงผลโดยการท่องไปในทรีซึ่งจากตัวอย่างจะได้ลำดับการคำนวณดังนี้

1. สั่งให้ TStatement ทำงาน
2. TStatement เรียก TAssignment ทำงานซึ่งถือว่าเป็น TExpression ตัวหนึ่ง
3. จากนั้น TAssignement จะทำงานดังนี้

    3.1 TAssignment เรียก TIdentifier ให้ทำงาน และ TIdentifier จะคืนการอ้างอิงตัวแปร y มาให้ TAssignment

    3.2 TAssignment จะเรียก TInfixExpression จากนั้น TInfixExpression จะทำงานดังนี้

        3.2.1 เรียก Tidentifier ให้ทำงานแล้วจะคืนค่าอ้างอิงของตัวแปร x มาให้ TInfixExpression

        3.2.2 เรียก TNumber ให้ทำงานแล้ว TNumber จะคืนค่า 1 ซึ่งเป็นข้อมูลชนิดจำนวนจริงมาให้ TInfixExpression

4. จากนั้น TInfixExpression จะนำค่าของ x และค่าของ TNumber ซึ่งเป็น 1 มาบวกกัน แล้วผลลัพธ์จะคืนไปให้กับ TAssignment

5. TAssignment ทำการนำผลลัพธ์ที่ได้ไปเก็บไว้ในตัวแปร y

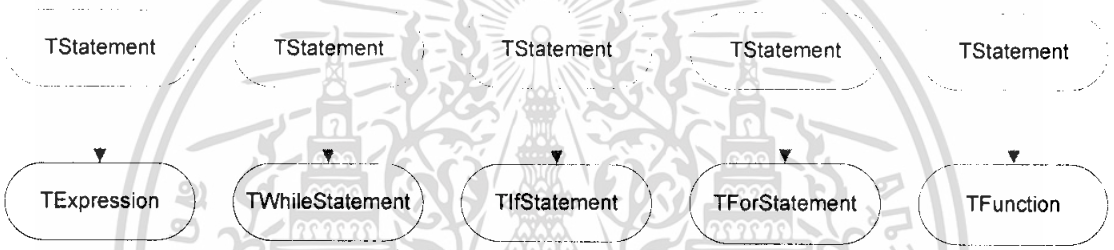
6. TAssignment คืนผลลัพธ์ไปให้ TStatement

7. นำผลลัพธ์จาก TStatement ไปแสดงผล

จะเห็นได้ว่าแต่ละคลาสทำหน้าที่ของตนเองและบางคลาสต้องอาศัยคลาสอื่นในการทำการประมวลผลในการเขียนโปรแกรมเราสามารถเขียนบนพื้นฐานของโครงสร้างข้อมูลแบบต้นไม้ได้โดยง่าย โดยที่ความซับซ้อนจะไปอยู่ในกระบวนการภายในคลาสแต่ละตัวแทน นอกจากนี้คลาสบางตัวรองรับโครงสร้างต้นไม้หลายแบบด้วย ซึ่งแต่ละคลาสจะมีโครงสร้างและรูปแบบการทำงานดังนี้

- TStatement

มีโครงสร้างอย่างใดอย่างหนึ่งดังนี้



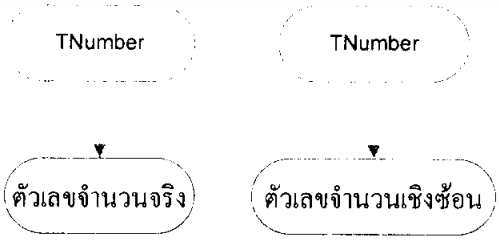
ภาพที่ 3.2 แสดงโครงสร้าง TStatement

ซึ่งการทำงานนั้นเพียงแค่ทำการเรียกคลาสใดคลาสหนึ่งทำงานและส่งผลลัพธ์ออกไปเท่านั้น

TExpression ประกอบไปด้วยโครงสร้างประเภทย่อยๆดังนี้

- TNumber

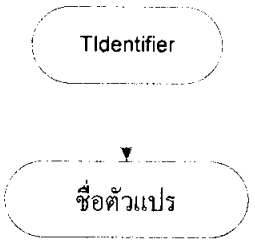
ทำหน้าที่แปลงค่าในโทเคนที่ได้เป็นตัวเลขซึ่งจะแปลงได้สองแบบคือจำนวนจริงหรือจำนวนเชิงซ้อน โดยมีโครงสร้างดังนี้



ภาพที่ 3.3 แสดงโครงสร้าง TNumber

-TIdentifier

ทำหน้าที่ค้นหาตัวแปรที่เก็บไว้ใน Symbol table เพื่อส่งตัวอ้างอิงไปให้ส่วนอื่นๆใช้งาน โดยมีโครงสร้างดังนี้



ภาพที่ 3.4 แสดงโครงสร้าง TIdentifier

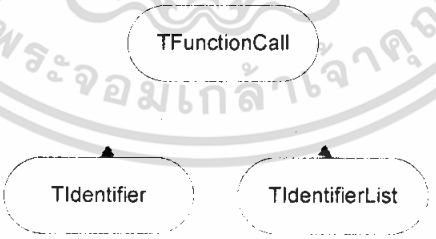
-TString

ทำหน้าที่แปลงโทเคนที่ได้เป็นข้อมูลสายอักขระให้ทำงานได้กับระบบ โดยมีโครงสร้างดังนี้

ภาพที่ 3.5 แสดงโครงสร้าง TString

-TFunctionCall

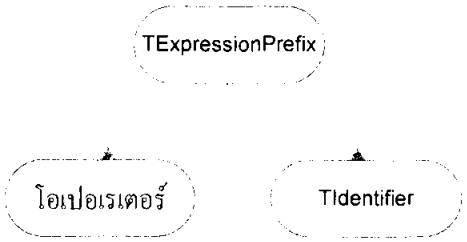
ทำหน้าที่เป็นตัวเรียกฟังก์ชั่นซึ่งฟังก์ชั่นแล้วคืนค่าออกมา โดยมีโครงสร้างดังนี้



ภาพที่ 3.6 แสดงโครงสร้าง TFunctionCall

-TExpressionPrefix

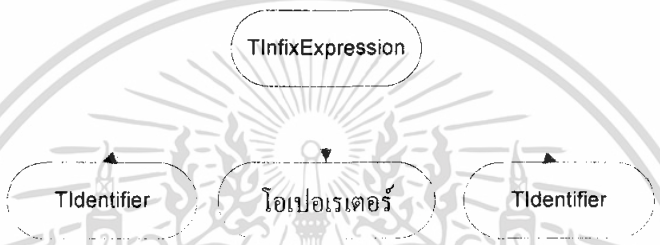
ทำหน้าที่คำนวณนิพจน์ทางคณิตศาสตร์แบบ Prefix



ภาพที่ 3.7 แสดงโครงสร้าง TExpressionPrefix

-TInfixExpression

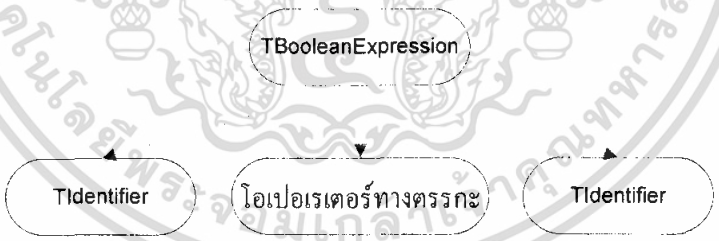
ทำหน้าที่คำนวณนิพจน์ทางคณิตศาสตร์แบบ Infix ซึ่งคือรูปแบบทั่วไปนั่นเอง โดยมีโครงสร้างดังนี้



ภาพที่ 3.8 แสดงโครงสร้าง TInfixExpression

-TBooleanExpression

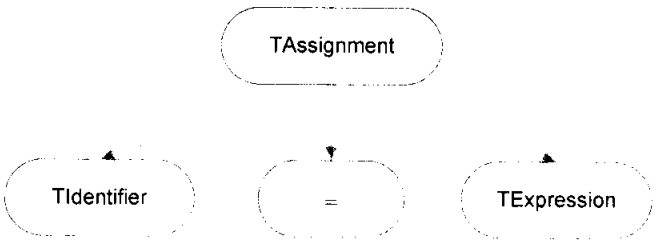
ทำหน้าที่คำนวณนิพจน์ทางตรรกะ โดยมีโครงสร้างดังนี้



ภาพที่ 3.9 แสดงโครงสร้าง TBooleanExpression

-TAssignment

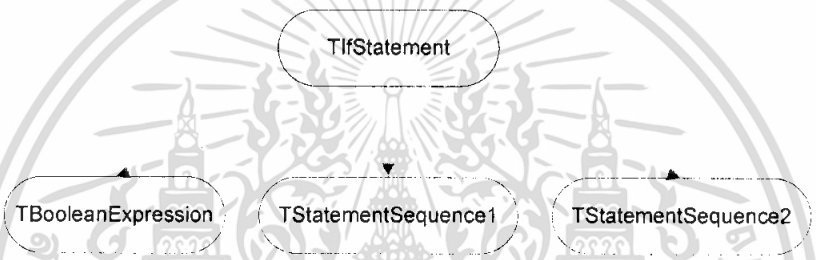
ทำหน้าที่นำค่าทางด้านซ้ายมือ(R-Value) ไปเก็บไว้ในตัวแปรทางด้านขวามือ (L-Value) โดยมีโครงสร้างดังนี้



ภาพที่ 3.10 แสดงโครงสร้าง TAssignment

-TIfStatement

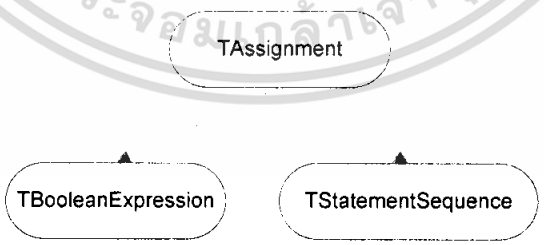
ทำหน้าที่ทดสอบเงื่อนไขจากนิพจน์จากคลาสที่เป็น TBooleanExpression หากค่าที่ได้เป็นจริงก็ให้เข้าไปทำงานในคลาสที่เป็น TStatementSequence หากไม่เป็นจริงก็จะไม่ทำงานหรือถ้าหากเป็นโครงสร้างที่มี else ด้วยก็ให้ไปทำงาน TStatementSequence ที่สอง โดยมีโครงสร้างดังนี้



ภาพที่ 3.11 แสดงโครงสร้าง TIfStatement

-TWhileStatement

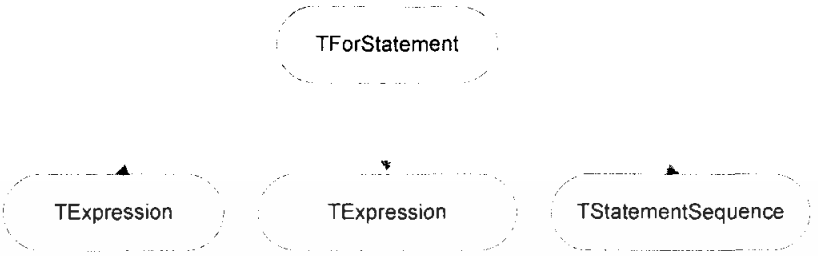
ทำหน้าที่ทำงานแบบวนซ้ำโดยจะต้องมีการทำการทดสอบเงื่อนไขนิพจน์จากคลาสที่เป็น TBooleanExpression หากค่าที่ได้เป็นจริงก็จะเข้าไปทำงานในคลาสที่เป็น TStatementSequence ทำซ้ำไปเรื่อยๆจนกว่าค่าที่ได้จากการทดสอบเงื่อนไขจะเป็นเท็จ



ภาพที่ 3.12 แสดงโครงสร้าง TWhileStatement

-TForStatement

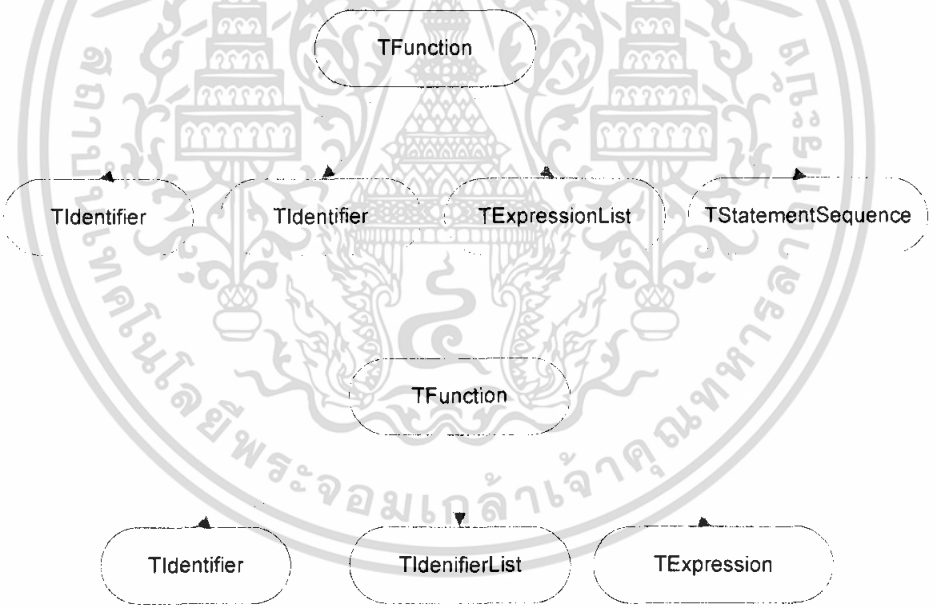
ทำหน้าที่ทำงานแบบวนซ้ำที่มีการนับเป็นลำดับโดยจะต้องมีการกำหนดค่าตัวแปรในการนับก่อนจากนั้นก็จะเป็นการกำหนดค่าสุดท้ายที่ทำงานก่อนหยุดการวนซ้ำซึ่งในระหว่างการวนซ้ำก็จะทำการเรียกคลาส TStatementSequence ให้ทำงานจนกว่าจะจบ



ภาพที่ 3.13 แสดงโครงสร้าง TForStatement

-TFunction

ทำหน้าที่ในการเก็บฟังก์ชั่นที่ผู้ใช้งานโปรแกรมกำหนดขึ้น โดยมีอยู่สองรูปแบบดังนี้

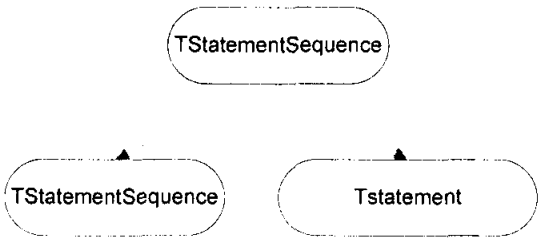


ภาพที่ 3.14 แสดงโครงสร้าง TFunction

ส่วนคลาสที่เป็นส่วนประกอบของคลาสอื่นๆได้แก่

-TStatementSequence

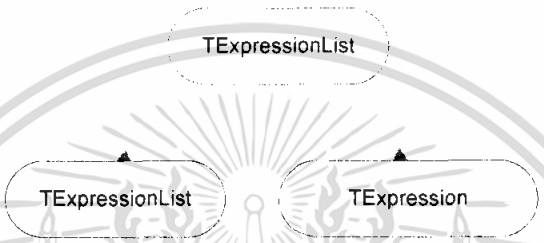
ทำหน้าที่ประมวลผลคลาส TStatement หลายๆตัวแล้วเก็บผลลัพธ์ไว้แสดง โดยมีโครงสร้างดังนี้



ภาพที่ 3.15 แสดงโครงสร้าง TStatementSequence

-TExpressionList

ทำหน้าประมวลผลคลาสที่อยู่ในหมวด TExpression หลายๆตัวแล้วเก็บรายการของผลลัพธ์ไว้



ภาพที่ 3.16 แสดงโครงสร้าง TExpressionList

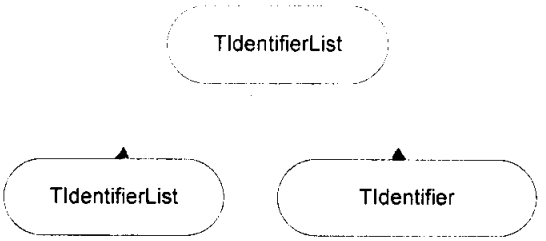
-TNumberList

ทำหน้าประมวลผลคลาสที่อยู่ในหมวด TNumber หลายๆตัวแล้วเก็บรายการของผลลัพธ์ไว้ โดยมีโครงสร้างดังนี้

ภาพที่ 3.17 แสดงโครงสร้าง TNumberList

-TIdentifierList

ทำหน้าประมวลผลคลาสที่อยู่ในหมวด TIdentifier หลายๆตัวแล้วเก็บรายการของผลลัพธ์ไว้ โดยมีโครงสร้างดังนี้



ภาพที่ 3.18 แสดงโครงสร้าง TIdentifierList

-TCollectionInitializer

ทำหน้าที่เป็นตัวใส่ค่าให้แก่ข้อมูลประเภทคอลเลคชั่น ประกอบไปด้วยโครงสร้างดังนี้

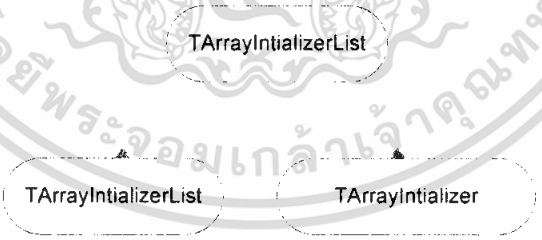
-TArrayInitializer

ทำหน้าที่เป็นตัวใส่ค่าให้แก่ข้อมูลคอลเลคชั่นแบบอะเรย์ ประกอบไปด้วยโครงสร้างดังนี้

ภาพที่ 3.19 แสดงโครงสร้าง TArrayInitializer

-TArrayInitializerList

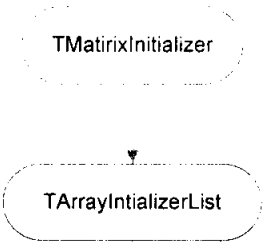
ทำหน้าที่เก็บรายการของอะเรย์ประกอบไปด้วยโครงสร้างดังนี้



ภาพที่ 3.20 แสดงโครงสร้าง TArrayInitializer

-TMatrixInitializer

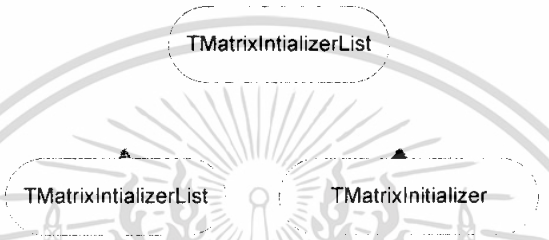
ทำหน้าที่เป็นตัวใส่ค่าให้แก่ข้อมูลคอลเลคชั่นแบบเมตริกซ์ ประกอบไปด้วยโครงสร้างดังนี้



ภาพที่ 3.21 แสดงโครงสร้าง TMatrixInitializer

-TMatrixInitializerList

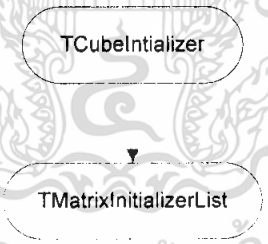
ทำหน้าที่เก็บรายการของเมตริกซ์ประกอบไปด้วยโครงสร้างดังนี้



ภาพที่ 3.22 แสดงโครงสร้าง TMatrixInitializer

-TCubeInitializer

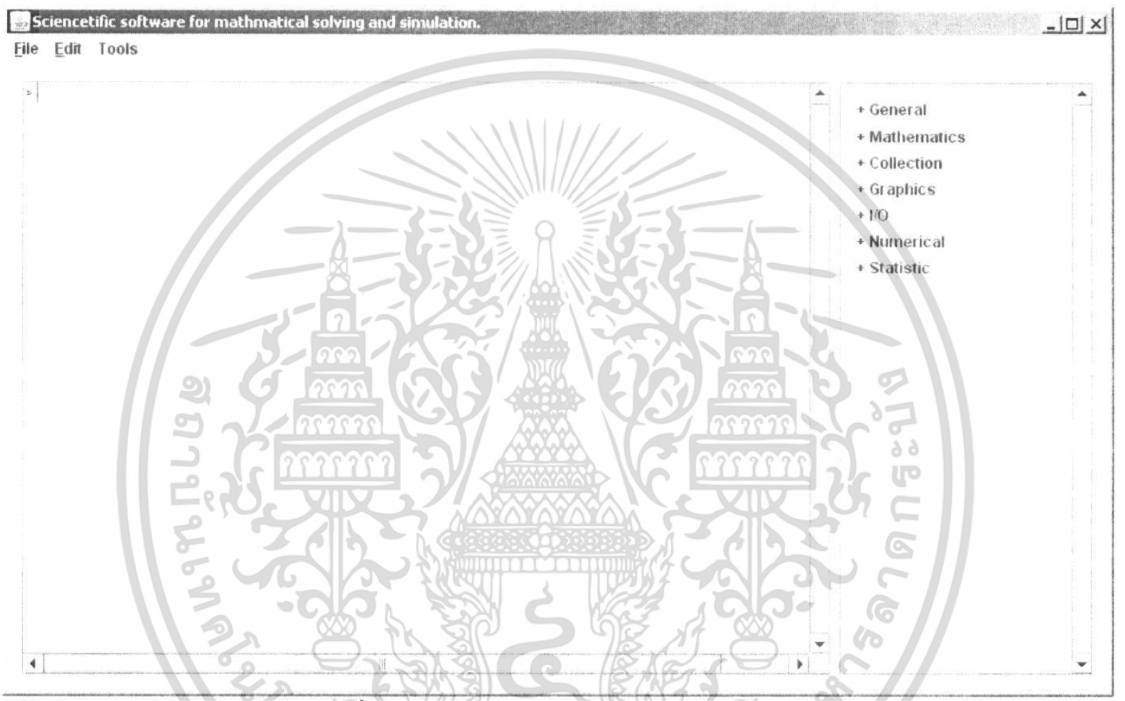
ทำหน้าที่เป็นตัวใส่ค่าให้แก่ข้อมูลคอลเลคชั่นแบบคิวบ์ ประกอบไปด้วยโครงสร้างดังนี้



ภาพที่ 3.23 แสดงโครงสร้าง TCubeInitializer

3.5 สร้างส่วน GUI เพื่อรับคำสั่งเข้ามาให้ตัวแปลภาษาทำงาน

ในการสร้าง GUI เพื่อรับคำสั่งโดยใช้โปรแกรม Netbeans ในการสร้างแล้วทำการ เชื่อมต่ออินพุตจากส่วน GUI และ ตัวแปลภาษาที่สร้างขึ้นมาแล้วก่อนหน้าซึ่งเมื่อเริ่มโปรแกรมขึ้นมานั้นจะพบอินเทอร์เฟส (Interface) ของโปรแกรมตามลักษณะดังภาพ ซึ่งแบ่งเป็นสวนของเมนูบาร์ (Menu bar) ซึ่งประกอบไปด้วยคำสั่ง File, Edit และ Tools ส่วนของเท็กซ์เอเรีย (Text area) ซึ่งเป็นส่วนสำหรับการป้อนคำสั่งของโปรแกรมและส่วนของเมนูลัดสำหรับฟังก์ชันทางด้านขวาของโปรแกรมเพื่อความสะดวกสำหรับผู้ใช้งานในการใช้คำสั่ง ดังแสดงในภาพ 3.24

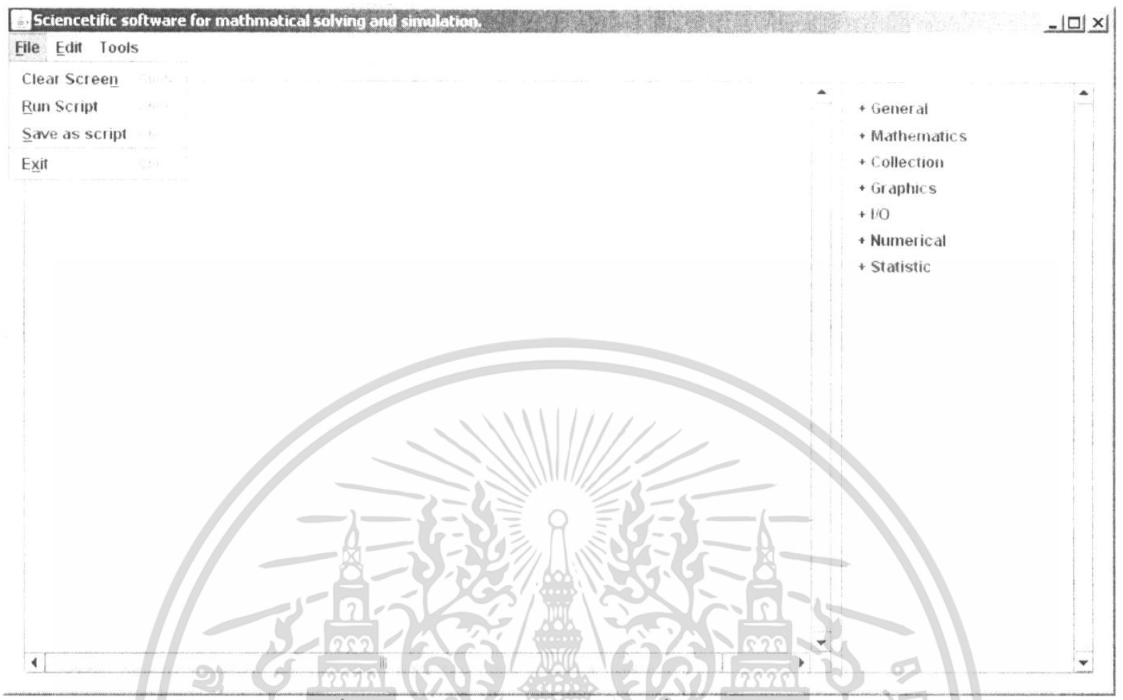


ภาพที่ 3.24 ภาพแสดงหน้าจอของโปรแกรม

ซึ่งในส่วนของเมนูนั้นจะประกอบไปด้วยเมนูหลักๆ 3 เมนูคือ File, Edit และ Tools ซึ่งรายละเอียดของแต่ละเมนูนั้นจะกล่าวถึงต่อไปดังต่อไปนี้

เมนูต่างๆในโปรแกรม

เมนู File

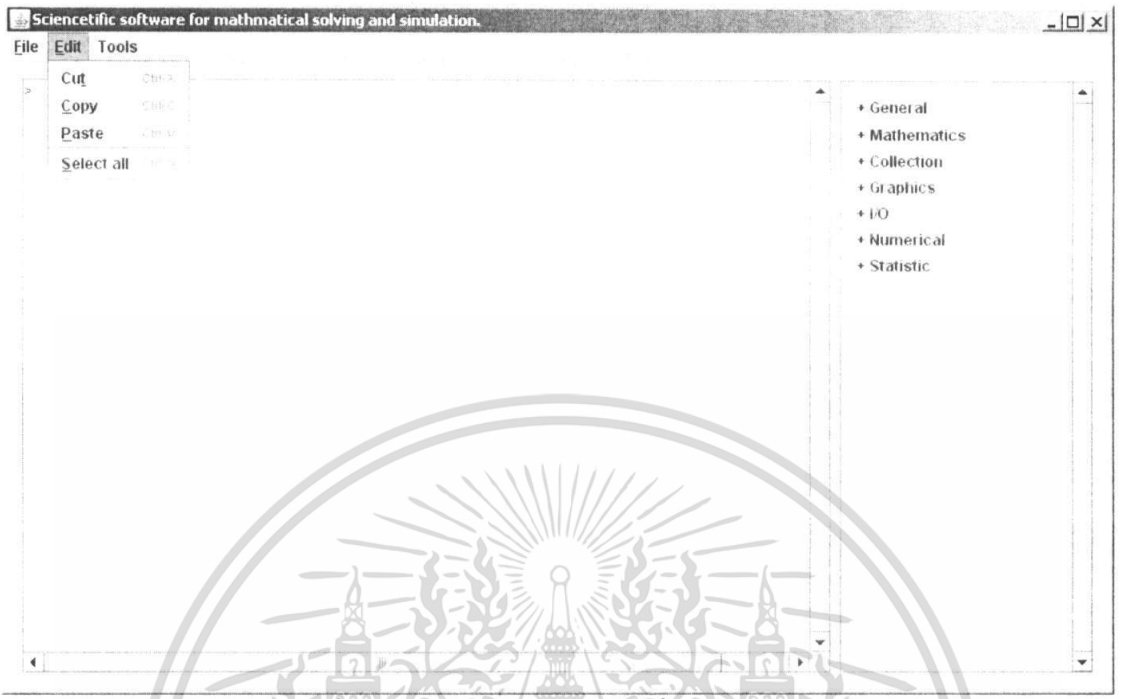


ภาพที่ 3.25 ภาพแสดงโปรแกรมเมื่อเลือกเมนู File

จากภาพเมื่อเราเลือกที่เมนู File นั้นเราจะพบฟังก์ชันต่างๆดังนี้

- Clear Screen เป็นฟังก์ชันสำหรับการล้างหน้าจอในส่วนของเท็กเอเรียทั้งหมด
- Run Script เป็นฟังก์ชันสำหรับการเปิดสคริปต์ไฟล์
- Save as script เป็นฟังก์ชันสำหรับการเซฟคำสั่งที่เราพิมพ์ไปทั้งหมดลงไปยังสคริปต์ไฟล์
- Exit ออกจากโปรแกรม

เมนู Edit

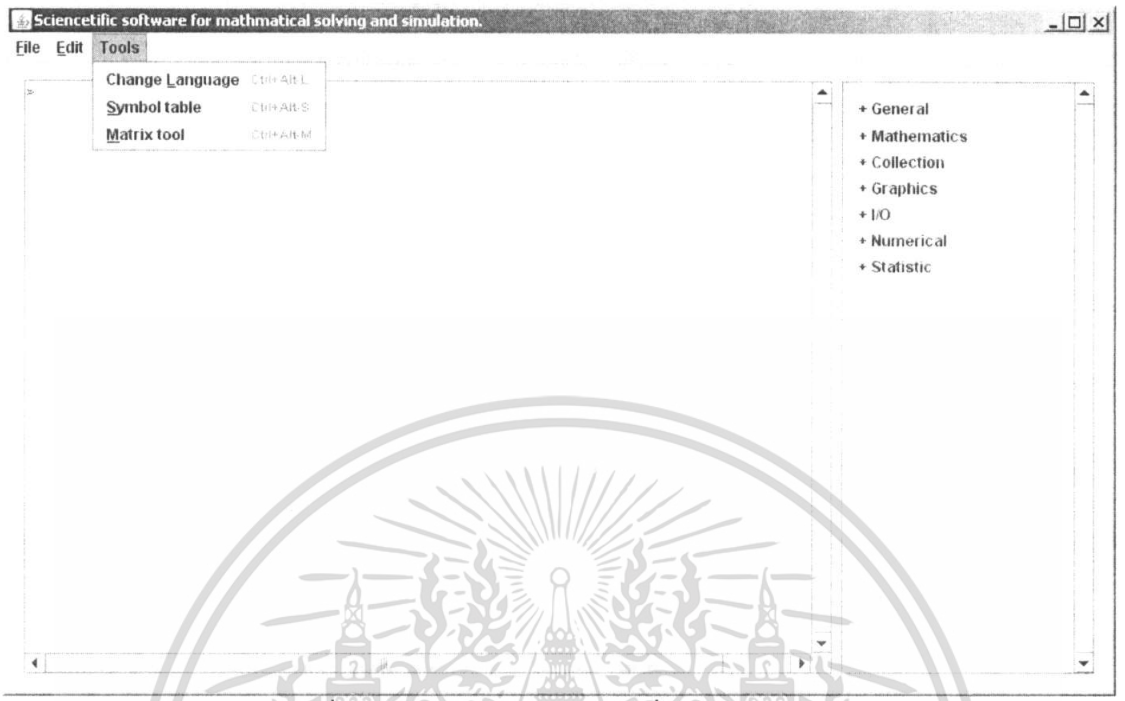


ภาพที่ 3.26 ภาพแสดงโปรแกรมเมื่อเลือกเมนู Edit

จากภาพเมื่อเราเลือกที่เมนู Edit นั้นเราจะพบฟังก์ชันต่างๆดังนี้

- Cut สำหรับคัดลอกข้อความแบบตัดออกไป
- Copy สำหรับคัดลอกข้อความ
- Paste สำหรับวางข้อความที่คัดลอกไว้
- Select ทำการเลือกข้อความทั้งหมด

เมนู Tools



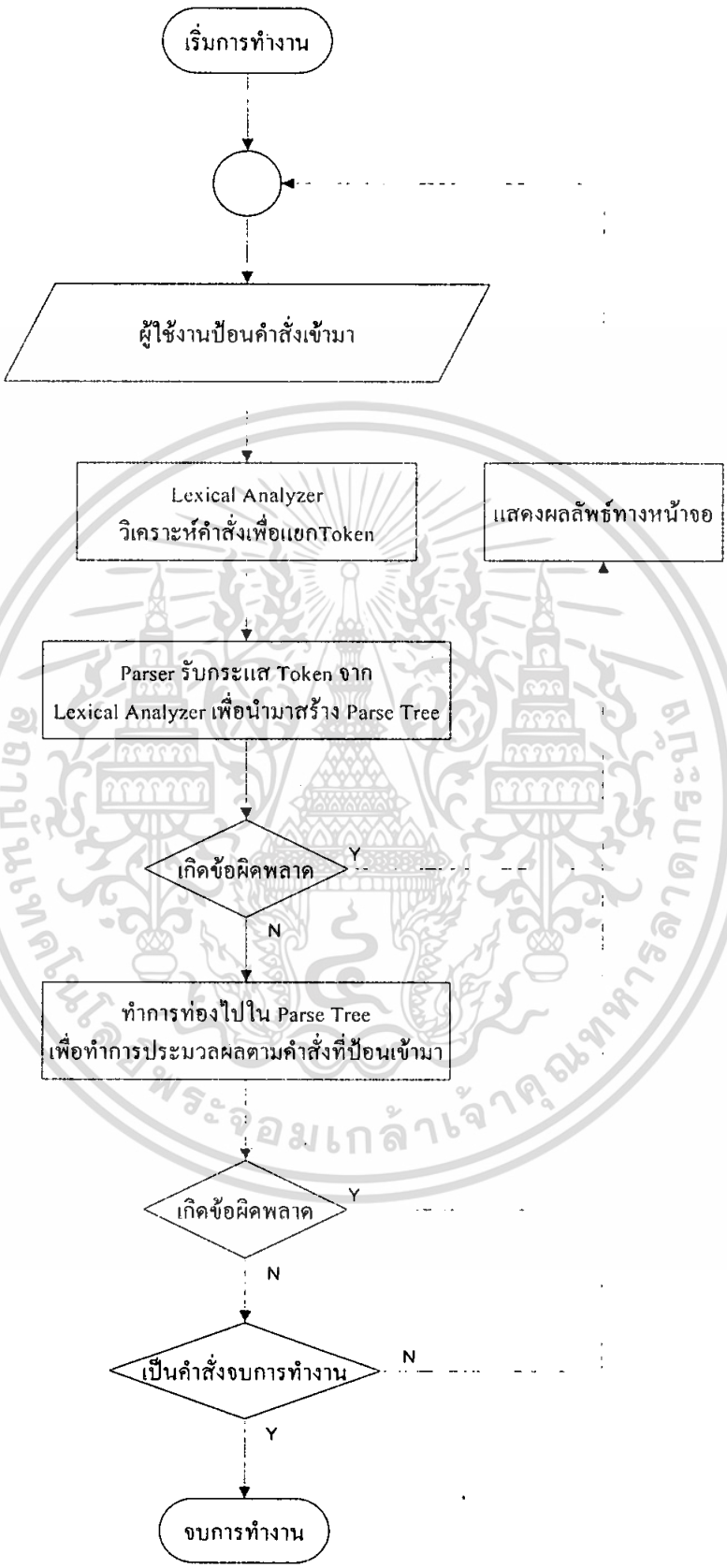
ภาพที่ 3.27 ภาพแสดงโปรแกรมเมื่อเลือกเมนู Tools

จากภาพเมื่อเราเลือกที่เมนู Edit นั้นเราจะพบฟังก์ชันต่างๆดังนี้

- Change Language สำหรับเปลี่ยนภาษา ไทย/อังกฤษ และการเปลี่ยนขนาดฟอนท์
- Symbol table ตารางแสดงรายชื่อ, ชนิดและค่าของตัวแปรหรือฟังก์ชันทั้งหมด
- Matrix tool เครื่องมือสำหรับการจัดการค่าในเมทริกซ์ที่ได้สร้างไว้แล้ว

และสำหรับในส่วนของเมนูลัดสำหรับฟังก์ชันนั้น จะเป็นการนำเอาฟังก์ชันที่ใช้ได้ทั้งหมดในโปรแกรมมาใส่ไว้ให้ในรูปแบบปุ่ม ซึ่งผู้ใช้งานสามารถกดใช้ได้ทันที ทั้งนี้เพื่อความสะดวกในกรณีที่ผู้ใช้งานจำคำสั่งไม่ได้ หรือเพื่อความสะดวกแก่ผู้ที่ไม่ถนัดที่จะใช้การพิมพ์มากๆ

ขั้นตอนการทำงานของโปรแกรม



ภาพที่ 3.28 แสดงลำดับขั้นตอนการทำงานของโปรแกรมในส่วนรับคำสั่ง

3.6 สร้างส่วน GUI ที่ทำการติดต่อกับไลบรารี่ของ JOGL เพื่อใช้ในการแสดงกราฟ

วิธีการติดตั้งพื้นฐาน

JOGL สามารถใช้งานร่วมกับ J2SE 1.4.2 ขึ้นไป ซึ่งในที่นี้จะใช้ Java 1.6.0 และ ดาวน์โหลด JSR-231 1.1.0 release candidate 2 ของ JOGL จาก https://jogl.dev.java.net ในคู่มือการใช้ของ JOGL แนะนำว่า JARs และ DLLs ควรจะถูกติดตั้งในไดเรกทอรี่ของตนเองแทนที่จะไปอยู่ใน JRE ไดเรกทอรี่ ซึ่งตัว JARs และ DLLs สามารถใช้โดยการกำหนด class-path และ ค่า java.library.path บน command line

The Callback Framework

JOGL มีคลาสของการสร้างส่วนแสดงผลกราฟิกหลักๆ 2 คลาสคือ GLCanvas และ GLJPanel ซึ่งนำมาใช้กับ GLAutoDrawable อินเตอร์เฟสซึ่งสามารถใช้งานได้เหมือนกับ drawing surfaces สำหรับคำสั่งใน OpenGL

GLCanvas นั้นมีไว้สำหรับนำไปใช้งานเหมือนกับ AWT's Cnavas คลาส และยังเป็นคอมโพเนนท์ขนาดใหญ่ตัวหนึ่งดังนั้นจึงต้องคำนึงเป็นอย่างมากหากนำไปใส่รวมกับ Swing อย่างไรก็ตาม GLCanvas นั้นสามารถทำงาน OpenGL ได้อย่างรวดเร็วมากเมื่อใช้คู่กับอุปกรณ์เร่งความเร็ว

GLJPanel นั้นเป็นเครื่องมือเล็กๆซึ่งใช้เพื่อแก้ปัญหาของ Swing ซึ่งในอดีตนั้นมีกิตติศัพท์ในเรื่องความช้าเนื่องจากมันจะทำการคัดลอก OpenGL frame buffer ลงไปยัง BufferedImage ก่อนที่จะทำการแสดงผล อย่างไรก็ตามใน Java SE 6 ก็ได้ถูกพัฒนาให้มีความเร็วขึ้นเป็นอย่างมาก สิ่งที่ทำให้ GLJPanel นั้นเหนือกว่า GLCanvas คือการยอมให้ ภาพ 3D ใน OpenGL กับ องค์ประกอบ 2D ใน Swing สามารถรวมกันได้ซึ่งเป็นหนทางใหม่ที่น่าตื่นตาตื่นใจ

การนำ GLCanvas ไปใช้

ออบเจ็กต์ของ GLCanvas นั้นจะคู่กับ GLEventListener ซึ่งจะทำการตอบสนองการเปลี่ยนแปลงใน GLCanvas และทำการวาดตามคำร้องขอ เมื่อ GLCanvas ถูกสร้างในครั้งแรก GLEventListener's เมธอด init() จะถูกเรียกขึ้นมาซึ่งเมธอดนี้ใช้ในการใส่ค่าเริ่มต้นให้แก่สถานะของ OpenGL ไม่ว่า canvas จะถูกเปลี่ยนขนาดรวมถึงตอนที่วาดครั้งแรก GLEventListener's reshape() จะทำงานขึ้นมาซึ่งมันสามารถที่จะทำการเขียนทับค่าเริ่มต้นของ viewport และ projection matrix ใน OpenGL ซึ่ง reshape() นั้นจะถูกเรียกใช้เมื่อ canvas นั้นมีการนำไปใช้กับ parent คอมโพเนนท์ด้วย ไม่ว่าเมื่อไหร่ก็ตามที่ display() เมธอด ถูกเรียกใช้ display() เมธอดใน GLEventListener จะทำงาน โค้ดในการวาดฉาก 3D นั้นควรจะถูกวางอยู่ในเมธอดนี้ รองไปจาก canvas และ listener เกมส่วนมากต้องการกรรมวิธีสำหรับการทริกเกอร์การอัพเดต canvas ซึ่งความสามารถนี้มีใน JOGL's FPSAnimator utility class ซึ่งสามารถกำหนดตารางเวลาไปยัง canvas สำหรับการเรียก display() เมธอดตามความถี่ที่ได้ตั้งเอาไว้

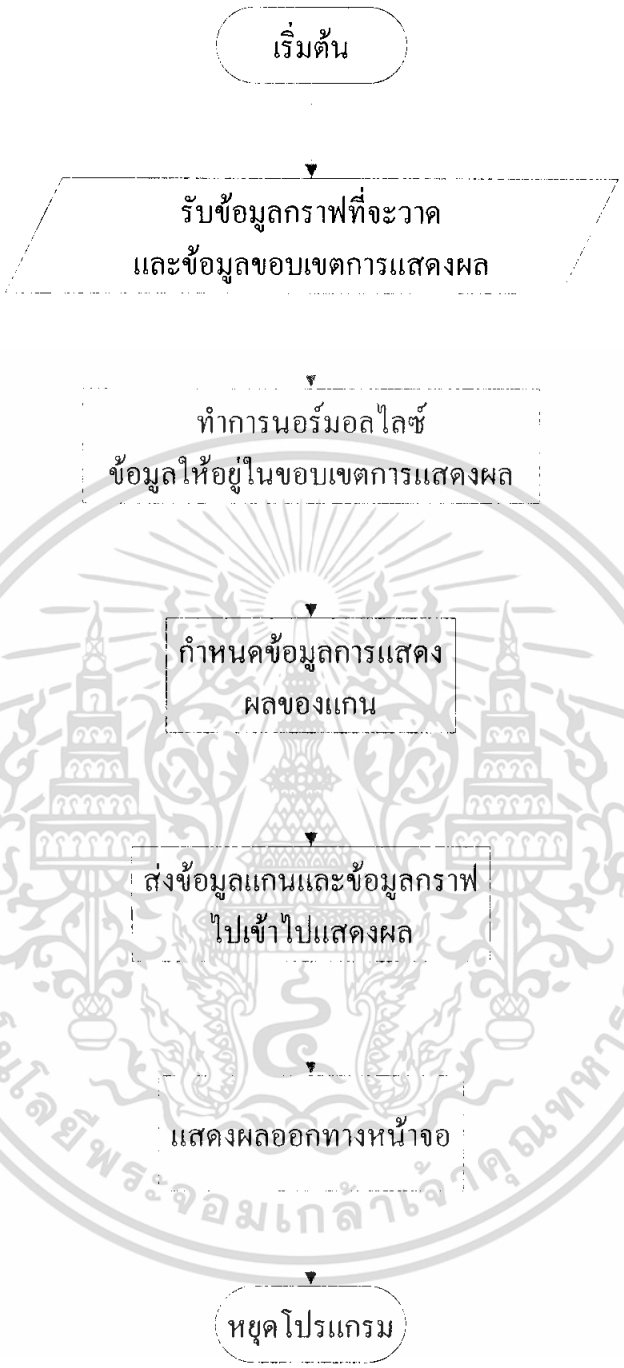
แสดงส่วนของโปรแกรมที่เขียนขึ้นเพื่อใช้งาน JOGL สำหรับการแสดงกราฟเป็นดังนี้

```
import com.sun.opengl.util.GLUT;
import javax.media.opengl.glu.GLU;
import javax.media.opengl.*;
import javax.swing.JFrame;

public class GLGraph extends JFrame implements GLEventListener {
   /* ประกาศ GLJPanel สำหรับเป็นพื้นที่แสดงผล */
   private GLJPanel glJPanel = null;
   /* ประกาศคลาส GLCapabilities สำหรับเก็บข้อมูลความสามารถในการแสดงผล */
   private GLCapabilities glCapabilities = null;
   /* ประกาศคลาส GLU รวมคำสั่งใน glu.h */
   private GLU glu = null;
   /* ประกาศคลาส GLUT รวมคำสั่งใน glut.h */
   private GLUT glut = null;

   public JoglExample() {
      setSize(400, 400);
      setLocationRelativeTo(null);
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
//สร้างตัวกำหนดคุณสมบัติ
glCapabilities = new GLCapabilities();
//กำหนดให้มีการใช้ฮาร์ดแวร์เร่งการประมวลผล
glCapabilities.setHardwareAccelerated(true);
//กำหนดให้มีการทำ double buffer
glCapabilities.setDoubleBuffered(true);
//สร้างพื้นที่แสดงผลให้มีคุณสมบัติตามที่กำหนด
glJPanel = new GLJPanel(glCapabilities);
//กำหนดตัวรองรับเหตการเป็นคลาสที่ทำงานปัจจุบัน
glJPanel.addGLEventListener(this);
```

```
//กำหนดพื้นที่แสดงผลให้กำหน้าต่างปัจจบัน
setContentPane(glJPanel);
glJPanel.setVisible(true);
//สร้าง GLU และ GLUT เพื่อใช้งาน
      glu = new GLU();
      glut = new GLUT();

      setVisible(true);
   }

   /* จะถูกเรียกให้ทำงานเมื่อเริ่มโปรแกรม สำหรับตั้งค่าพื้นฐานต่างๆ */
   public void init(GLAutoDrawable drawable) {
      GL gl = drawable.getGL();
      gl.glClearColor(1.0f, 1.0f, 1.0f, 0.0f);
      gl.glClearDepth(1.0f);
      gl.glEnable(GL.GL_CULL_FACE);
      gl.glFrontFace(GL.GL_CCW);
      gl.glShadeModel(GL.GL_SMOOTH);
      reshape(drawable, 0, 0, getWidth() , getHeight());
   }

/* จะถูกเรียกให้ทำงานสำหรับการแสดงผลออกทางหน้าจอ */
public void display(GLAutoDrawable drawable) {
      GL gl = drawable.getGL();
      gl.glClearColor(0.0f, 0.0f, 1.0f, 1.0f);
      gl.glClear( GL.GL_COLOR_BUFFER_BIT | GL.GL_DEPTH_BUFFER_BIT );
      gl.glLoadIdentity();

         //ทำการวาดกราฟที่นี่
   }
```

```
/* จะถูกเรียกเมื่อมีการเปลี่ยนขนาดของหน้าต่างโปรแกรม */
public void reshape(GLAutoDrawable drawable, int x, int y, int width, int height) {
    if (height == 0)
        height = 1;
    GL gl = drawable.getGL();
    // กำหนดพื้นที่ viewport
    gl.glViewport(0, 0, width, height);
    gl.glMatrixMode(GL.GL_PROJECTION);
    gl.glLoadIdentity();
    // กำหนดการโปรเจกชั่นเป็นแบบ perspective
    glu.gluPerspective(90.0f, (float)width/(float)height, 0.0001f, 50.0f);
    gl.glMatrixMode(GL.GL_MODELVIEW);
  }
/* จะถูกเรียกเมื่อมีการเปลี่ยนการตั้งค่าของหน้าจอ */
  public void displayChanged(GLAutoDrawable drawable,
      boolean modeChanged, boolean deviceChanged) {
  }
}
```

ลำดับขั้นตอนการแสดงผลเป็นกราฟ



ภาพที่ 3.29 แสดงลำดับขั้นตอนการทำงานของโปรแกรมในส่วนการแสดงกราฟ

หลักการที่ใช้ในการวาดกราฟ 3 มิติ

ภาพสามมิติแบบออบลิค (Oblique Drawing) ภาพออบลิคเป็นภาพเขียนแบบที่ด้านหน้ามีลักษณะตั้งตรง ส่วนภาพด้านข้างและด้านบนจะเอียงลึกลงไปเพียงด้านเดียว โดยมีขนาดที่ขนานเท่ากันตลอด มีมุมเอียง 0-90 องศา มีอยู่ 2 แบบ ดังนี้

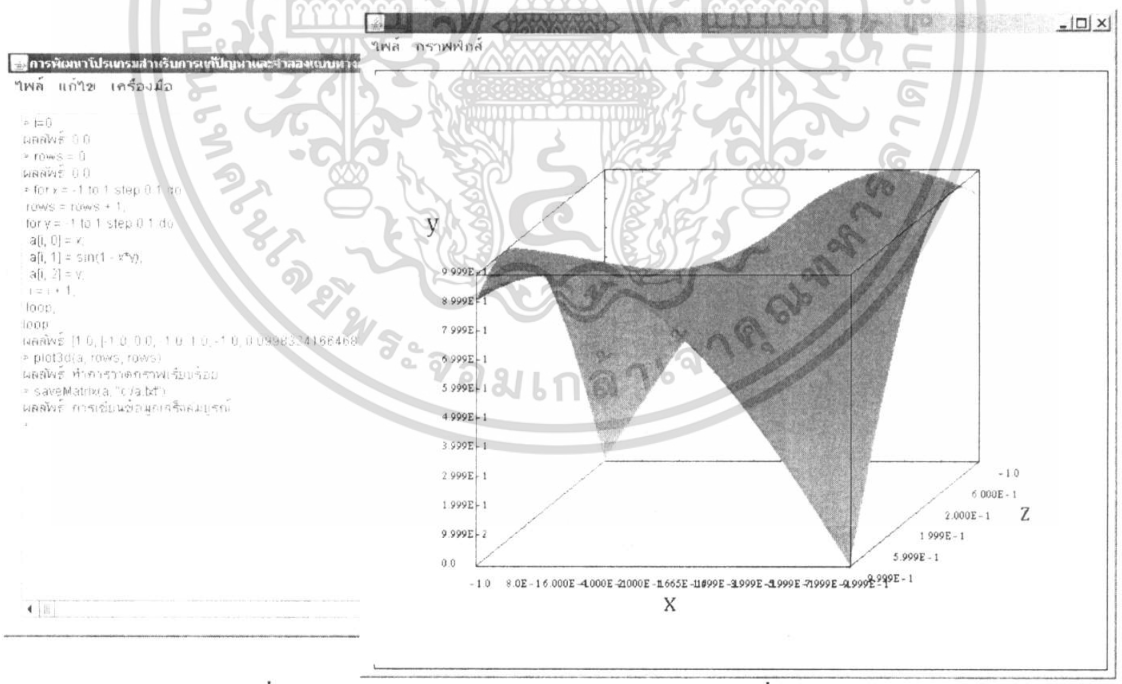
1. ภาพออบลิคแบบเต็มส่วน (Cavalier) เป็นแบบที่มีอัตราส่วนภาพระหว่าง ความกว้าง : ความสูง : ความลึก ของภาพเป็น 1 : 1 : 1 ดังแสดงในรูป

2. ภาพออบลิคแบบครึ่งส่วน (Cabinet) เป็นแบบที่มีอัตราส่วนภาพระหว่าง ความกว้าง : ความสูง : ความลึก ของภาพเป็น 1 : 1 : 0.5 ดังแสดงในรูป

โดยการแสดงผลของกราฟ 3 มิติจะแสดงผลเป็นแบบภาพออบลิคแบบครึ่งส่วนโดยมีรูปแบบของโปรเจคชันเมทริกดังนี้

$$\begin{bmatrix} 1 & 0 & -s\times\cos(\theta) & 0 \\ 0 & 1 & -s\times\sin(\theta) & 0 \\ 0 & 0 & \dfrac{-1}{z_{near}-z_{far}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

ตัวอย่างการวาดกราฟของฟังก์ชั่น



ภาพที่ 3.29 แสดงตัวอย่างการวาดกราฟของฟังก์ชั่น sin(1-x*z)

3.7 ทำให้ส่วนการแสดงผลกราฟสามารถแสดงผลออกทางเครื่องพริ้นเตอร์ได้

ใน JDK 1.0 และ 1.1 มีคลาสเกี่ยวกับการพิมพ์ออก Printers ที่ใช้งานค่อนข้างลำบาก แต่เริ่มจาก JDK 1.2 ใน package java.awt.print ก็มีคลาสชุดใหม่สำหรับพิมพ์ออก Printers ที่ใช้งานง่ายกว่าและยังยอมให้ใช้ Graphics2D ทำการ rendering ภาพที่จะพิมพ์ได้ด้วยวิธีเดียวกับภาพที่จะแสดงบนจอภาพ คลาสที่เกี่ยวกับการพิมพ์คือ PrintJob, PageFormat และ Printable

คลาส PrintJob ใช้สำหรับควบคุมนำ printing dialog หรือ page setup ขึ้นแสดง และสั่งให้เริ่มทำการพิมพ์ โปรแกรมที่ทำการพิมพ์ต้องสร้าง print job ขึ้นก่อน แต่คลาสนี้เป็น abstract class ดังนั้นจึงต้องมี factory method สำหรับสร้าง instance ขึ้นคือ

```
public static PrintJob getPrinterJob();
```

เมื่อได้ printer job แล้วเราต้องกำหนดสิ่งที่จะพิมพ์ ซึ่งเป็นคลาส implements Printable และนำไปกำหนดให้ print job โดย

```
public abstract void setPrintable(Printable);
public abstract void setPrintable(Printable, PageFormat);
```

หากต้องการแสดง printing dialog ให้เลือกกำหนด option ในการพิมพ์ ก็เรียก

```
public abstract boolean printDialog();
```

ถ้าค่าที่ส่งกลับมาเป็น true แสดงว่าผู้ใช้ไม่ได้ยกเลิกการพิมพ์ นอกนั้นเป็น false เมื่อต้องการใช้เริ่มการพิมพ์ก็เรียก

```
public abstract void print() throws PrinterException;
```

และหากจะยกเลิกก่อนการพิมพ์เสร็จก็เรียก

```
public abstract void cancle();
```

Printable เป็น interface สำหรับนำไปสร้างเป็นคลาสกำหนดสิ่งที่จะพิมพ์ใน interface นี้มีเพียง method เดียวคือ

public abstract int print(Graphic g, PageFormat p, int ind) throws PrinterException;

ซึ่งคลาสที่ implement Printable จะต้องสร้างขึ้นเพื่อกำหนดว่าจะวาดอะไรลงไปใน graphic ที่เป็นพารามิเตอร์นั้นบาง และจะใช้ page format เช่นใด ส่วนพารามิเตอร์ที่เป็น int กำหนดจุด index ของ page ที่จะพิมพ์ โดยเริ่มจาก page ที่ 0 เสมอ ใน interface นี้มีค่าคงที่อีกสองตัวคือ

public static final int PAGE_EXISTS;

public static final int NO_SUCH_PAGE;

print() จะส่ง PAGE_EXISTS ออกมาถ้ามี page สำหรับ index นั้น แต่จะส่ง NO_SUCH_PAGE ถ้าไม่มี page นั้น เมื่อเริ่มทำการพิมพ์ printer job จะเรียกไปที่ print() ของ Printable เพื่อนำ page ที่ถูก rendered แล้วนั้นไปพิมพ์ คลาส PageFormat ใช้กำหนดขนาดและ orientation ของ page ที่จะพิมพ์ มี constructor เป็น

public PageFormat();

ซึ่งจะให้ page format ที่เป็น default หรือกำหนดโดย printing dialog แต่หากเราต้องการกำหนด page format เองก็จะต้องใช้

public void setPaper(Paper p);

คลาส Paper อยู่ใน Package java.awt.print ใช้กำหนดขนาดกระดาษพิมพ์มี constructor เป็น

public Paper();

ซึ่งให้ paper ที่มีขนาด default เป็น 8.5" x 11" โดยปกติเราจะไม่พิมพ์จนถึงติดขอบกระดาษ แต่จะพิมพ์เฉพาะภายในพื้นที่ที่เรียกว่า imageable area ซึ่งมีขนาด default เป็น 6.5" x 9"

เราสามารถกำหนดขนาดของ paper และ imageable ได้โดยใช้

Public void setSize(double width, double height);

Public void setImageableArea(double x, double y, double width,double height);

ในคลาส PageFormat มีค่าคงที่สำหรับกำหนด orientation ของ page คือ

Public static final int LANDSCAPE;

Public static final int PORTRAIT;

Public static final int REVERSE_LANDSCAPE;

โดย PORTRAIT เป็นกระดาษยาวในแนวตั้ง ส่วน LANDSCAPE เป็นกระดาษยาวในแนวนอน

คลาส PageFormat มี method ให้เราตรวจสอบ orientation คือ

Public int getOrientation();

หากต้องการทราบขนาดของ paper ที่ใช้อยู่ก็เรียก

Public double getWidth();

Public double getHeight();

ค่าที่ให้ออกมามีหน่วยเป็น 1/72" ดังนั้น ขนาดมาตรฐานของ protrait page คือ 8.5" x 11" จะมีค่าเป็น 612 x 792 ถ้าต้องการราบ imageable area ก็ใช้

Public double getImageableX();

Public double getImageableY();

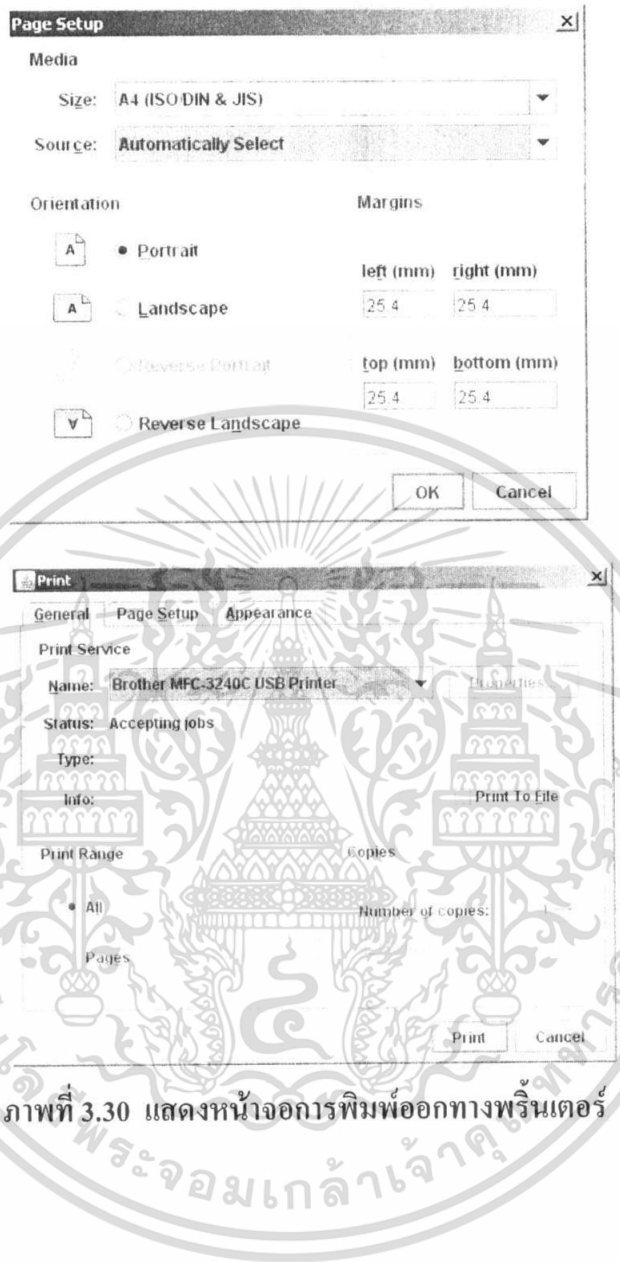
Public double getImageableWidth();

Public double getImageableHeight();

และถ้าต้องการทราบ paper ของ page format นี้ก็ใช้

Public Paper getPaper();

ตัวอย่างแสดงการพิมพ์ออก printer



ภาพที่ 3.30 แสดงหน้าจอการพิมพ์ออกทางพริ้นเตอร์

```
// PrintTest.java
import java.awt.*;
import java.awt.print.*;
class MyPrintable implements Printable {
        public int print(Graphics g, PageFormat pf, int i) {
                if (I < 0)
                        return NO_SUCH_PAGE;
                Graphics2D g2 = (Graphics2D) g;
                G2.setFont(new Font("Courier New"), Font.PLAIN, 30);
                G2.setPaint(Color.red);
                G2.drawString("HELLO!", 140, 140);
                Return PAGE_EXISTS;
                                }
}
class PrintTest {
        public satic void main(String args[]) {
                PrinterJob pj = PrinterJob.getPrinterJob();
                pj.setPrintable(new MyPrintable());
                if (pj.printDialog())
                        try {
                                pj.print();
                        } catch (PrinterException e) {
```

คลาส MyPrintable ทำการ implement Printable มี prin() สำหรับ rendering ภาพที่จะวาดลงใส่ Graphics g ในตัวอย่างนี้เพียงแต่วาด "Hello!" เท่านั้น

คลาส PrintTest สร้าง PrinterJob pj ขึ้นโดยใช้ getPrinterJob() ใช้ setPrintable() กำหนด my printable ที่จะพิมพ์ให้ จากนั้นเรียก pointDialog() เพื่อแสดง printing dialog หากผู้ใช้เลือก option และ ตอบตกลงจะได้ค่ากลับมาเป็น true ก็จะเริ่มการพิมพ์โดย print()

บทที่ 4
การประเมินผล

จากการศึกษาเนื้องานที่ผ่านมา ได้ทำการพัฒนาโปรแกรมการคำนวณทางคณิตศาสตร์ ซึ่งเมื่อได้นำมาเปรียบเทียบกับโปรแกรม MATLAB ซึ่งเป็นที่ยอมรับอย่างกว้างขวางนั้นปรากฏว่ามีค่าใกล้เคียงกันอย่างมาก จึงอาจกล่าวได้ว่าโปรแกรมที่พัฒนาขึ้นมานี้ให้ผลลัพธ์ที่สามารถเชื่อถือได้

4.1 ผลการทำงานของโปรแกรม

เพื่อพิสูจน์ให้เห็นว่าค่าผลลัพธ์ที่ได้จากการคำนวณโดยโปรแกรมที่ได้พัฒนาขึ้นมานี้มาความน่าเชื่อถือ จึงได้ทำการเปรียบเทียบผลลัพธ์กับโปรแกรมที่เป็นที่ยอมรับกันอย่างกว้างขวาง นั่นคือ โปรแกรม MATLAB ซึ่งเราจะแสดงให้เห็นโดยใช้กรณีศึกษาซึ่งดังต่อไปนี้

4.1.1 กรณีศึกษา : การใช้โปรแกรมสำหรับการหาค่า ห.ร.ม และ ค.ร.น.

เราสามารถใช้โปรแกรมนี้ทำงานได้เหมือนเครื่องคิดเลขทั่วๆไปได้ซึ่งสามารถทำงานพื้นฐานได้อย่างครบถ้วน และนอกจากนั้นยังสามารถใช้ฟังก์ชันทางคณิตศาสตร์ทั่วไปต่างๆได้ด้วย โดยการทำงานคำสั่งหนึ่งๆนั้นเราสามารถป้อนฟังก์ชันทีละฟังก์ชันหรือผสมหลายๆฟังก์ชันด้วยกันในแต่ละชุดคำสั่งก็ได้ โดยเราจะยกตัวอย่างโดยการเขียนฟังก์ชันในรูปของสคริปท์ไฟล์สำหรับการหาค่าหารร่วมมาก (ห.ร.ม.) และคูณร่วมน้อย (ค.ร.น.)

- หารร่วมมาก (Great common divide)

```
function v = gcd(a, b)
c= -1;
if a < b then
  c = a;
  a = b;
  b = c;
end if;

while c != 0 do
   c = a % b;
   a = b;
   b = c;
loop;
```

```
 v = a;
end function$
```

ซึ่งเราสามารถนำไปใช้โดยการเรียกฟังก์ชัน gcd ได้ดังนี้

```
> gcd(25,250)
Output.: 25.0
```

คุณร่วมน้อย (Least common multiple)

```
function v = lcm(a, b)
 c= -1;
 x = a;
 y = b;
 if a < b then
  c = a;
  a = b;
  b = c;
 end if;

 while c != 0 do
  c = a % b;
  a = b;
  b = c;
 loop;
 z = a;
 v = (x*y)/z;
end function$
```

ซึ่งเราสามารถนำไปใช้โดยการเรียกฟังก์ชัน lcmได้ดังนี้

```
> lcm(25,250)
Output.: 250.0
```

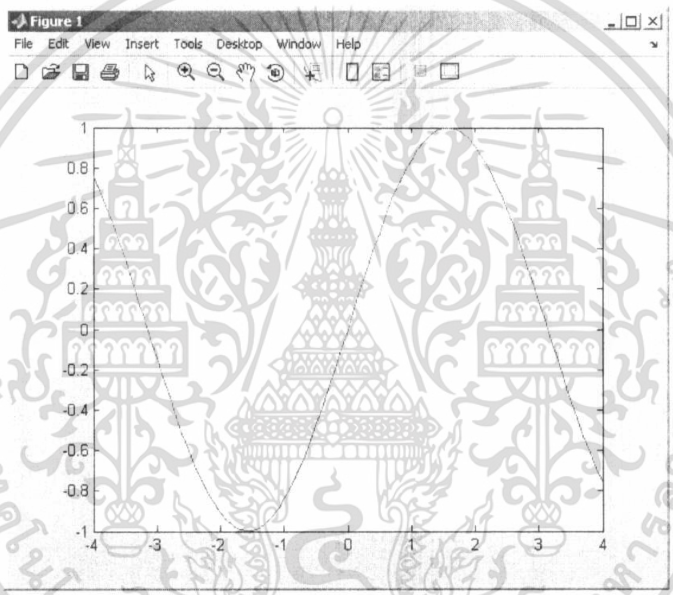
4.1.2 กรณีศึกษา : การประมวลผลและวาดกราฟ 2 มิติ

- ตัวอย่างการวาดภาพใน MATLAB

โดยเราจะวาดกราฟ y = sin(x) โดยการใช้ฟังก์ชันดังนี้

```
>> x = -4:0.1:4
y = sin(x)
plot(x,y)
```

จะได้กราฟดังรูป



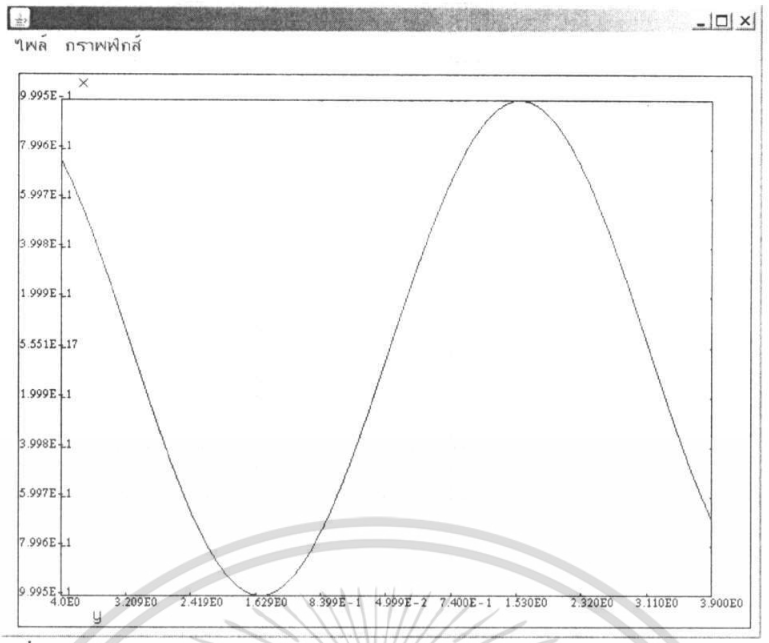
ภาพที่ 4.1 แสดงผลลัพธ์ของกราฟฟังก์ชัน f = sin(x) ในโปรแกรม MATLAB

- ตัวอย่างการวาดภาพใน MatVis

โดยเราจะวาดกราฟ y = sin(x) โดยการใช้ฟังก์ชันดังนี้

```
> function f(x) = sin(x) end function
ผลลัพธ์: ชื่อของฟังก์ชั่น: "f", ประเภทของฟังก์ชั่น: ฟังก์ชั่นของผู้ใช้งาน.
> fplot2d("f",-4,4,0.1)
ผลลัพธ์: ทำการวาดกราฟเรียบร้อย
```

จะได้กราฟดังรูป

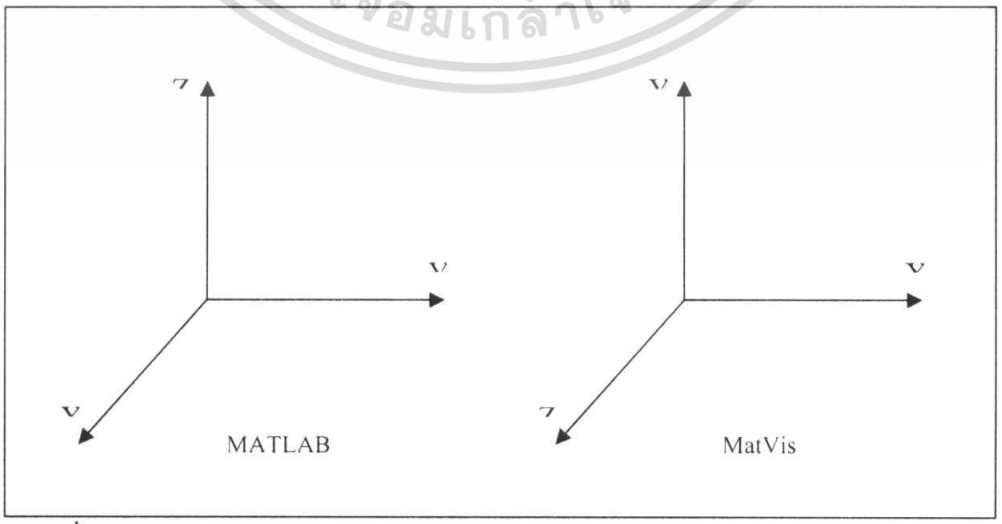


ภาพที่ 4.2 แสดงผลลัพธ์ของกราฟฟังก์ชัน f = sin(x) ในโปรแกรม Matvis

จากรูปข้างบนเป็นการวาดกราฟของฟังก์ชัน f = sin(x) โดยค่า x อยู่ในช่วง -4 ถึง 4 และเพิ่มค่าขึ้นทีละ 0.1 ซึ่งเมื่อเปรียบเทียบภาพกราฟกันระหว่างสองโปรแกรมนั้น พบว่ามีความใกล้เคียงกันมาก

4.1.3 กรณีศึกษา : การประมวลผลและวาดกราฟ 3 มิติแบบเส้น

ในการแสดงผลในรูปแบบของกราฟ 3 มิติระหว่างโปรแกรม MatVis และ MATLAB นั้น จะมีข้อแตกต่างกันตรงที่ตำแหน่งของแกน ซึ่งระบบแกนของ MatVis นั้นจะใช้ระบบแกนตามแบบของ OpenGL โดยรูปแบบของแกนจะเป็นไปตามที่แสดงดังภาพ



ภาพที่ 4.3 แสดงความแตกต่างของระบบแกนของโปรแกรม MATLAB และ MatVis

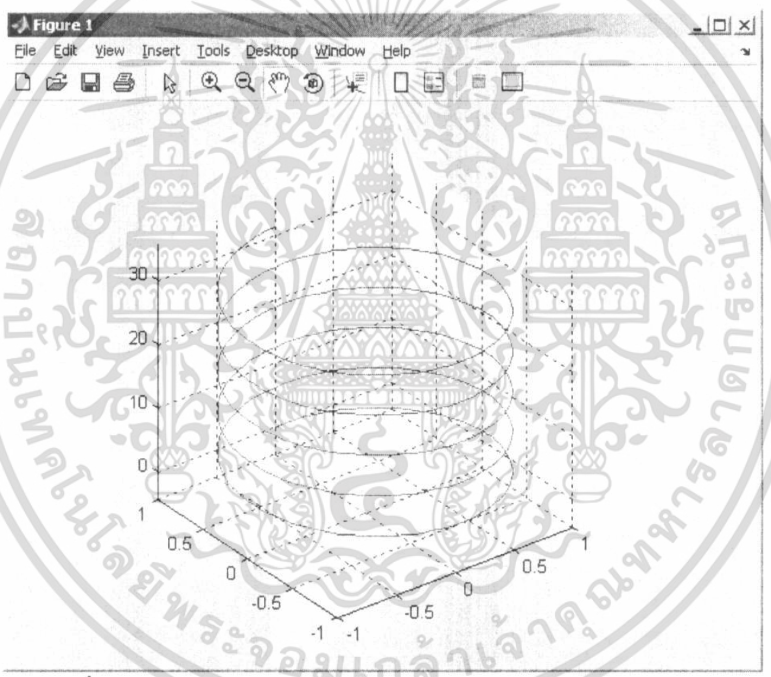
โดยเราจะแสดงตัวอย่างการวาดกราฟจากฟังก์ชั่นดังต่อไปนี้

- ตัวอย่างการวาดกราฟใน MATLAB

โดยเราจะวาดกราฟของค่า sin(t),cos(t) และ t โดยเรียงตามแกน x,y,z ตามลำดับ โดยการใช้ฟังก์ชันดังนี้

```
>>t = 0:pi/50:10*pi;
>>plot3(sin(t),cos(t),t)
>>axis square; grid on
```

จะได้กราฟดังรูป



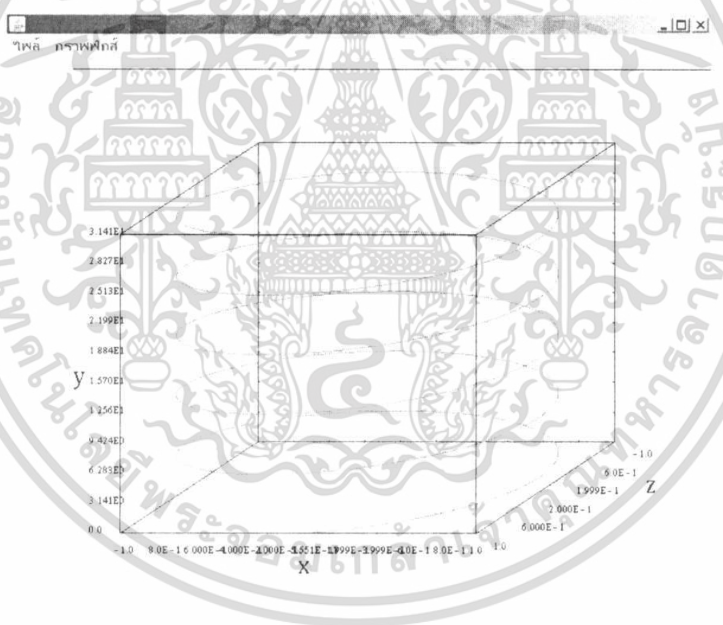
ภาพที่ 4.4 แสดงผลลัพธ์ของกราฟ 3 มิติแบบเส้นใน MATLAB

- ตัวอย่างการวาดกราฟใน MatVis

โดยเราจะวาดกราฟของค่า sin(t), cos(t) และ t โดยเรียงตามแกน x,y,z ตามลำดับ โดยการใช้ฟังก์ชันดังนี้

```
> i=0
ผลลัพธ์: 0.0
> for t = 0 to 10 * _PI step _PI / 50 do
 m[i, 0] = sin(t);
 m[i, 1] = t;
 m[i, 2] = cos(t);
 i = i + 1;
loop
>lplot3d(m)
```

จะได้กราฟดังรูป



ภาพที่ 4.5 แสดงผลลัพธ์ของกราฟ 3 มิติแบบเส้นใน MatVis

จะเห็นว่ารูปกราฟนั้นมีจะมีความใกล้เคียงกันมาก ซึ่งจะแตกต่างกันเพียงรูปแบบการแสดงผลของแกนเท่านั้น

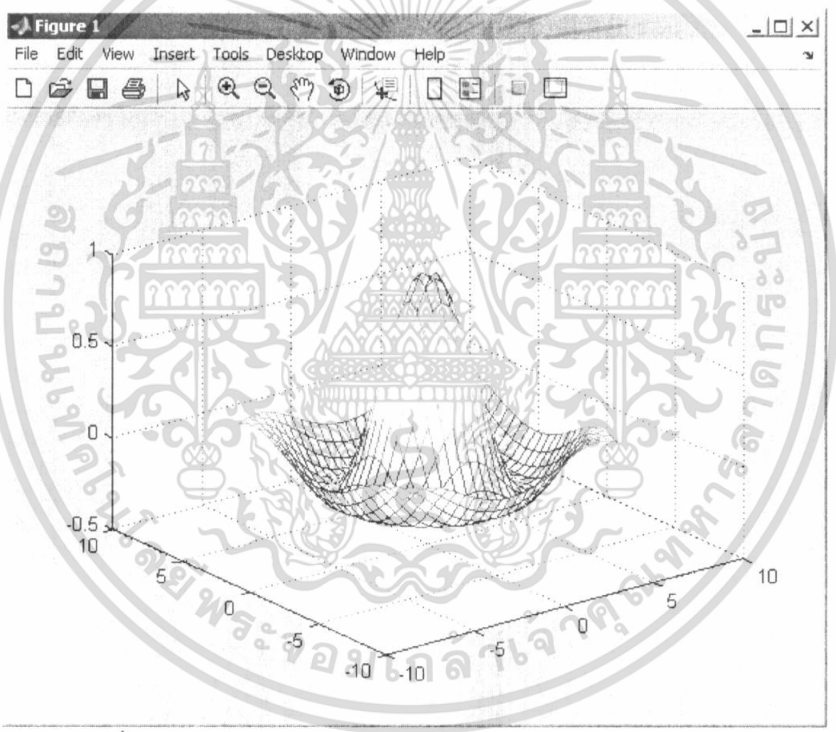
4.1.4 กรณีศึกษา : การประมวลผลและวาดกราฟ 3 มิติแบบแสดงพื้นผิวและแบบตาข่าย

- ตัวอย่างการวาดกราฟใน MATLAB

โดยเราจะวาดกราฟโดยการใช้ฟังก์ชันดังนี้

```
[X,Y] = meshgrid(-6:.5:6);
R = sqrt(X.^2 + Y.^2)
Z = sin(R)./R;
mesh(X,Y,Z)
```

จะได้กราฟดังรูป



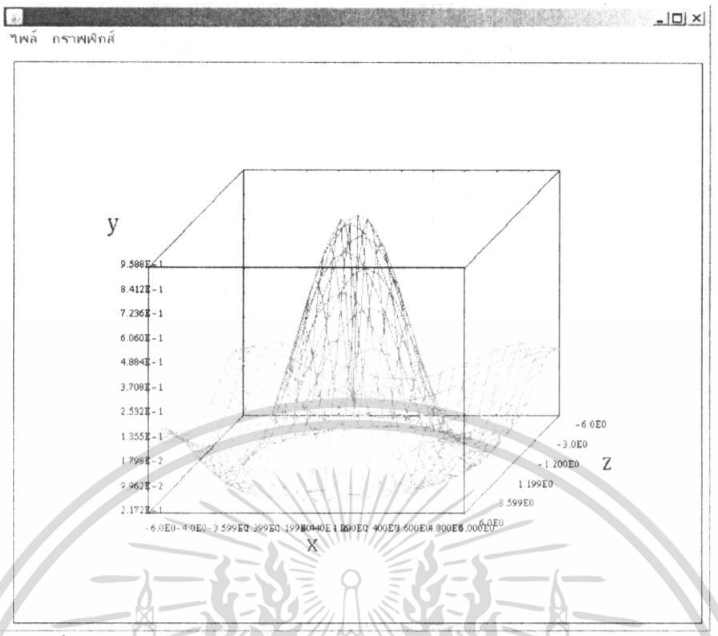
ภาพที่ 4.6 แสดงผลลัพธ์ของกราฟ 3 มิติแบบตาข่ายใน MatVis

- ตัวอย่างการวาดกราฟใน MatVis

โดยเราจะวาดกราฟโดยการใช้ฟังก์ชันดังนี้

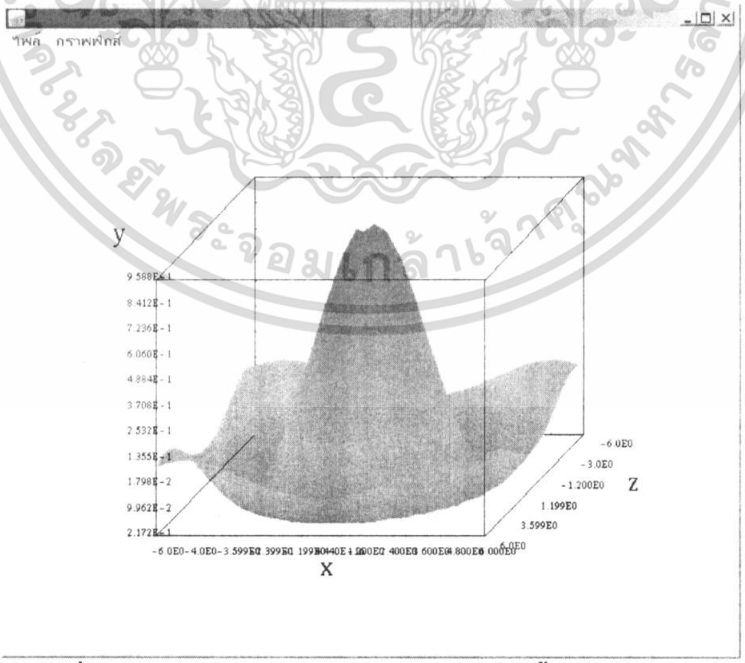
```
function f(x,y) = sin(sqrt(x^2+y^2)) / sqrt(x^2+y^2) end function
m = meshgrid("f",-6,6,0.5)
plot3d(m,25,25)
```

จะได้กราฟดังรูป



ภาพที่ 4.7 แสดงผลลัพธ์ของกราฟ 3 มิติแบบตาข่ายใน MatVis

ซึ่งเราสามารถเปลี่ยนโหมดการแสดงผลระหว่างแบบพื้นผิวและตาข่ายได้โดยการเลือกที่เมนูกราฟฟิกส์ > แสดงผลแบบเส้น นอกจากนั้นยังสามารถปรับค่าอื่นๆได้อีกด้วย



ภาพที่ 4.8 แสดงผลลัพธ์ของกราฟ 3 มิติแบบพื้นผิวใน MatVis

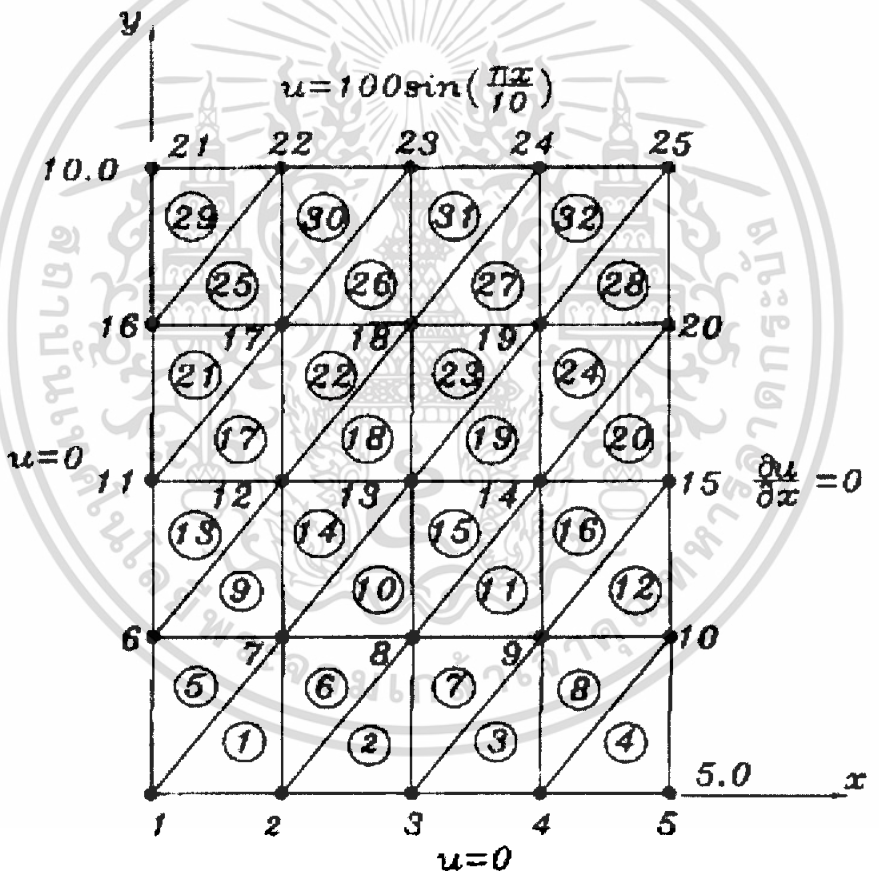
4.1.5 กรณีศึกษา : การแก้ปัญหาแบบไฟไนต์อิลิเมนท์

การใช้โปรแกรม MATLAB และ MatVis แก้ปัญหาการวิเคราะห์การถ่ายเทความร้อนแบบอุณหภูมิคงที่ โดยในส่วนนี้จะเป็นการแสดงผลการแก้ปัญหา 2 มิติ โดยใช้โปรแกรม MATLAB และ MatVis และนำผลมาเปรียบเทียบกัน

โดยเราจะแสดงตัวอย่างการแก้ปัญหาสมการลาปลาส 2 มิติด้วย ซึ่งมีเงื่อนไขดังนี้

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$$

โดยที่โดเมนและขอบเขตจะแสดงดังรูป

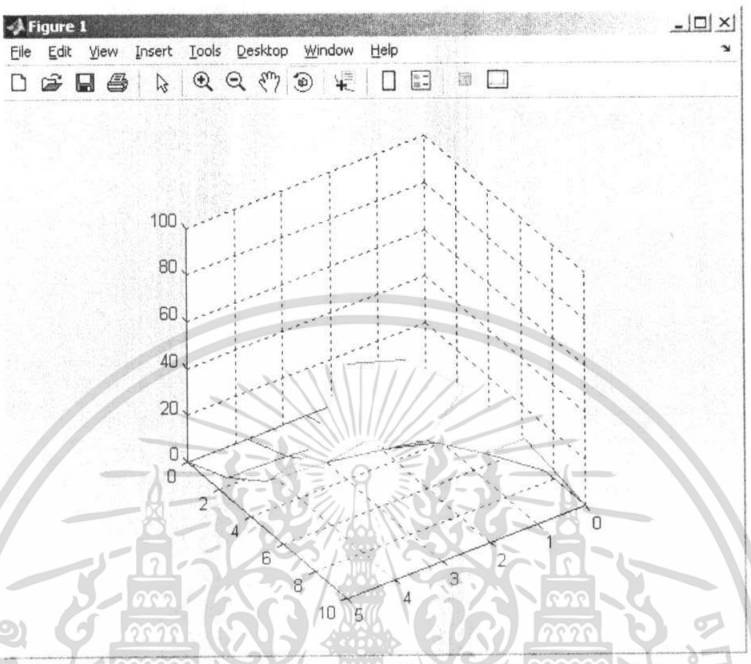


ภาพที่ 4.9 ตารางของ Triangular elements

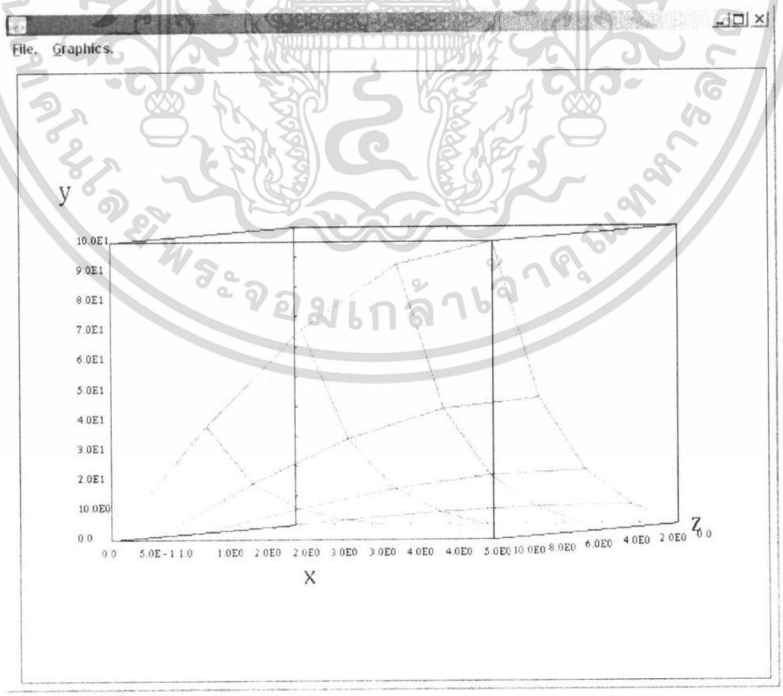
สำหรับที่ 0 < x < 5 และ 0 <y < 10 ขอบเขตเงื่อนไขคือ u(x,0) = 0 สำหรับ 0 < x <5 0, u(0,y) = 0 สำหรับ 0 < y < 10, u(x,10) = $100\sin((\frac{\pi x}{10}))$ $\frac{\partial u(5,y)}{\partial x}$ = 0 สำหรับ 0 < y < 10 โดย

โปรแกรมหลักที่มีฟังก์ชั่นที่ใช้ใน MATHLAB โดยในส่วนของโปรแกรมการแก้ปัญหาของทั้ง MATLAB และ MatVis นั้นสามารถดูได้ที่ภาคผนวก ข

ซึ่งแสดงผลลัพธ์ออกมาในรูปของกราฟได้ดังนี้



ภาพที่ 4.10 แสดงกราฟของการแก้ปัญหาแบบไฟไนต์อิลิเมนท์ของโปรแกรม MATLAB



ภาพที่ 4.11 แสดงกราฟของการแก้ปัญหาแบบไฟไนต์อิลิเมนท์ของโปรแกรม MatVis

ซึ่งเราได้นำค่าผลลัพธ์มาเปรียบเทียบได้ผลดังตารางดังนี้

โหนด	ค่า Exact	ผลลัพธ์ใน MATLAB	ผลลัพธ์ใน MatVis	ผลต่างใน MATLAB	ผลต่างใน MatVis
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0
6	0	0	0	0	0
7	2.878461189	3.051584651	3.051584651	6.013548723	6.013015505
8	5.318702742	5.638593307	5.638593307	6.014627635	6.014501795
9	6.949219993	7.367177268	7.367177268	6.015080873	6.01475376
10	7.521781465	7.974175226	7.974175226	6.014517802	6.014188445
11	0	0	0	0	
12	7.625671765	7.961473285	7.961473285	4.403530168	4.403179842
13	14.0904041	14.71088539	14.71088539	4.406671398	4.406567717
14	18.41000007	19.22069855	19.22069855	4.403585008	4.403577128
15	19.92684034	20.80433412	20.80433412	4.402569378	4.402740594
16	0	0	0	0	
17	17.32360723	17.71960664	17.71960664	2.283537289	2.283575593
18	32.00985222	32.74157327	32.74157327	2.28553577	2.285452259
19	41.82288724	42.77893019	42.77893019	2.285584487	2.285656664
20	45.26876661	46.30357756	46.30357756	2.285449204	2.285399636
21	0	0	0	0	0
22	38.26834377	38.2683	38.2683	0.000783945	0.000783945
23	70.71067894	70.7107	70.7107	0.000424262	0.000424262
24	92.38795392	92.388	92.388	0	0
25	100	100	100	0	0

ตารางที่ 4.1 แสดงการเปรียบเทียบผลลัพธ์ของการแก้ปัญหาแบบไฟไนต์อิลิเมนต์ระหว่าง MATLAB และ MatVis

ซึ่งจากตารางจะเห็นว่าความแตกต่างระหว่างค่าจริงกับค่าที่คำนวณได้ของทั้งโปรแกรม MATLAB และ MatVis นั้นมีความใกล้เคียงกันมาก ซึ่งแสดงให้เห็นว่าโปรแกรม MatVis นั้นมีประสิทธิภาพที่น่าเชื่อถือได้เมื่อนำมาเทียบกับโปรแกรมที่มีผู้ใช้ที่มีความนิยมมากทั่วโลกอย่าง MATLAB

4.2 ข้อจำกัดของโปรแกรม

- ในการรับชื่อฟังก์ชันนั้นจำเป็นต้องใส่เครื่องหมาย " " ปิดหัวปิดท้ายที่ชื่อด้วยทุกครั้ง ตัวอย่างเช่น fplot2d("f",4,-4,0.1) ซึ่งหมายความว่าเราจะพล็อตกราฟจากฟังก์ชันชื่อ f ในช่วงค่าต่ำสุดที่ -4 ถึงค่าสูงสุดคือ 4 โดยมีการเพิ่มค่าทีละ 0.1 ซึ่งถ้าเราไม่ใส่เครื่องหมาย " " นั้นจะทำให้โปรแกรมไม่สามารถประมวลผลได้และตีข้อความผิดพลาด
- ยังมีปัญหาเรื่องการจัดการหน่วยความจำตอนคำนวณข้อมูลปริมาณมากๆ ซึ่งยังได้ความเร็วที่ไม่น่าพอใจนัก
- มุมมองของกราฟนั้นสามารถหมุนดูได้ในมุม 90 องศาเท่านั้น ซึ่งทำให้อาจเป็นอุปสรรคในการดูเรื่องมุมมองของกราฟ
- ในการขึ้นบรรทัดใหม่แต่ละครั้งนั้นถือว่าเป็นการรันคำสั่ง 1 คำสั่ง เพราะฉะนั้นเวลาที่เราต้องการจะพิมพ์คำสั่งที่มีมากกว่า 1 บรรทัดนั้น เราจำเป็นที่ต้องกดชิป (Shift) ค้างไว้ทุกครั้งก่อนทำการขึ้นบรรทัดเพื่อไม่ให้โปรแกรมทำการรันคำสั่งที่เรายังพิมพ์ไม่ครบนั้นๆ
- เมื่อทำการเปลี่ยนภาษาของโปรแกรมนั้นจำเป็นต้องเริ่มการทำงานของโปรแกรมใหม่ทุกครั้งเพื่อให้แสดงผลตามที่เปลี่ยนไป
- เสปคของเครื่องที่ใช้โปรแกรมนี้ควรมีแรมไม่ต่ำกว่า 512 MB เพราะถ้ามีหน่วยความจำน้อยกว่าที่กำหนดไว้อาจทำให้ประสิทธิภาพเรื่องความเร็วของโปรแกรมนั้นลดลงไป ในกรณีที่ใช้งานฟังก์ชันที่มีการคำนวณมากๆ

บทที่ 5
สรุปและข้อเสนอแนะ

5.1 สรุปผล

การศึกษาทางด้านวิทยาศาสตร์และวิศวกรรมศาสตร์ในปัจจุบันนั้นเราไม่สามารถปฏิเสธได้เลยว่าคณิตศาสตร์เป็นสิ่งที่จำเป็นสำหรับการศึกษาหรือแก้ปัญหาต่างๆ ซึ่งในการแก้ปัญหาทางคณิตศาสตร์นั้นจำเป็นต้องใช้เครื่องมือเข้ามาช่วยในการคำนวณ ดังนั้นคอมพิวเตอร์และซอฟต์แวร์ทางด้านคณิตศาสตร์จึงมีบทบาทช่วยทุ่นแรงและลดเวลาในการคำนวณได้เป็นอย่างมาก ในประเทศไทยนั้นแทบจะไม่มีหรือค่อนข้างหายากสำหรับโปรแกรมทางด้านคณิตศาสตร์และวิทยาศาสตร์ที่พัฒนาขึ้นมาใช้งานเอง ซึ่งในปัจจุบันเรามักใช้ซอฟต์แวร์ที่เป็นของต่างประเทศเป็นเครื่องมือ ซึ่งในประเทศไทยยังมีการพัฒนาโปรแกรมคำนวณทางด้านคณิตศาสตร์น้อยมาก ซึ่งส่วนใหญ่เน้นด้านการคำณวนเฉพาะปัญหา และยังต้องใช้โปรแกรมอื่นในการวาดกราฟ ดังนั้นจึงน่าจะมีการพัฒนาโปรแกรมทางด้านนี้ให้ใช้งานง่ายไม่ซับซ้อน และเป็นพื้นฐานในการผลักดันและพัฒนาซอฟต์แวร์ทางวิทยาศาสตร์และกราฟิกต่อไปต่อไป

โครงงานนี้ได้ทำการศึกษาสมการหาข้อมูลเพื่อทำการสร้างโปรแกรมที่สามารถเป็นเครื่องมือในการแก้ปัญหาทางคณิตศาสตร์ขึ้น โดยได้ทำการศึกษาวิธีพัฒนาโปรแกรมเพื่อช่วยในการคำนวณและแก้ปัญหาพื้นฐานทางคณิตศาสตร์ โดยแกนหลักของโปรแกรมต้องอาศัยพื้นฐานทางด้านทฤษฎีคอมไพเลอร์ และสามารถแสดงกราฟในรูปแบบ 2 มิติ และ 3 มิติซึ่ง

ในการทดลองได้ใช้ปัญหาของการหาผลเฉลยของและใช้ระเบียบวิธีไฟไนต์เอลิเมนท์ในการหาผลเฉลย แล้วพัฒนาโปรแกรมขึ้นมาเพื่อหาผลเฉลยดังกล่าวโดยใช้โปรแกรมที่พัฒนาขึ้นแล้วนำไปเปรียบเทียบผลที่ได้กับโปรแกรม MATLAB ซึ่งโปรแกรมที่พัฒนาขึ้นมาเป็นการแก้ปัญหาที่เป็นสี่เหลี่ยมมุมฉาก และมีการแสดงผลลัพธ์ด้วยกราฟฟิคให้เข้าใจง่าย

ผลลัพธ์ที่ได้จากโปรแกรมเมื่อนำไปเปรียบเทียบกับโปรแกรม MATLAB ซึ่งเป็นซอฟต์แวร์ที่เป็นที่ยอมรับ จะมีความแตกต่างกันเล็กน้อยซึ่งตัวเลขที่แสดงผลออกมาจากโปรแกรมที่พัฒนาขึ้นมาดีกว่าเล็กน้อย และแทบจะไม่พบปัญหาใดๆในการทำงานของโปรแกรม ส่วนของการแสดงผลกราฟนั้นก็แทบไม่มีความผิดเพี้ยน แตกต่างกันแค่เพียงเฉดสีของกราฟ

5.2 ข้อเสนอแนะ

1. ในเรื่องมุมมองของกราฟเวลาปรับเปลี่ยนมุมมองนั้น ทำได้ค่อนข้างยากเนื่องจากการบังคับมุมกล้องในมุมมอง 3 มิตินั้น ค่อนข้างจะอิสระมาก ซึ่งบางทีอาจทำให้ผู้ใช้งานสับสน ดังนั้นจึงใช้วิธีการจำกัดมุมมองไว้ที่ 90 องศาเพื่อไม่ให้การปรับมุมมองนั้นอิสระมากจนเกินไป เพื่อนำมาแก้ปัญหาในเรื่องนี้
2. ในการใช้งานฟังก์ชันต่างๆในโปรแกรมนั้นบางฟังก์ชันก็ใช้งานได้ยากถ้าเราไม่รู้โครงสร้างของฟังก์ชันนั้นๆ หรือถ้าผู้ใช้งานใช้ฟังก์ชันที่ผิดประเภทโปรแกรมก็ยังไม่สามารถบอกได้ละเอียดนักว่าให้เราใช้ข้อมูลประเภทอะไร ซึ่งส่วนนี้เป็นส่วนที่ต้องปรับปรุงในเรื่องของการทำข้อความแจ้งเตือนเวลาผิดพลาด ซึ่งต้องพัฒนาในส่วนนี้ให้ดีขึ้นต่อไป โดยก่อนอื่นนั้นเราจึงได้จัดทำเอกสารคู่มือการใช้งานเพื่อให้ผู้ใช้สามารถรู้ได้ว่ามีฟังก์ชันใดบ้างที่สามารถใช้ได้
3. ในด้านความเร็วของโปรแกรมนั้นเมื่อมีการประมวลผลคำสั่งที่มีข้อมูลปริมาณมากๆนั้น ก็อาจทำให้ความเร็วนั้นไม่เร็วเท่าที่ควร ซึ่งจุดนี้ต้องทำการปรับปรุงเรื่องการจัดการทรัพยากร และนอกจากนั้นความเร็วของเครื่องผู้ใช้งานก็มีผลต่อความเร็วในการคำนวณเช่นกัน ดังนั้นเราจึงต้องกำหนดความต้องการขั้นต่ำของเครื่องที่ใช้เพื่อให้ การให้การใช้งานของผู้ใช้เป็นไปได้อย่างราบรื่นที่สุด
4. ในการพิมพ์ฟังก์ชันที่ต้องใช้การเขียนรูปประโยคหลายๆบรรทัดเช่น การวนลูป การประกาศฟังก์ชัน เวลาเราจะขึ้นบรรทัดใหม่จำเป็นต้องกดปุ่ม Shift ไว้ด้วย ซึ่งถ้าเราไม่ได้กดไว้และทำการขึ้นบรรทัดใหม่ โปรแกรมก็จะถือว่าที่เราพิมพ์ไปนั้นเป็นคำสั่งที่ต้องการประมวลผล ซึ่งส่งผลให้เกิดความผิดพลาดเพราะ เรายังเขียนฟังก์ชันไม่ครบตามที่เราต้องการ ซึ่งในส่วนนี้จำเป็นต้องปรับปรุงเพิ่มเติมในอนาคตเพื่อให้สามารถใช้งานได้ง่ายขึ้น
5. โปรแกรมนี้สนับสนุนทั้งภาษาไทยและอังกฤษ ซึ่งสามารถใช้เมนูเปลี่ยนภาษาเปลี่ยนได้ แต่ยังมีข้อจำกัดอยู่ที่เราจำเป็นต้องเริ่มการใช้งานโปรแกรมใหม่ทุกครั้งเพื่อให้โปรแกรมเปลี่ยนภาษาเป็นไปตามที่เราต้องการ ซึ่งในจุดนี้ยังเป็นจุดที่ต้องปรับปรุงเพื่อให้โปรแกรมสามารถปรับเปลี่ยนได้ทันที
6. ฟังก์ชันการสั่งพิมพ์ยังมีปัญหาในเรื่องของความเร็วอยู่ ซึ่งส่งผลให้บางครั้งเมื่อทำการสั่งพิมพ์นั้นทำให้โปรแกรมค้างไปชั่วเวลาหนึ่ง ซึ่งเป็นสิ่งที่ไม่ควรจะเกิดขึ้นและต้องนำไปปรับปรุงต่อไป
7. ภาพที่ได้จากการสั่งพิมพ์รูปนั้น ยังมีปัญหาเรื่องของความคมชัดของภาพที่ได้ออกมา ซึ่งภาพที่ได้นั้นจะไม่เหมือนกับที่แสดงจริงๆคือจะมีลักษณะภาพที่แตกและไม่ละเอียดอย่างที่ควรจะเป็น

รายการอ้างอิง

[1] ดร.วีระศักดิ์ ซึ้งถาวร(2557), Java Programming Volume II, ซีเอ็ดยูเคชั่นจำกัด

[2] ปราโมทย์ เดชะอำไพ(2547), ไฟไนต์เอลิเมนต์ในงานวิศวกรรม, สำนักพิมพ์แห่งจุฬาลงกรณ์มหาวิทยาลัย

[3] ผู้ช่วยศาสตราจารย์ ดร. สัลยุทธ์ สว่างวรรณ, คอมไพเลอร์, วิรัตน์ เอ็ดยูเคชั่น

[4] Craig Larma(2005), Applying UML and Patterns Third Edition, Prentice Hall Imprint of Pearson Education, Inc.

[5] David a Watt & Deryck F Brown(2000), Programmimg Language Processor in Java, Prentice Hall Imprint of Pearson Education, Inc.

[6] Donal Hearn and Miss Pauline Baker(20004), Computer Graphics with OpenGL Third Edition, Prentic Grap Imprint of Pearson Education, Inc.

[7] Leen Ammeraal(2004), Computer Graphics with OpenGL Third Edition for JavaProgrammers, Join Wiley & Sons,inc

[8] Thomas B. Bahder(1995), Mathematica for scientists and Engineers, Addison Wesley, Inc

ภาคผนวก ก.

คู่มือการใช้งาน

วิธีการใช้งานโปรแกรม

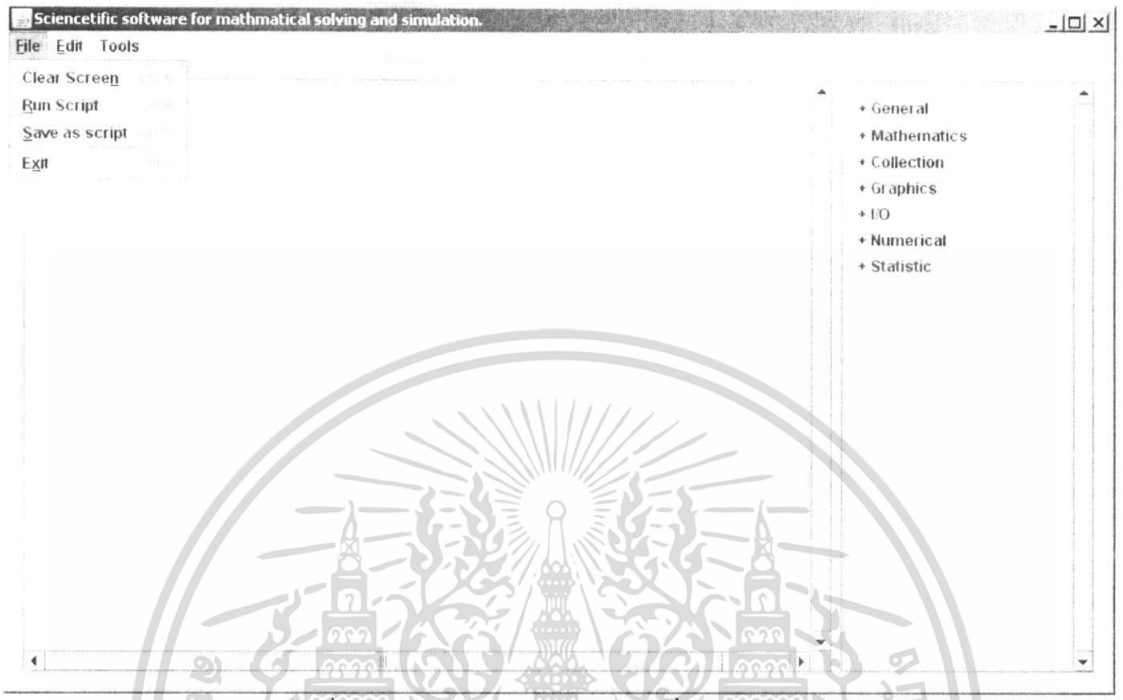
เมื่อเริ่มโปรแกรมขึ้นมานั้นจะพบอินเทอร์เฟส (Interface) ของโปรแกรมตามลักษณะดังภาพ ซึ่งแบ่งเป็นส่วนของเมนูบาร์ (Menu bar) ซึ่งประกอบไปด้วยคำสั่ง File, Edit และ Tools และส่วนของเท็กซ์เอเรีย (Text area) ซึ่งเป็นส่วนสำหรับการป้อนคำสั่งของโปรแกรมเรา และในส่วนของเมนูลัดฟังก์ชันทางด้านขวาของโปรแกรม ดังภาพ ก.1

ภาพที่ ก.1 แสดงหน้าจอโดยรวมของโปรแกรม

ซึ่งในส่วนของเมนูนั้นจะประกอบไปด้วยเมนูหลักๆ 3 เมนูคือ File, Edit และ Tools ซึ่งรายละเอียดของแต่ละเมนูนั้นจะกล่าวถึงต่อไปดังต่อไปนี้

ก.1 เมนูต่างๆในโปรแกรม

ก.1.1) เมนู File

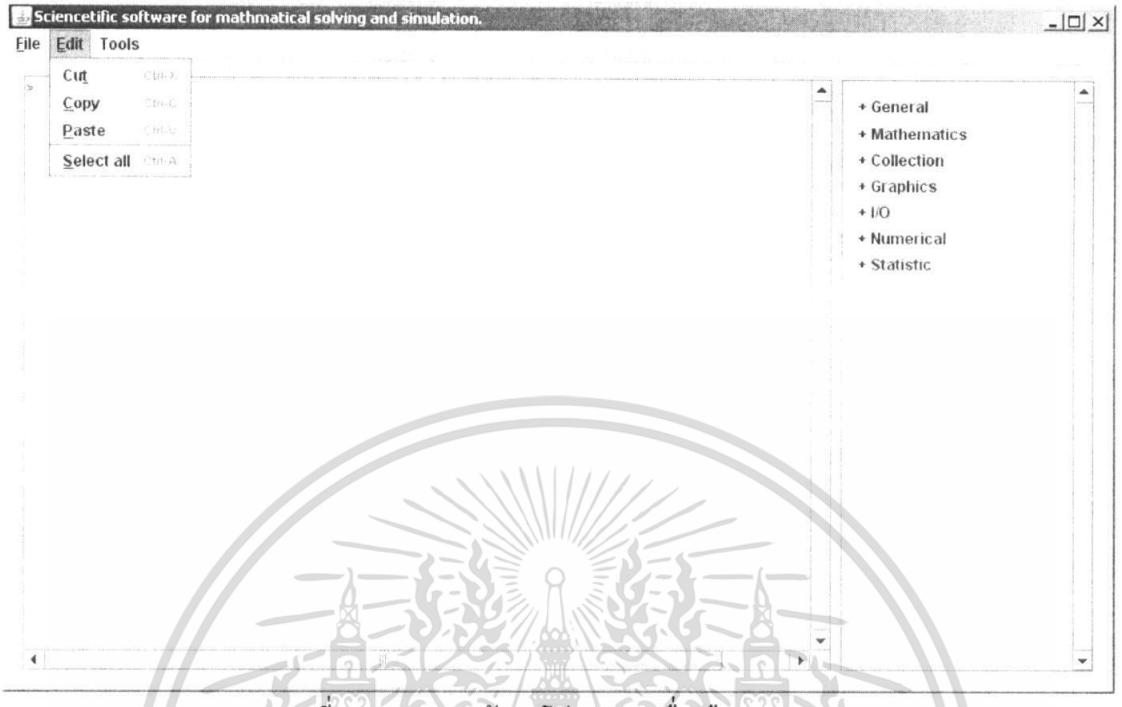


ภาพที่ ก.2 แสดงหน้าจอโปรแกรมเมื่อเลือกเมนู File

จากภาพเมื่อเราเลือกที่เมนู File นั้นเราจะพบฟังก์ชันต่างๆดังนี้

- Clear Screen เป็นฟังก์ชันสำหรับการล้างหน้าจอในส่วนของเท็กซ์เอเรียทั้งหมด
- Run Script เป็นฟังก์ชันสำหรับการเปิดสคริปต์ไฟล์
- Save as script เป็นฟังก์ชันสำหรับการเซฟคำสั่งที่เราพิมพ์ไปทั้งหมดลงไปยังสคริปต์ไฟล์
- Exit ออกจากโปรแกรม

ก.1.2) เมนู Edit

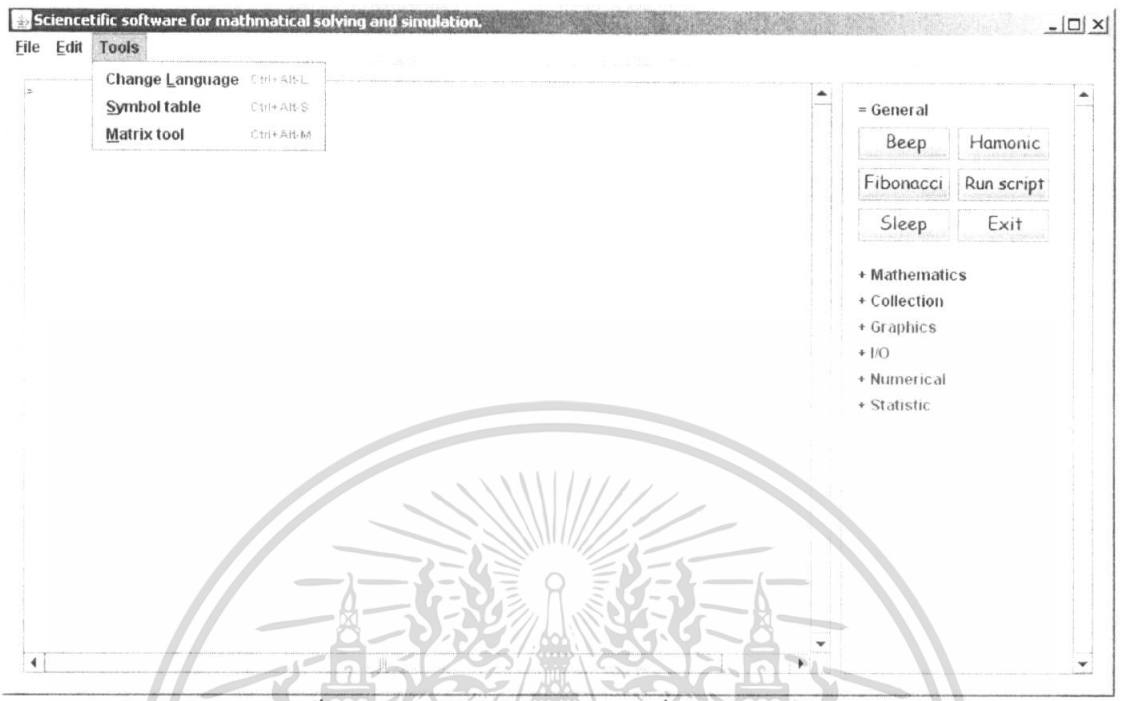


ภาพที่ ก.3 แสดงหน้าจอโปรแกรมเมื่อเลือกเมนู Edit

จากภาพเมื่อเราเลือกที่เมนู Edit นั้นเราจะพบฟังก์ชันต่างๆดังนี้

- Cut สำหรับคัดลอกข้อความแบบตัดออกไป
- Copy สำหรับคัดลอกข้อความ
- Paste สำหรับวางข้อความที่คัดลอกไว้
- Select ทำการเลือกข้อความทั้งหมด

ก.1.3 เมนู Tools



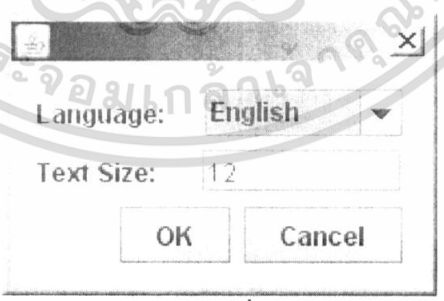
ภาพที่ ก.4 แสดงหน้าจอโปรแกรมเมื่อเลือกเมนู Tools

จากภาพเมื่อเราเลือกที่เมนู Edit นั้นเราจะพบฟังก์ชันต่างๆดังนี้

- Change Language

Language สำหรับเปลี่ยนภาษา ไทย/อังกฤษ

Text Size สำหรับการเปลี่ยนขนาดฟอนท์ในการแสดงผมข้อความในเท็กซ์เอเรีย

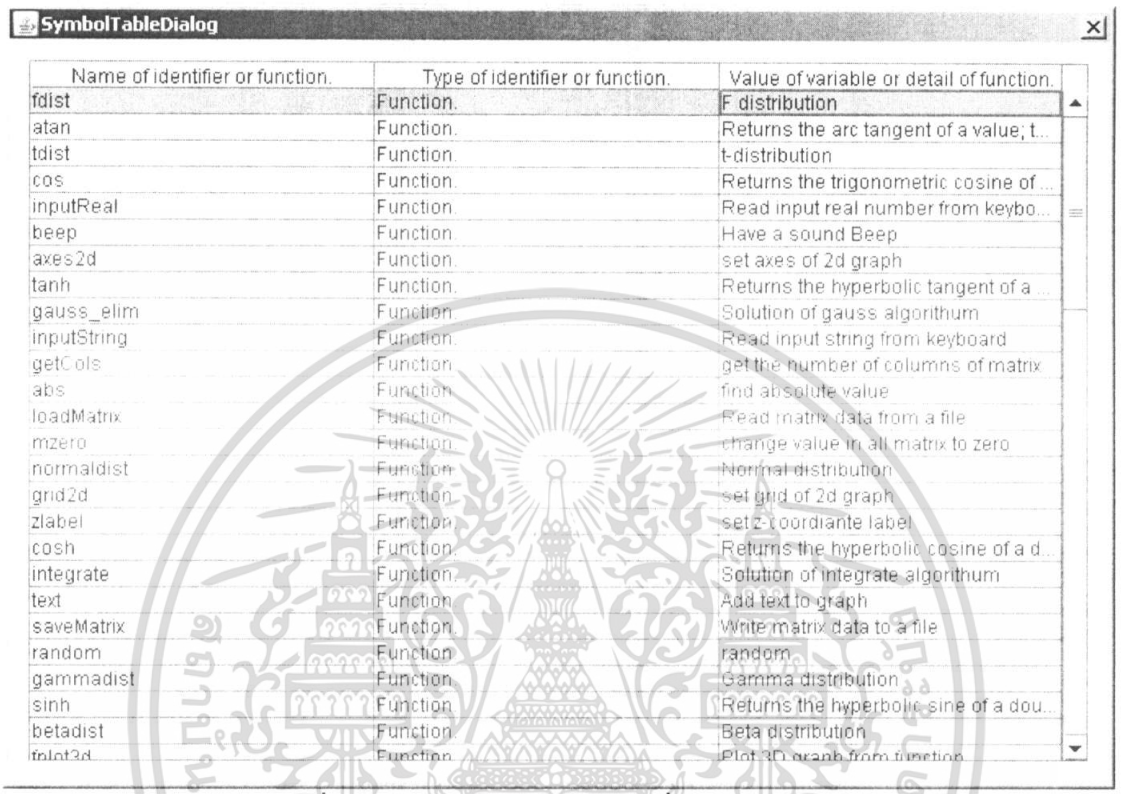


ภาพที่ ก.5 แสดงหน้าจอโปรแกรมเมื่อเลือกเมนู Change Language

สำหรับการเปลี่ยนภาษานั้นจำเป็นต้องมีการเริ่มโปรแกรมใหม่เพื่อให้แสดงผลที่ต้องการ

- Symbol table

ตารางแสดงรายชื่อ, ชนิดและค่าของตัวแปรหรือฟังก์ชันทั้งหมด ซึ่งเมื่อเลือกคำสั่งนี้ก็จะแสดงผลลัพธ์ดังภาพ



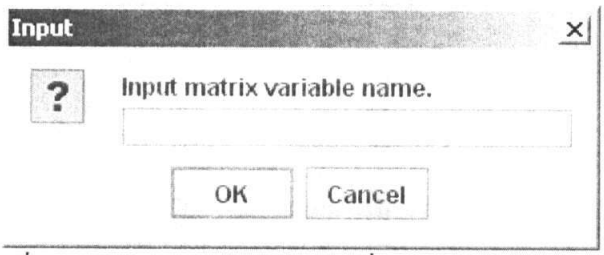
ภาพที่ ก.6 แสดงหน้าจอโปรแกรมเมื่อเลือกเมนู Symbol table

โดยที่ในส่วนภาษาของคำอธิบายฟังก์ชันนั้นจะเปลี่ยนไปตามภาษาที่เลือกใช้ในโปรแกรม ณ ขณะนั้น

- Matrix tool

เครื่องมือสำหรับการจัดการค่าในเมทริกซ์ที่ได้สร้างไว้แล้ว

ตัวอย่างเช่น เราได้สร้างเมทริกซ์ขึ้นมาตัวหนึ่งโดยกำหนดให้ a = {{1,2},{2,3}}



ภาพที่ ก.7 แสดงหน้าจอโปรแกรมเมื่อเลือกเมนู Matrix tool

จากภาพที่ ก.7 สมมติว่าเราต้องการจัดการกับเมทริกซ์ ที่เราได้สร้างไว้ข้างต้น ก็ให้เราทำการใส่ชื่อเมทริกซ์ที่ตั้งไป ซึ่ง ในที่นี้ก็คือ a

ภาพที่ ก.8 แสดงหน้าจอโปรแกรม Matrix Viewer

จากภาพที่ ก.8 โปรแกรมจะแสดงค่าภายในเมทริกซ์ที่เราได้สร้างไว้ให้เราดู โดยเราสามารถทำการแก้ไขค่าในตารางนี้ได้ทันที

ก.1.4 เมนูลัดสำหรับการใช้ฟังก์ชัน

เพื่อความสะดวกสำหรับผู้ใช้งาน โปรแกรมได้จัดเตรียมเมนูฟังก์ชันต่างๆทั้งหมดของโปรแกรมเอาไว้ให้ ซึ่งสามารถเห็นได้ทางขวาของโปรแกรม ซึ่งในส่วนของเมนูลัดนี้ จะแบ่งออกเป็นหมวดหมู่ขึ้นอยู่กับประเภทของฟังก์ชันที่ต้องการใช้งาน ดังนี้

- General ฟังก์ชันทั่วไป
- Mathematics ฟังก์ชันทางคณิตศาสตร์พื้นฐาน
- Collection ฟังก์ชันข้อมูลชนิดคอลเลคชัน
- Graphics ฟังก์ชันการแสดงผลทางกราฟฟิก
- I/O ฟังก์ชัน I/O
- Numerical ฟังก์ชันสำหรับการดำเนินการเชิงตัวเลข
- Statistic ฟังก์ชันทางสถิติ

ภาพที่ ก.9 แสดงตัวอย่างหน้าจอพร้อมเมนูลัดสำหรับการใช้งานฟังก์ชัน

การใช้งานก็เพียงคลิกที่ฟังก์ชันที่ต้องการ รายชื่อของฟังก์ชันที่เลือกนั้นก็จะไปแสดงในส่วนเทกซ์เอเรีย พร้อมด้วยค่าพารามิเตอร์ที่ต้องใช้สำหรับฟังก์ชันนั้น โดยจะสื่อเป็นตัวหนังสือเพื่อให้ผู้ใช้เข้าใจในระดับหนึ่งว่า พารามิเตอร์นั้นต้องใช้ค่าประเภทใด อย่างในภาพที่ ก.9 เราเลือกฟังก์ชัน fplot2d โปรแกรมก็จะแสดงชื่อฟังก์ชันและพารามิเตอร์คือ fplot2d("func_name", xMin, xMax, xInc) ซึ่งหมายความว่าต้องใช้ 4 พารามิเตอร์ เราจำเป็นต้องใส่ให้ครบตามนี้

ในส่วนของรายละเอียดของแต่ละฟังก์ชันนั้นจะได้กล่าวถึงในหัวข้อถัดไป

ก.2 ฟังก์ชั่นที่ใช้ได้ในโปรแกรม

ในส่วนนี้จะอธิบายถึงฟังก์ชันต่างๆในโปรแกรม โดยจะแสดงชนิดของการคืนค่า ชนิดข้อมูลที่รับ และผลลัพธ์ที่ได้จากการใช้ฟังก์ชันนั้นๆ

ก.2.1 ฟังก์ชันทั่วไป (General Function)

beeb

String beep()

Return

เสียงเตือน

fibonacci

Real fibonacci(Real r)

Parameter

r::= จำนวนจริงใดๆ

Return

ค่าของฟังก์ชั่นฟิโบนักซี

hamonic

Real fibonacci(Real r)

Parameter

r::= จำนวนจริงใดๆ

Return

ค่าของฟังก์ชั่นฮาโมนิก

exit

void exit()

Return

จบการทำงาน

sleep

void sleep(Real time)

Parameter

time::= ค่าของเวลาเป็นจำนวนจริงใดๆ (หน่วยเป็นวินาที)

Return

หยุดรอตามเวลาที่กำหนด

run

String run(String file_name)

Parameter

file_name::= ชื่อสคริปท์ไฟล์ที่ต้องการโหลด โดยต้องระบุ Path ของไฟล์ให้ชัดเจน สมมติเช่นไฟล์อยู่ในไดรฟ์ c: ก็ต้องระบุเป็น "c:\path\file_name" สังเกตว่าเราต้องมีการใส่เครื่องหมาย "" ปิดหัวปิดท้ายด้วย เพราะข้อมูลที่รับนั้นเป็น String

Return

ทำการรันสคริป

ก.2.2 ฟังก์ชันทางคณิตศาสตร์พื้นฐาน (Mathematics Function)

sin

Real sin(Real r)

Parameter

r::= ค่าของมุม (หน่วยเป็นเรเดียน)

Return

คืนค่าไซน์ในตรีโกณมิติ

cos

Real cos(Real r)

Parameter

r::= ค่าของมุม (หน่วยเป็นเรเดียน)

Return

คืนค่าโคไซน์ในตรีโกณมิติ

tan

Real tan(Real r)

Parameter

r::= ค่าของมุม (หน่วยเป็นเรเดียน)

Return

คืนค่าแทนเจนท์ในตรีโกณมิติ

asin

Real asin(Real r)

Parameter

r::= ค่าของมุม (หน่วยเป็นเรเดียน)

Return

คืนค่าของอาร์คไซน์ โดยจะอยู่ในช่วง -pi/2 ถึง pi/2

acos

Real acos(Real r)

Parameter

r::= ค่าของมุม (หน่วยเป็นเรเดียน)

Return

คืนค่าของอาร์คโคไซน์ โดยจะอยู่ในช่วง -pi/2 ถึง pi/2

atan

Real atan(Real r)

Parameter

r::= ค่าของมุม (หน่วยเป็นเรเดียน)

Return

คืนค่าของอาร์คแทนเจนทน์ โดยจะอยู่ในช่วง -pi/2 ถึง pi/2

cuberoot

Real cuberoot(Real r)

Parameter

r::= จำนวนจริงใดๆ

Return

ค่าของรากที่สาม

ceil

Real ceil(Real r)

Parameter

r::= จำนวนจริงใดๆ

Return

ทำการปัดเศษขึ้น

floor

Real floor(Real r)

Parameter

r::= จำนวนจริงใดๆ

Return

ทำการปัดเศษลง

sinh

Real sinh(Real r)

Parameter

r::= ค่าของมุม (หน่วยเป็นเรเดียน)

Return

ค่าของไฮเปอร์โบลิกไซน์

cosh

Real cosh(Real r)

Parameter

r::= ค่าของมุม (หน่วยเป็นเรเดียน)

Return

ค่าของไฮเปอร์โบลิกโคไซน์

tanh

Real tanh(Real r)

Parameter

r::= ค่าของมุม (หน่วยเป็นเรเดียน)

Return

ค่าของไฮเปอร์โบลิกแทนเจนท์

exp

Real exp(Real r)

Parameter

r::= จำนวนจริงใดๆ

Return

ค่าของออยเลอร์ อยู่ในรูป

loge

Real loge(Real r)

Parameter

r::= จำนวนจริงใดๆ

Return

คืนค่าของลอการิทึมฐานธรรมชาติ(e)

log

Real log(Real r)

Parameter

r::= จำนวนจริงใดๆ

Return

คืนค่าของลอการิทึมฐาน 10

sqrt

Real sqrt(Real r)

Parameter

r::= จำนวนจริงใดๆ

Return

ค่ารากที่สอง

max

Real max(Real r1 ,Real r2)

Parameter

r1::= จำนวนจริงใดๆ

r2::= จำนวนจริงใดๆ

Return

ค่าสูงสุดจากจำนวนสองจำนวน

min

Real min(Real r1,Real r2)

Parameter

r1::= จำนวนจริงใดๆ

r2::= จำนวนจริงใดๆ

Return

ค่าต่ำสุดจากจำนวนสองจำนวน

random

Real random()

Return

สุ่มค่าออกมาเป็นเลขทศนิยม

abs

Real abs(Real r)

Parameter

r::= จำนวนจริงใดๆ

Return

คืนค่าสมบูรณ์

sum

Real sum(String func_name, Real r1 ,Real r2)

Parameter

func_name::= ชื่อของฟังก์ชั่น โดยฟังก์ชันที่ใช้นั้นต้องกำหนดเป็นฟังก์ชันชนิดคืนค่าและรับพารามิเตอร์เพียงค่าเดียว

r1::= จำนวนจริงใดๆ ค่าต่ำสุดที่จะเริ่มคำนวณ

r2::= จำนวนจริงใดๆ เป็นค่าสูงสุดที่จะนำมาคำนวณ

Return

ค่าผลรวมของฟังก์ชั่น

ก.2.3 ฟังก์ชันข้อมูลชนิดคอลเลคชัน (Collection Function)

matabs

Matrix matabs(Matrix m)

Parameter

m::= เมทริกซ์ใดๆ

Return

ทำให้ค่าในเมทริกซ์เป็นค่าบวกทั้งหมด

det

Matrix det(Matrix m)

Parameter

m::= เมทริกซ์ โดยที่ต้องเป็นเมทริกซ์ที่มีขนาดของแถวและหลักที่เท่ากันด้วย

Return

ค่าของดีเทอร์มิแนนท์

transpose

Matrix transpose(Matrix m)

Parameter

m::= เมทริกซ์ใดๆ

Return

ทำการสลับตำแหน่งของเมทริก

mzero

Matrix mzero(Matrix m)

Parameter

m::= เมทริกซ์ใดๆ

Return

เปลี่ยนทุกค่าในเมทริกให้เท่ากับศูนย์

getRows

Real getRows(Matrix m)

Parameter

m::= เมทริกซ์ใดๆ

Return

ค่าจำนวนแถวของเมทริกซ์นั้นๆ

getCols

Real getCols(Matrix m)

Parameter

m::= เมทริกซ์ใดๆ

Return

ค่าจำนวนหลักองเมทริกซ์นั้นๆ

arrayLen

Real arrayLen(Array arr)

Parameter

arr::= อะเรย์ใดๆ

Return

จำนวนพารามิเตอร์ในอะเรย์นั้นๆ

arrayToMat

Matrix arrayToMat(Array arr)

Parameter

arr::= อะเรย์ใดๆ

Return

แปลงอะเรย์ให้เป็นเมทริกซ์ขนาด 1xN เมื่อ N คือจำนวนพารามิเตอร์ที่อยู่ในอะเรย์นั้นๆ

meshgrid

Matrixl meshgrid(String func_name, Real rMin, Real rMax, Real inc)

Parameter

function_name::= ชื่อของฟังก์ชัน โดยฟังก์ชันที่ใช้นั้นต้องกำหนดเป็นฟังก์ชันชนิดรับค่าและคืนค่า 2 ตัวแปร

rMin::= ค่าต่ำสุดที่นำไปคำนวณ

rMax::= ค่าสูงสุดที่นำไปคำนวณ

inc::= ค่าที่เพิ่มขึ้นในแต่ละรอบ

Return

เมทริกซ์ที่เก็บข้อมูลสำหรับกราฟ 3 มิติ

ก.2.4 ฟังก์ชันการแสดงผลทางกราฟฟิก (Graphic Function)

plot2d

String plot2d(Matrix m)

Parameter

m::= เมทริกซ์ โดยที่ในการวาดกราฟนั้นจะอ้างอิงจากจุดดังนี้

m[N,0] แทนค่าที่ตำแหน่ง x

m[N,1] แทนค่าที่ตำแหน่ง y

Return

วาดกราฟ 2 มิติ

fplot2d

String fplot2d(String func_name, Real xMin, Real xMax, Real xInc)

Parameter

func_name::= ชื่อของฟังก์ชัน โดยฟังก์ชันที่ใช้นั้นต้องกำหนดเป็นฟังก์ชันชนิดรับและคืนค่าพารามิเตอร์ 1 ค่า

xMin::= ค่าในช่วงต่ำสุดที่นำไปคำนวณ

xMax::= ค่าในช่วงสูงสุดที่นำไปคำนวณ

xInc::= ค่าที่เพิ่มขึ้นในแต่ละรอบการคำนวณ

Return

วาดกราฟ 2 มิติจากฟังก์ชั่น

lplot3d

String lplot3d(Matrix m)

Parameter

m::= เมทริกซ์ โดยที่ในการวาดกราฟนั้นจะอ้างอิงจากจุดดังนี้

m[N,0] แทนค่าที่ตำแหน่ง x

m[N,1] แทนค่าที่ตำแหน่ง y

m[N,2] แทนค่าที่ตำแหน่ง z

Return

วาดกราฟ 3 มิติจากข้อมูลเชิงเส้น

plot3d

String plot3d (Matrix m, Real xMax, Real zMax)

Parameter

m::= เมทริกซ์ โดยที่ในการวาดกราฟนั้นจะอ้างอิงจากจุดดังนี้

m[N,0] แทนค่าที่ตำแหน่ง x

m[N,1] แทนค่าที่ตำแหน่ง y

m[N,2] แทนค่าที่ตำแหน่ง z

xMax::= ค่า x สูงสุดเป็นขอบเขตที่จะวาดกราฟ

zMax::= ค่า z สูงสุดเป็นขอบเขตที่จะวาดกราฟ

Return

วาดกราฟ 3 มิติ

fplot3d

String fplot3d (String f_name, Real xMin, Real xMax, Real xInc, Real zMin, Real zMax, zInc)

Parameter

f_name::= ชื่อของฟังก์ชัน โดยฟังก์ชันที่ใช้นั้นต้องกำหนดเป็นฟังก์ชันชนิดรับและคืนค่าพารามิเตอร์ 2 ค่า

xMin::= ค่า x ในช่วงต่ำสุดที่นำไปคำนวณ

xMax::= ค่า x ในช่วงสูงสุดที่นำไปคำนวณ

xInc::= ค่า x ที่เพิ่มขึ้นในแต่ละรอบการคำนวณ

zMin::= ค่า z ในช่วงต่ำสุดที่นำไปคำนวณ

zMax::= ค่า z ในช่วงสูงสุดที่นำไปคำนวณ

zInc::= ค่า z ที่เพิ่มขึ้นในแต่ละรอบการคำนวณ

Return

วาดกราฟ 3 มิติจากฟังก์ชั่น

axes2d

String axes2d(Real xMin, Real xMax, Real yMin, Real yMax)

Parameter

xMin::= ค่าแกน x ที่ต่ำสุด

xMax::= ค่าแกน x ที่สูงสุด

yMin::= ค่าแกน y ที่ต่ำสุด

yMax::= ค่าแกน y ที่สูงสุด

Return

การกำหนดแกนบนกราฟ 2 มิติ

axes3d

String axes3d(Real xMin, Real xMax, Real yMin, Real yMax, Real zMin, Real zMax)

Parameter

xMin::= ค่าแกน x ที่ต่ำสุด

xMax::= ค่าแกน x ที่สูงสุด

yMin::= ค่าแกน y ที่ต่ำสุด

yMax::= ค่าแกน y ที่สูงสุด

zMin::= ค่าแกน z ที่ต่ำสุด

zMax::= ค่าแกน z ที่สูงสุด

Return

การกำหนดแกนบนกราฟ 3 มิติ

title

String title(String name)

Parameter

name::= ชื่อกราฟที่ต้องการกำหนด

Return

กำหนดชื่อของกราฟ 2 มิติ

xlabel

String xlabel(String name)

Parameter

name::= ชื่อแกน x ที่ต้องการกำหนด

Return

กำหนดชื่อให้แก่แกน x

yabel

String ylabel(String name)

Parameter

name::= ชื่อแกน y ที่ต้องการกำหนด

Return

กำหนดชื่อให้แก่แกน y

zlabel

String zlabel(String name)

Parameter

name::= ชื่อแกน z ที่ต้องการกำหนด

Return

กำหนดชื่อให้แก่แกน z

text

String title(Real xPos, Real yPos, Real zPos, String msg)

Parameter

xPos::= ตำแหน่งบนแกน x

xPos::= ตำแหน่งบนแกน y

xPos::= ตำแหน่งบนแกน z

msg::= ข้อความที่ต้องการแสดง

Return

เพิ่มข้อความไปในกราฟบนตำแหน่งที่ระบุไว้

grid2d

String grid2d(Real nx, Real ny)

Parameter

nx::= ขนาดของกริดในแกน x

ny::= ขนาดของกริดในแกน y

Return

กำหนดขนาดของกริดในกราฟ 2 มิติ

grid3d

String grid3d(Real nx, Real ny, Real nz)

Parameter

nx::= ขนาดของกริดในแกน x

ny::= ขนาดของกริดในแกน y

nz::= ขนาดของกริดในแกน z

Return

กำหนดขนาดของกริดในกราฟ 3 มิติ

gfxclean

String gfxclear()

Return

ทำการเปลี่ยนค่าทุกอย่างที่เกี่ยวกับกราฟิกกลับไปยังค่าเริ่มต้น

ก.2.5ฟังก์ชัน I/O (I/O Function)

inputReal

Real inputReal(String msg)

Parameter

msg::= ข้อความที่ต้องการให้แสดงตอนขึ้นหน้าจอรับค่าข้อมูล

Return

ค่าจำนวนจริงที่รับจากคีย์บอร์ด

inputComplex

Complex inputComplex(String msg)

Parameter

msg::= ข้อความที่ต้องการให้แสดงตอนขึ้นหน้าจอรับค่าข้อมูล

Return

ค่าจำนวนเชิงซ้อนที่รับจากคีย์บอร์ด

inputString

String inputString(String msg)

Parameter

msg::= ข้อความที่ต้องการให้แสดงตอนขึ้นหน้าจอรับค่าข้อมูล

Return

ข้อความที่รับจากคีย์บอร์ด

loadMatrix

Real loadMatrix(String file_name)

Parameter

file_name::= ชื่อไฟล์ที่เก็บค่าเมทริกซ์ที่ต้องการโหลด โดยต้องระบุ Path ของไฟล์ให้ชัดเจน สมมติเช่นไฟล์อยู่ในไดรฟ์ c: ก็ต้องระบุเป็น "c:\path\file_name" สังเกตว่าเราต้องมีการใส่เครื่องหมาย "" ปิดหัวปิดท้ายด้วย เพราะข้อมูลที่รับนั้นเป็น String

Return

ค่าจำนวนจริงที่รับจากคีย์บอร์ด

saveMatrix

Real loadMatrix(Matrix m, String file_name)

Parameter

m::= ชื่อเมทริกซ์ที่ต้องการเซฟ

file_name::= ชื่อไฟล์ที่ต้องการเก็บ โดยเมทริกซ์ที่ถูกเซฟจะไปอยู่ที่โฟลเดอร์เดียวกับที่ติดตั้งโปรแกรมไว้

Return

บันทึกค่าเมทริกซ์ลงในไฟล์

ก.2.6 ฟังก์ชันสำหรับการดำเนินการเชิงตัวเลข (Numerical Function)

bisection

Real bisection(String func_name, Real lowerBounds, Real upperBounds, Real epsilon)

Parameter

func_name::= ชื่อฟังก์ชัน

lowerBounds::= ค่าขอบล่าง

upperBound::= ค่าขอบบน

epsilon::= ค่าความผิดพลาด

Return

ค่ารากของสมการที่อยู่ในช่วงแบบไบเซ็คชั่น

integrate

Real bisection(String func_name, Real lowerLimit, Real upperLimit, Real epsilon, Real n)

Parameter

func_name::= ชื่อฟังก์ชัน

lowerLimits::= ค่าจำกัดขอบเขตล่าง

upperLimit::= ค่าจำกัดขอบเขตบน

epsilon::= ค่าความผิดพลาด

n::= จำนวนรอบที่ต้องการคำนวณ

Return

ค่ารากของสมการที่อยู่ในช่วงแบบแทรปไซดอล อินทริเกชั่น

gauss_elim

Matrix gauss_elim(Matrix m, Matrix n)

Parameter

m::= เมทริกซ์ใดๆ

n::= เมทริกซ์ใดๆ

Return

เมทริกซ์ที่ผ่านวิธีแก้ปัญหาการกำจัดตัวแปรแบบเกาส์

polym_interpolate

Real polym_interpolate(Real r, Matrix m)

Parameter

r::= จำนวนจริงใดๆ

m::= เมทริกซ์

Return

วิธีแก้ปัญหาแบบโพลิโนเมี่ยล

ludecompos

Matrix ludecompos(Matrix m,Matrix n)

Parameter

m::= เมทริกซ์ขนาด MxN

n::= เมทริกซ์ขนาด Nx1

ค่าจำนวนหลักของเมทริกซ์ m นั้นต้องเท่ากับจำนวนแถวในเมทริกซ์ n

Return

วิธีแก้ปัญหาแบบ แอลยูดีคอมโพสิชั่น

ก.2.7 ฟังก์ชันทางสถิติ (Statistic Function)

uniformdist

Real uniformdist(Real r1, Real r2)

Parameter

r1::= จำนวนจริงใดๆ

r2::= จำนวนจริงใดๆ

Return

การกระจายแบบยูนิฟอร์ม

benullidist

Real benullidist(Real r)

Parameter

r::= จำนวนจริงใดๆ

Return

การกระจายแบบเบนูลี่

binomialdist

Real binomialdist(Real r1, Real r2)

Parameter

r1::= จำนวนจริงใดๆ

r2::= จำนวนจริงใดๆ

Return

การกระจายแบบไบนอมอล

geodist

Real geodist(Real r)

Parameter

r::= จำนวนจริงใดๆ

Return

การกระจายแบบจีโอเมทริก

negbindist

Real negbindist(Real r1, Real r2)

Parameter

r1::= จำนวนจริงใดๆ

r2::= จำนวนจริงใดๆ

Return

การกระจายแบบเนกาทีฟไบนอมอ

poissondist

Real poissondist(Real r)

Parameter

r::= จำนวนจริงใดๆ

Return

การกระจายแบบพอสสัน

gammadist

Real gammadist(Real r1, Real r2)

Parameter

r1::= จำนวนจริงใดๆ

r2::= จำนวนจริงใดๆ

Return

การกระจายแบบแกรมม่า

weibulldist

Real weibulldist(Real r1, Real r2)

Parameter

r1::= จำนวนจริงใดๆ

r2::= จำนวนจริงใดๆ

Return

การกระจายแบบเวลบลู

snormaldist

Real snormaldist()

Return

การกระจายแบบนอมอลอย่างง่าย

normaldist

Real normaldist(Real r1, Real r2)

Parameter

r1::= จำนวนจริงใดๆ

r2::= จำนวนจริงใดๆ

Return

การกระจายแบบนอมอล

chisqrdist

Real chisqrdist(Real r)

Parameter

r::= จำนวนจริงใดๆ

Return

การกระจายแบบไคสแควร์

tdist

Real tdist (Real r)

Parameter

r::= จำนวนจริงใดๆ

Return

การกระจายแบบ T

lognormaldist

Real lognormaldist (Real r1, Real r2)

Parameter

r1::= จำนวนจริงใดๆ

r2::= จำนวนจริงใดๆ

Return

การกระจายแบบล๊อคนอมอล

betadist

Real betadist(Real r1, Real r2)

Parameter

r1::= จำนวนจริงใดๆ

r2::= จำนวนจริงใดๆ

Return

การกระจายแบบเบต้า

fdist

Real betadist(Real r1, Real r2)

Parameter

r1::= จำนวนจริงใดๆ

r2::= จำนวนจริงใดๆ

Return

การกระจายแบบเอฟ

ก.3 การสร้างตัวแปรแบบคอลเล็คชั่น

เราสามารถสร้างตัวแปรแบบคอลเล็คชั่นโดยการใช้เครื่องหมายปีกกา "{ }" เป็นตัวแบ่งขอบเขตของข้อมูลโดยมีเครื่องหมายลูกน้ำ "," เป็นตัวแบ่งข้อมูล โดยเราสามารถสร้างตัวแปรแบบคอลเล็คชั่นได้ 2 รูปแบบคือ

ก.3.1 อะเรย์

มีรูปแบบดังนี้

varName = {val1, val2, …, val N}

ตัวอย่างเช่น

> a = {1,2,3}

ผลลัพธ์: 1.0, 2.0, 3.0

ก.3.2 เมทริกซ์

มีรูปแบบดังนี้

varName = { {val_01, val_02, …, val_0N}, { val_11, val_12, …, val_1N }, …, { val_N1, val_N2, …, val_NN } }

ตัวอย่างเช่น

> b = { {1,2}, {3}, {4,5,6} }

ผลลัพธ์: [0, n]

1.0, 2.0, 0.0

[1, n]

3.0, 0.0, 0.0

[2, n]

4.0, 5.0, 6.0

ก.4 การสร้างฟังก์ชัน

เราสามารถประกาศฟังก์ชันได้ 5 รูปแบบดังนี้

ก.4.1 ฟังก์ชันชนิดคืนค่าเดียว

มีรูปแบบดังนี้

function returnName = funcName(param_list)

Statement sequence;

End function

ตัวอย่างเช่น

```
> function x = f(y,z)
x = y + z + 1;
end function
```

ผลลัพธ์: ชื่อของฟังก์ชั่น: "f", ประเภทของฟังก์ชั่น: ฟังก์ชั่นของผู้ใช้งาน.

ตัวอย่างการเรียกใช้งานฟังก์ชันที่สร้างขึ้นมา

```
> f(10,10)
```

ผลลัพธ์: [21.0]

ก.4.2 ฟังก์ชันชนิดคืนค่าหลายค่า

มีรูปแบบดังนี้

Function [return_name_list] = funcName(param_list)

Statement sequence;

End function

ตัวอย่างเช่น

```
> function [x,y] = f(a,b)
x = b^a + 1;
y = a*b + 2;
end function
```

ผลลัพธ์: ชื่อของฟังก์ชั่น: "f", ประเภทของฟังก์ชั่น: ฟังก์ชั่นของผู้ใช้งาน.

ตัวอย่างการเรียกใช้งานฟังก์ชันที่สร้างขึ้นมา

> f(2.3)

ผลลัพธ์: [10.0, 8.0]

ก.4.3 การประกาศฟังก์ชันแบบ Expression

มีรูปแบบดังนี้

Function funcName(param_list) = expression end function

ตัวอย่างเช่น

> function f(x) = sin(x) end function

ผลลัพธ์: ชื่อของฟังก์ชั่น: "f", ประเภทของฟังก์ชั่น: ฟังก์ชั่นของผู้ใช้งาน.

ตัวอย่างการเรียกใช้งานฟังก์ชันที่สร้างขึ้นมา

> f(1)

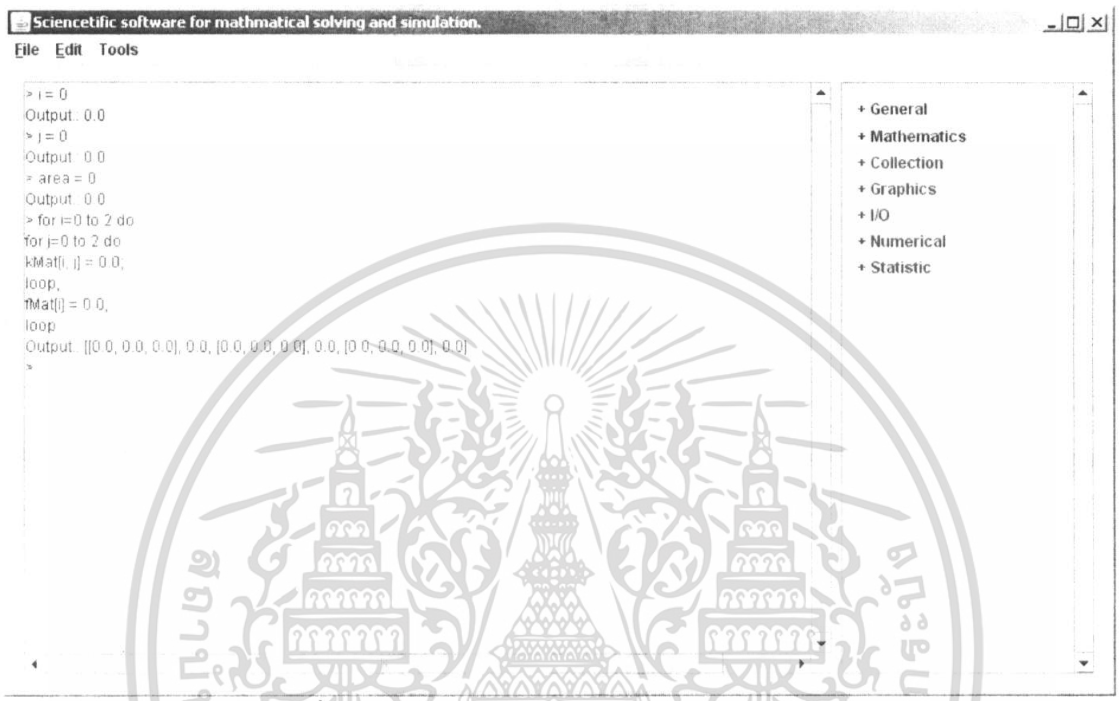
ผลลัพธ์: 0.8414709848078965

ก.5 การเรียกใช้งานจากสคริปต์ไฟล์

เราสามารถทำการเรียกใช้งานฟังก์ชันจากสคริปต์ไฟล์ที่สร้างไว้แล้วได้โดยการ เลือกที่ File > Run Script และเลือกไฟล์ที่ได้ทำการสร้างไว้ โดยรูปแบบของไฟล์ที่ใช้นั้นจะเซฟเป็นสกุล .mvs โดยรูปแบบของไฟล์สคริปต์นั้น สามารถเขียนได้ดังตัวอย่างเช่น

```
i = 0$
j = 0$
area = 0$
for i=0 to 2 do
for j=0 to 2 do
kMat[i, j] = 0.0;
loop;
fMat[i] = 0.0;
loop$
```

ซึ่งรูปแบบของการเขียนสคริปต์ไฟล์ จะมีเครื่องหมาย ดอลล่าไซน์ ($) ปิดท้ายคำสั่งในแต่ละคอมมานด์ ที่ต้องการจะให้แสดงผล ดังตัวอย่างในรูป จะเห็นว่า โปรแกรมจะแสดง output ในทุกคำสั่งที่มีการเขียนเครื่องหมายดอลล่าไซน์ปิดท้าย ซึ่งเปรียบเสมือนว่าเครื่องหมายดอลล่าไซน์ที่ปิดท้ายแต่ละบรรทัดนั้นก็คือการกดเอนเทอร์ (Enter) ในโปรแกรมนั่นเอง



ภาพที่ ก.10 แสดงตัวอย่างหลังจากการรันสคริปต์ไฟล์

ภาคผนวก ข.

คู่มือการติดตั้งโปรแกรม

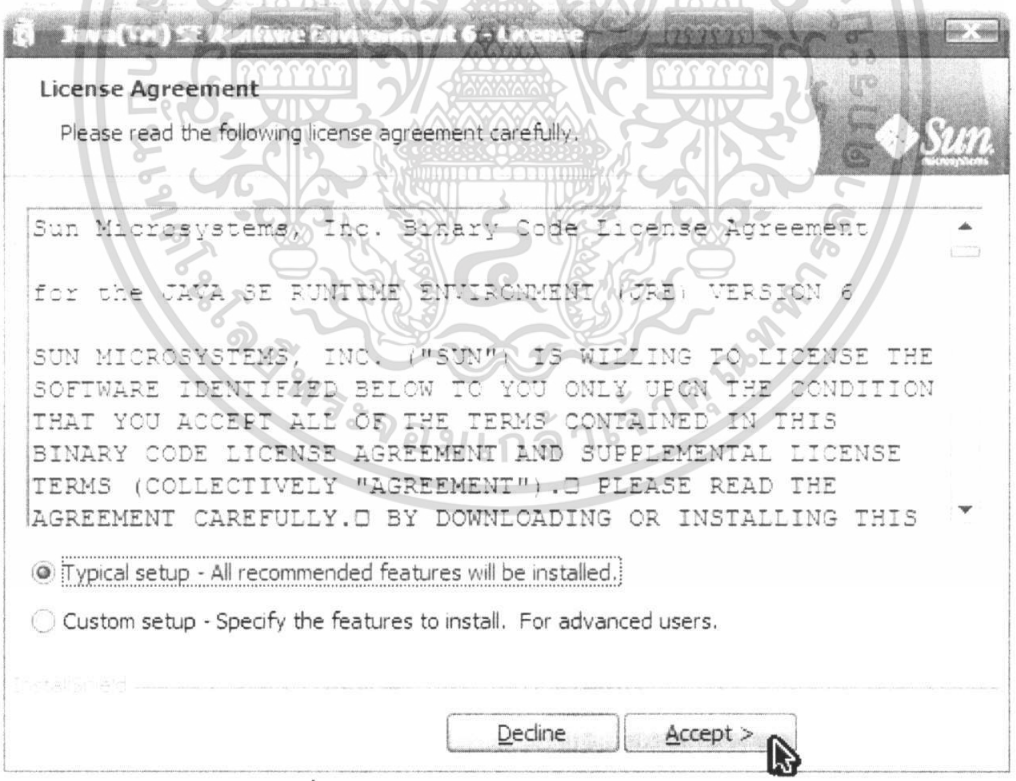
สำหรับการใช้งานโปรแกรม MatVis นั้นจำเป็นต้องมีการติดตั้ง Java Runtime Environment (JRE) ก่อนจึงจะสามารถใช้งานได้ ซึ่งในส่วนนี้จะแนะนำถึงวิธีการติดตั้งโปรแกรมในส่วนต่างๆเพื่อให้สามารถใช้งานได้อย่างสมบูรณ์ ซึ่งในส่วนนี้จะขอกล่าวถึงขั้นตอนติดตั้งสำหรับการใช้งานบนระบบปฏิบัติการวินโดวส์เท่านั้น

ข.1 วิธีการติดตั้ง Java(TM) SE Runtime Environment

- ทำการดับเบิ้ลคลิกที่ไฟล์ jre-6-windows-i586.exe ดังที่แสดงในภาพ ข.1

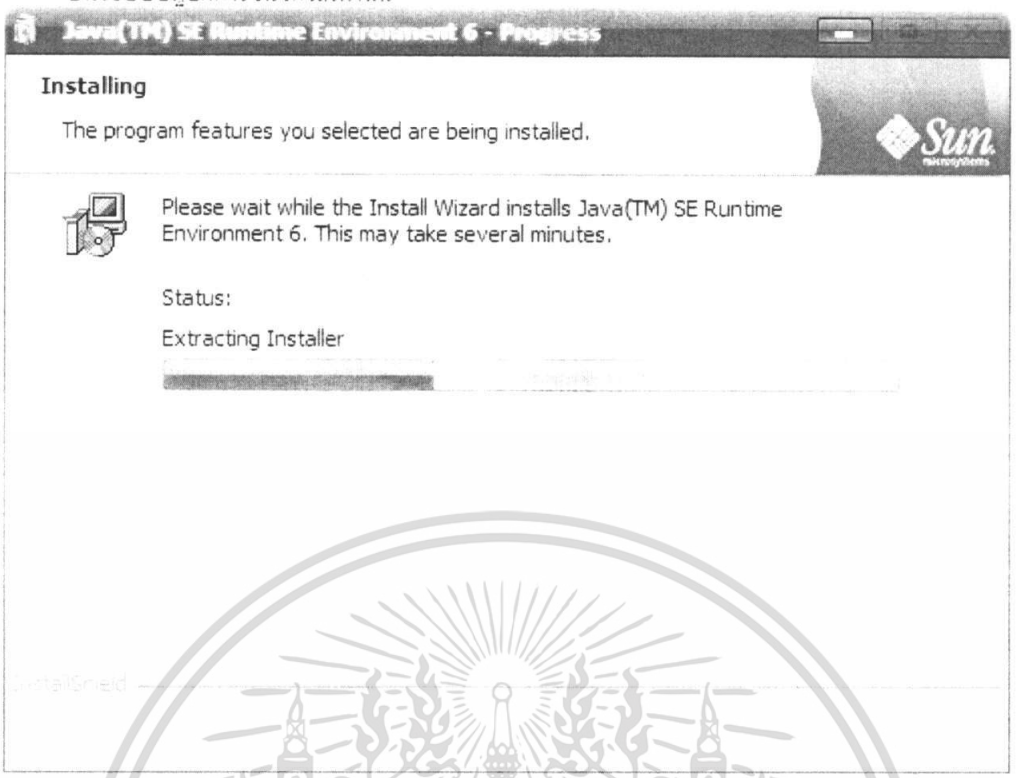
ภาพที่ ข.1 แสดงไฟล์สำหรับติดตั้ง JRE

- เมื่อเข้าสู่หน้า License Agreement เลือก Typical setup หลังจากนั้นกดปุ่ม Accept ดังที่แสดงในภาพ ข.2

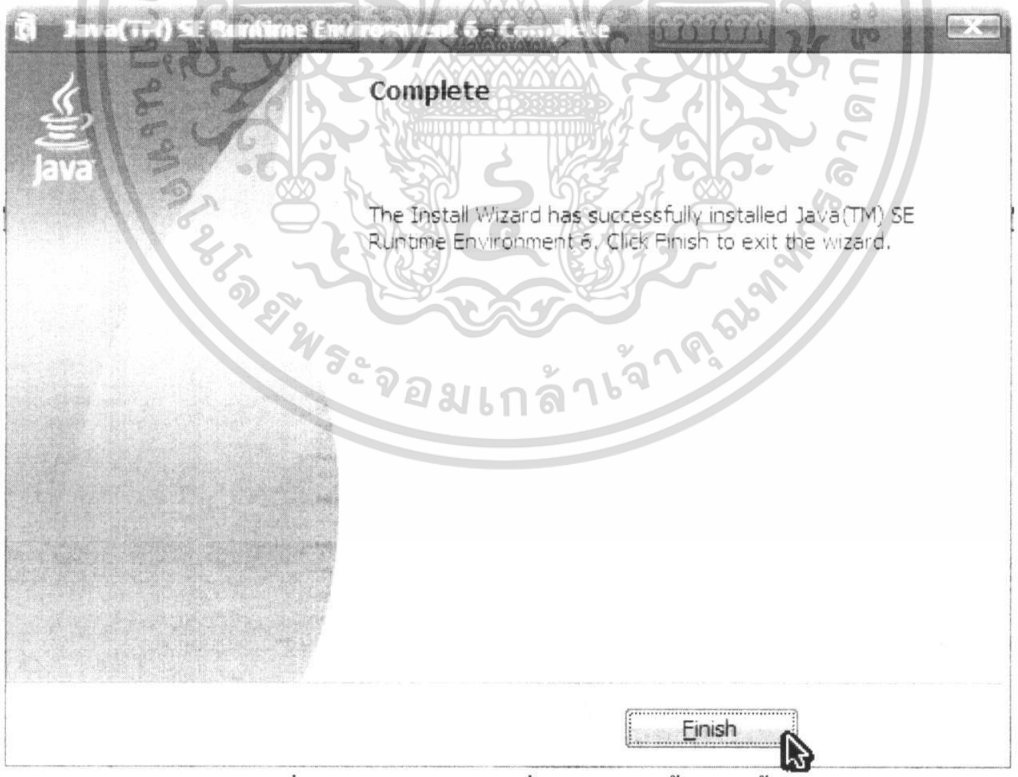


ภาพที่ ข.2 แสดงหน้าจอ License Agreement

- หลังจากนั้นรอจนกระทั่งโปรแกรมติดตั้งจนเสร็จ จนกระทั่งขึ้นหน้าจอ Complete ให้เรากด Finish ก็เป็นอันเสร็จสิ้นขั้นตอนการติดตั้ง Java(TM) SE Runtime Environment 6



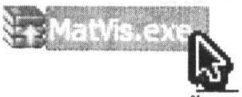
ภาพที่ ข.3 แสดงหน้าจอขณะกำลังติดตั้ง



ภาพที่ ข.4 แสดงหน้าจอเมื่อทำการติดตั้งเสร็จสิ้น

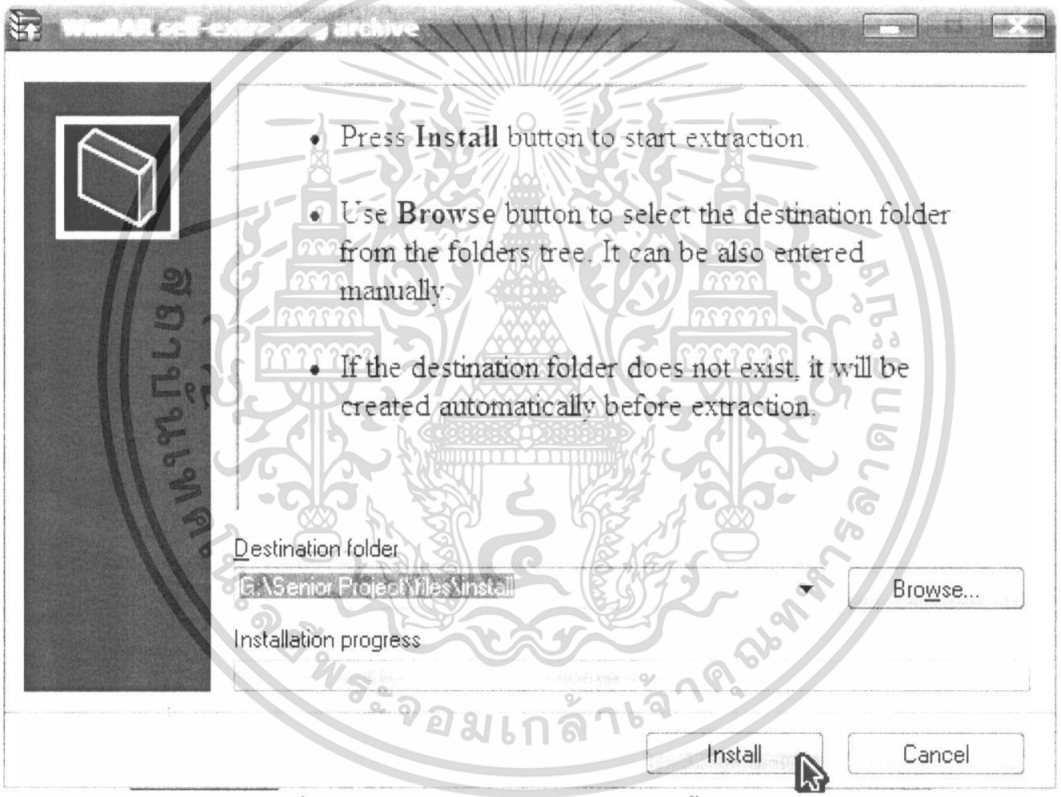
ข.2 วิธีการติดตั้งโปรแกรม MatVis

- ทำการดับเบิ้ลคลิกที่ไฟล์ชื่อ MatVis.exe ดังแสดงในภาพ ข.5



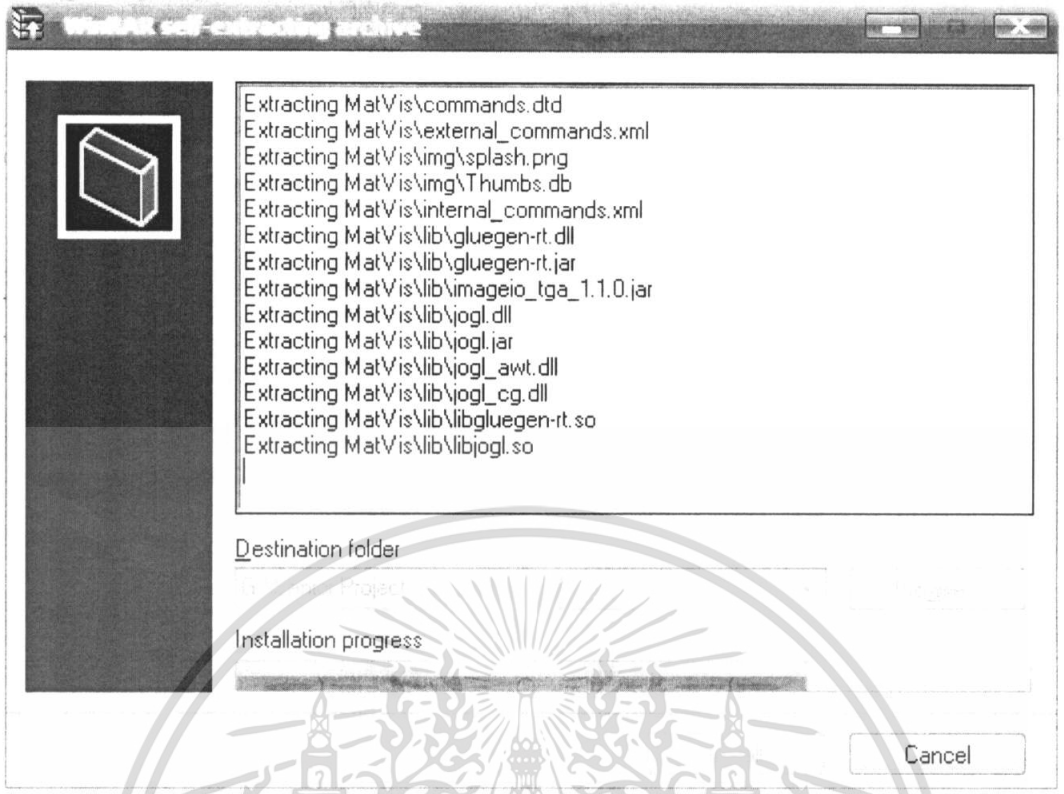
ภาพที่ ข.5 แสดงไฟล์สำหรับติดตั้งโปรแกรม MatVis

- หลังจากนั้นจะเข้าสู่หน้าสำหรับติดตั้งโปรแกรม ให้เราเลือกโฟลเดอร์ปลายทางที่ต้องการ จากนั้นกด Install ดังแสดงในภาพ ข.6



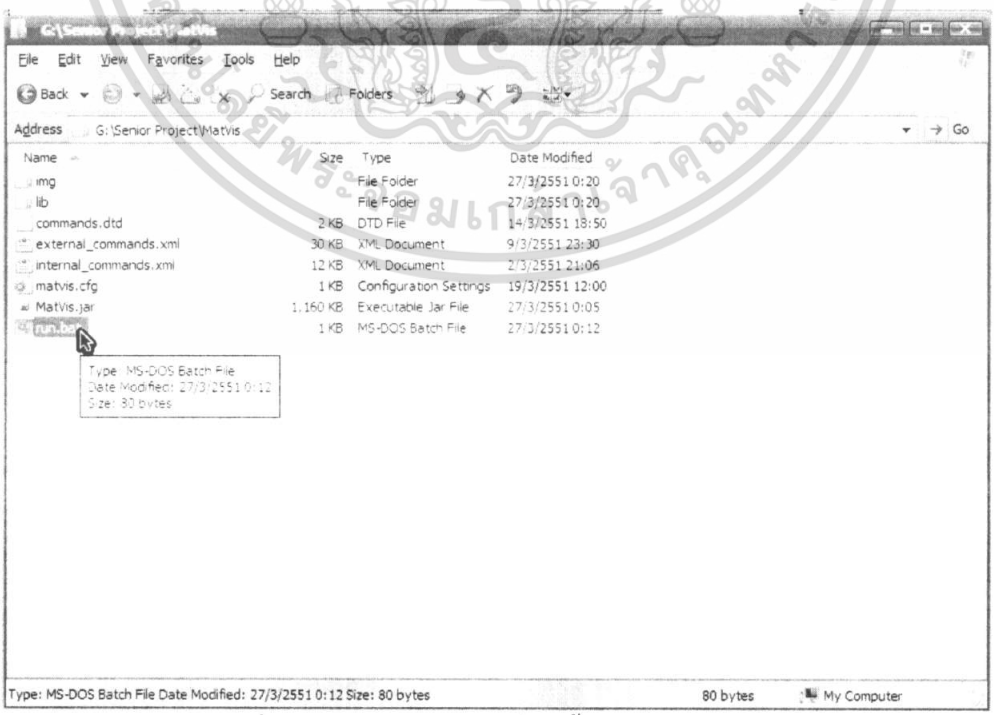
ภาพที่ ข.6 แสดงหน้าจอสำหรับการติดตั้งโปรแกรม

- หลังจากนั้นรอจนกระทั่งการติดตั้งเสร็จสมบูรณ์ หน้าจอการติดตั้งจะปิดตัวเองไปเองโดยอัตโนมัติ



ภาพที่ ข.7 แสดงหน้าจอขณะกำลังติดตั้ง

- เข้าไปยังโฟลเดอร์ที่เราเลือกติดตั้งโปรแกรมไว้ จากนั้นเลือกไฟล์ run.bat เพื่อทำการเริ่มใช้งานโปรแกรม



ภาพที่ ข.8 แสดงโปรแกรมที่ติดตั้งเสร็จเรียบร้อยแล้ว

- โปรแกรมจะเริ่มการทำงานโดยมีหน้าจอสแปลชแสดงชื่อโปรแกรมและชื่อผู้จัดทำ โดยหลังจากนั้นก็จะตัดเข้าสู่หน้าจอโปรแกรม ซึ่งก็เป็นอันเสร็จสมบูรณ์ขั้นตอนการติดตั้งโปรแกรม

ภาพที่ ข.9 แสดงหน้าจอสแปลชก่อนเข้าสู่โปรแกรม MatVis

ภาพที่ ข.10 แสดงหน้าจอโปรแกรมที่พร้อมใช้งาน

ภาคผนวก ค.

ตัวอย่างโปรแกรมที่ใช้ในการแก้ปัญหาไฟไนต์อิลิเมนท์ ในโปรแกรม MATLAB และ MatVis

ค.1 ปัญหาการวิเคราะห์การถ่ายเทความร้อนแบบอุณหภูมิคงที่ โดยใช้ MATLAB

```
%--------------------------------------------------------------------------
%Example
%to solve the two-dimensional Laplace equation given as
%u,xx+u,yy=0,0 < x < 5 ,0 < y < 10
%u(x,0) = 0,u(x,10) = 100sin(pi*x/10),
%u(0,y) = 0,u,x(5,y) = 0
%using linear triangle elements
%
%Variable descriptions
%k = element matrix
%f = element vector
%kk= system matrix
%ff= system vector
%gcoord = coordinate value of each node
%nodes = nodal connectivity of each element
%index = a vector containing system dofs associated with each element
%bcdof = a vector containing  dofs associated with boundary conditions
%bcval = a vector containing  boundary conditions  values associated with
%       the dofs in bcdof
%--------------------------------------------------------------------------
%
%---------------------------------
%input data for control parameter
%---------------------------------
nel=32;                 % number of elements
nnel=3;                 % number of node per element
ndof=1;                 % number of dofs per node
nnode=25;                 % total number of nodes in system
sdof=nnode*ndof;          % total system dofs
%
%-----------------------------------------------
```

```
% input data for nodal coordinate values
% gcoord(i,j)where i-> node no. and j-> x or y
%-----------------------------------------------
gcoord(1,1)=0.0;gcoord(1,2)=0.0;
gcoord(2,1)=1.25;gcoord(2,2)=0.0;
gcoord(3,1)=2.5;gcoord(3,2)=0.0;
gcoord(4,1)=3.75;gcoord(4,2)=0.0;
gcoord(5,1)=5.0;gcoord(5,2)=0.0;
gcoord(6,1)=0.0;gcoord(6,2)=2.5;
gcoord(7,1)=1.25;gcoord(7,2)=2.5;
gcoord(8,1)=2.5;gcoord(8,2)=2.5;
gcoord(9,1)=3.75;gcoord(9,2)=2.5;
gcoord(10,1)=5.0;gcoord(10,2)=2.5;
gcoord(11,1)=0.0;gcoord(11,2)=5.0;
gcoord(12,1)=1.25;gcoord(12,2)=5.0;
gcoord(13,1)=2.5;gcoord(13,2)=5.0;
gcoord(14,1)=3.75;gcoord(14,2)=5.0;
gcoord(15,1)=5.0;gcoord(15,2)=5.0;
gcoord(16,1)=0.0;gcoord(16,2)=7.5;
gcoord(17,1)=1.25;gcoord(17,2)=7.5;
gcoord(18,1)=2.5;gcoord(18,2)=7.5;
gcoord(19,1)=3.75;gcoord(19,2)=7.5;
gcoord(20,1)=5.0;gcoord(20,2)=7.5;
gcoord(21,1)=0.0;gcoord(21,2)=10;
gcoord(22,1)=1.25;gcoord(22,2)=10;
gcoord(23,1)=2.5;gcoord(23,2)=10;
gcoord(24,1)=3.75;gcoord(24,2)=10;
gcoord(25,1)=5.0;gcoord(25,2)=10;
%
%-----------------------------------------------------------
% input data for nodal connectivity for each element
% nodes(i,j)where i-> element no. and j-> connected nodes
```

```
%-------------------------------------------------------------
nodes(1,1)=1;nodes(1,2)=2;nodes(1,3)=7;
nodes(2,1)=2;nodes(2,2)=3;nodes(2,3)=8;
nodes(3,1)=3;nodes(3,2)=4;nodes(3,3)=9;
nodes(4,1)=4;nodes(4,2)=5;nodes(4,3)=10;
nodes(5,1)=1;nodes(5,2)=7;nodes(5,3)=6;
nodes(6,1)=2;nodes(6,2)=8;nodes(6,3)=7;
nodes(7,1)=3;nodes(7,2)=9;nodes(7,3)=8;
nodes(8,1)=4;nodes(8,2)=10;nodes(8,3)=9;
nodes(9,1)=6;nodes(9,2)=7;nodes(9,3)=12;
nodes(10,1)=7;nodes(10,2)=8;nodes(10,3)=13;
nodes(11,1)=8;nodes(11,2)=9;nodes(11,3)=14;
nodes(12,1)=9;nodes(12,2)=10;nodes(12,3)=15;
nodes(13,1)=6;nodes(13,2)=12;nodes(13,3)=11;
nodes(14,1)=7;nodes(14,2)=13;nodes(14,3)=12;
nodes(15,1)=8;nodes(15,2)=14;nodes(15,3)=13;
nodes(16,1)=9;nodes(16,2)=15;nodes(16,3)=14;
nodes(17,1)=11;nodes(17,2)=12;nodes(17,3)=17;
nodes(18,1)=12;nodes(18,2)=13;nodes(18,3)=18;
nodes(19,1)=13;nodes(19,2)=14;nodes(19,3)=19;
nodes(20,1)=14;nodes(20,2)=15;nodes(20,3)=20;
nodes(21,1)=11;nodes(21,2)=17;nodes(21,3)=16;
nodes(22,1)=12;nodes(22,2)=18;nodes(22,3)=17;
nodes(23,1)=13;nodes(23,2)=19;nodes(23,3)=18;
nodes(24,1)=14;nodes(24,2)=20;nodes(24,3)=19;
nodes(25,1)=16;nodes(25,2)=17;nodes(25,3)=22;
nodes(26,1)=17;nodes(26,2)=18;nodes(26,3)=23;
nodes(27,1)=18;nodes(27,2)=19;nodes(27,3)=24;
nodes(28,1)=19;nodes(28,2)=20;nodes(28,3)=25;
nodes(29,1)=16;nodes(29,2)=22;nodes(29,3)=21;
nodes(30,1)=17;nodes(30,2)=23;nodes(30,3)=22;
nodes(31,1)=18;nodes(31,2)=24;nodes(31,3)=23;
```

```
nodes(32,1)=19;nodes(32,2)=25;nodes(32,3)=24;
%
%-------------------------------------------------
% input data for boundary conditions
%-------------------------------------------------
bcdof(1)=1;            % first node is constrian
bcval(1)=0;            % whose describe value is 0
bcdof(2)=2;            % second node is constrian
bcval(2)=0;            % whose describe value is 0
bcdof(3)=3;            % thrid node is constrian
bcval(3)=0;            % whose describe value is 0
bcdof(4)=4;            % fourth node is constrian
bcval(4)=0;            % whose describe value is 0
bcdof(5)=5;            % fifth node is constrian
bcval(5)=0;            % whose describe value is 0
bcdof(6)=6;            % sixth node is constrian
bcval(6)=0;            % whose describe value is 0
bcdof(7)=11;           % 11th node is constrian
bcval(7)=0;            % whose describe value is 0
bcdof(8)=16;           % 16th node is constrian
bcval(8)=0;            % whose describe value is 0
bcdof(9)=21;           % 21th node is constrian
bcval(9)=0;            % whose describe value is 0
bcdof(10)=22;          % 22th node is constrian
bcval(10)=38.2683;     % whose describe value is 38.2683
bcdof(11)=23;          % 23th node is constrian
bcval(11)=70.7107;     % whose describe value is 70.7107
bcdof(12)=24;          % 24th node is constrian
bcval(12)=92.3880;     % whose describe value is 92.3880
bcdof(13)=25;          % 25th node is constrian
bcval(13)=100;         % whose describe value is 100
%
```

```
%-------------------------------------------------------------------
% initialization of matrices and vectors
%-------------------------------------------------------------------
ff=zeros(sdof,1);          % initialization of system force vector
kk=zeros(sdof,sdof);       % initialization of index vector
index=zeros(nnel*ndof,1);  % initialization of index vector
%
%-------------------------------------------------------------------
% computation of element matrices and vectors and their assembly
%-------------------------------------------------------------------
for iel=1:nel          % loop for the total number of elements
%
nd(1)=nodes(iel,1);    % 1st connected node for (iel)-th element
nd(2)=nodes(iel,2);    % 2nd connected node for (iel)-nd element
nd(3)=nodes(iel,3);    % 3rd connected node for (iel)-rd element
x1=gcoord(nd(1),1);y1=gcoord(nd(1),2);       % coord values of 1st node
x2=gcoord(nd(2),1);y2=gcoord(nd(2),2);       % coord values of 2nd node
x3=gcoord(nd(3),1);y3=gcoord(nd(3),2);       % coord values of 3rd node
%
index=feeldof(nd,nnel,ndof);           % extract system dofs for the element
%
k=felp2dt3(x1,y1,x2,y2,x3,y3);          % compute element matrix
%
kk=feasmbl1(kk,k,index);               % assemble element matrices
%
end
%
%---------------------------------------
% apply boundary conditions
%---------------------------------------
[kk,ff]=feaplyc2(kk,ff,bcdof,bcval);
%
```

```
%----------------------------------------
% solve the matrix equation
%----------------------------------------
fsol=kk\ff;
%
%-----------------------------
% analytical solution
%-----------------------------
for i=1:nnode
   x=gcoord(i,1);y=gcoord(i,2);
   esol(i)=100*sinh(0.31415927*y)*sin(0.31415927*x)/sinh(3.1415927);
end
%
%-----------------------------------
% print both exact and fem solutions
%-----------------------------------
num=1:1:sdof;
store=[num' fsol esol']
%
%-------------------------------------------------------------------
for i=1:5
 for j=1:5
  zz(i,j) = fsol( (5*(i-1)) + j)
 end
 xx(i) = gcoord(i,1)
 yy(i) = gcoord(5*i, 2)
end
mesh(xx,yy,zz)
```

Laplace's and Poisson Equations

```
function [kk]=feasmbl1(kk,k,index)
%-------------------------------------------------------------------------
%Purpose:
%Assenbly of element matrices into the system matrix
%
%Synopsis:
%[kk]=feasmbl1(kk,k,index)
%
%Variable Description:
%kk - system matrix
%k - element matrix
%index - dof vector associated with an element
%-------------------------------------------------------------------------
%
edof = length(index);
for i=1:edof
   ii=index(i);
   for j=1:edof
      jj=index(j);
      kk(ii,jj)=kk(ii,jj)+k(i,j);
   end
end
%-------------------------------------------------------------------------


function [index]=feeldof(nd,nnel,ndof)
%-------------------------------------------------------------------------
%Purpose:
%Compute system dofs associated with each element
%
%Synopsis:
```

```
%[index]=feeldof(nd,nnel,ndof)
%
%Variable Description:
%index - system dof vector associated with element iel
%nd - element node numbers whose system dofs are to be determine
%nnel - number of nodes per element
%ndof - number of dofs per node
%-------------------------------------------------------------------------
%
edof=nnel*ndof;
k=0;
for i=1:nnel
   start=(nd(i)-1)*ndof;
   for j=1:ndof
      k=k+1;
      index(k)=start+j;
   end
end
%-------------------------------------------------------------------------



function [k]=felp2dt3(x1,y1,x2,y2,x3,y3)
%-------------------------------------------------------------------------
%Purpose:
%element matrix for two-dimensional Laplace's equation
%using three-node linear triangular element
%
%Synopsis:
%[k]=felp2dt3(x1,y1,x2,y2,x3,y3)
%
%Variable Description:
%k - element stiffness matrix (size of 3*3)
```

```
%x1,y1 - x and y coordinate values of the first node of element
%x2,y2 - x and y coordinate values of the second node of element
%x3,y3 - x and y coordinate values of the third node of element
%-------------------------------------------------------------------------
%
%element matrix
%
A=0.5*(x2*y3+x1*y2+x3*y1-x2*y1-x1*y3-x3*y2);
%                                   %area of the triangle
k(1,1)=((x3-x2)*(x3-x2)+(y2-y3)*(y2-y3))/(4*A);
k(1,2)=((x3-x2)*(x1-x3)+(y2-y3)*(y3-y1))/(4*A);
k(1,3)=((x3-x2)*(x2-x1)+(y2-y3)*(y1-y2))/(4*A);
k(2,1)=k(1,2);
k(2,2)=((x1-x3)*(x1-x3)+(y3-y1)*(y3-y1))/(4*A);
k(2,3)=((x1-x3)*(x2-x1)+(y3-y1)*(y1-y2))/(4*A);
k(3,1)=k(1,3);
k(3,2)=k(2,3);
k(3,3)=((x2-x1)*(x2-x1)+(y1-y2)*(y1-y2))/(4*A);
%-------------------------------------------------------------------------
```

ค.2 ปัญหาการวิเคราะห์การถ่ายเทความร้อนแบบอุณหภูมิคงที่ โดยใช้ MatVis

laplace.mvs

```
// ------------------------------------------------------------------------
// Example:
// to solve the two-dimensional Laplace equation given  as
// u,  xx + u,  yy  =  0, 0 < x < 5 , 0 < y < 10
// u[x, 0]  =  0, u[x, 10]  =  100sin[pi*x/10],
// u[0, y]  =  0, u, x[5, y]  =  0
// using linear triangle elements
// [see Fig.5.9.1 for the finite element mesh]
//
// variable descriptions
// k  =  element matrix
// f  =  element vector[array]
// kk =  system matrix
// ff =  system vector[array]
// gcoord  =  coordinate value of each node
// nodes  =  nodal connectivity of each element
// index  =  a vector containing system dofs associated with each element
// bcdof  =  a vector containing  dofs associated with boundary conditions
// bcval  =  a vector containing  boundary conditions  values associated with
//       the dofs in bcdof
// ------------------------------------------------------------------------


//
// ---------------------------------
//  run finite elements functions
// ---------------------------------
run("D:/WorkEx/MatVisProb/prob1/laplace_functions.mvs")$
```

```
//
// ---------------------------------
// input data for control parameter
// ---------------------------------
nx = 5$
ny = 5$
nel = 32$                  // number of elements
nnel = 3$                  // number of node per element
ndof = 1$                  // number of dofs per node
nnode = 25$                // total number of nodes in system
sdof = nnode * ndof$       // total system dofs

//
// -------------------------------------------
//  input data for nodal coordinate values
//  gcoord[i, j]where i-> node no. and j-> x or y
// -------------------------------------------
//
// -------------------------------------------
//  input data for nodal coordinate values
//  gcoord[i, j]where i-> node no. and j-> x or y
// -------------------------------------------
gcoord[0, 0] = 0.0$ gcoord[0, 1] = 0.0$
gcoord[1, 0] = 1.25$ gcoord[1, 1] = 0.0$
gcoord[2, 0] = 2.5$ gcoord[2, 1] = 0.0$
gcoord[3, 0] = 3.75$ gcoord[3, 1] = 0.0$
gcoord[4, 0] = 5.0$ gcoord[4, 1] = 0.0$
gcoord[5, 0] = 0.0$ gcoord[5, 1] = 2.5$
gcoord[6, 0] = 1.25$ gcoord[6, 1] = 2.5$
gcoord[7, 0] = 2.5$ gcoord[7, 1] = 2.5$
gcoord[8, 0] = 3.75$ gcoord[8, 1] = 2.5$
gcoord[9, 0] = 5.0$ gcoord[9, 1] = 2.5$
```

```
gcoord[10, 0] = 0.0$ gcoord[10, 1] = 5.0$
gcoord[11, 0] = 1.25$ gcoord[11, 1] = 5.0$
gcoord[12, 0] = 2.5$ gcoord[12, 1] = 5.0$
gcoord[13, 0] = 3.75$ gcoord[13, 1] = 5.0$
gcoord[14, 0] = 5.0$ gcoord[14, 1] = 5.0$
gcoord[15, 0] = 0.0$ gcoord[15, 1] = 7.5$
gcoord[16, 0] = 1.25$ gcoord[16, 1] = 7.5$
gcoord[17, 0] = 2.5$ gcoord[17, 1] = 7.5$
gcoord[18, 0] = 3.75$ gcoord[18, 1] = 7.5$
gcoord[19, 0] = 5.0$ gcoord[19, 1] = 7.5$
gcoord[20, 0] = 0.0$ gcoord[20, 1] = 10.0$
gcoord[21, 0] = 1.25$ gcoord[21, 1] = 10.0$
gcoord[22, 0] = 2.5$ gcoord[22, 1] = 10.0$
gcoord[23, 0] = 3.75$ gcoord[23, 1] = 10.0$
gcoord[24, 0] = 5.0$ gcoord[24, 1] = 10.0$
//
// ---------------------------------------------------
// input data for nodal connectivity for each element
// nodes[i, j]where i-> element no. and j-> connected nodes
// ---------------------------------------------------
nodes[0, 0] = 0$ nodes[0, 1] = 1$ nodes[0, 2] = 6$
nodes[1, 0] = 1$ nodes[1, 1] = 2$ nodes[1, 2] = 7$
nodes[2, 0] = 2$ nodes[2, 1] = 3$ nodes[2, 2] = 8$
nodes[3, 0] = 3$ nodes[3, 1] = 4$ nodes[3, 2] = 9$
nodes[4, 0] = 0$ nodes[4, 1] = 6$ nodes[4, 2] = 5$
nodes[5, 0] = 1$ nodes[5, 1] = 7$ nodes[5, 2] = 6$
nodes[6, 0] = 2$ nodes[6, 1] = 8$ nodes[6, 2] = 7$
nodes[7, 0] = 3$ nodes[7, 1] = 9$ nodes[7, 2] = 8$
nodes[8, 0] = 5$ nodes[8, 1] = 6$ nodes[8, 2] = 11$
nodes[9, 0] = 6$ nodes[9, 1] = 7$ nodes[9, 2] = 12$
nodes[10, 0] = 7$ nodes[10, 1] = 8$ nodes[10, 2] = 13$
nodes[11, 0] = 8$ nodes[11, 1] = 9$ nodes[11, 2] = 14$
```

```
nodes[12, 0] = 5$ nodes[12, 1] = 11$ nodes[12, 2] = 10$
nodes[13, 0] = 6$ nodes[13, 1] = 12$ nodes[13, 2] = 11$
nodes[14, 0] = 7$ nodes[14, 1] = 13$ nodes[14, 2] = 12$
nodes[15, 0] = 8$ nodes[15, 1] = 14$ nodes[15, 2] = 13$
nodes[16, 0] = 10$ nodes[16, 1] = 11$ nodes[16, 2] = 16$
nodes[17, 0] = 11$ nodes[17, 1] = 12$ nodes[17, 2] = 17$
nodes[18, 0] = 12$ nodes[18, 1] = 13$ nodes[18, 2] = 18$
nodes[19, 0] = 13$ nodes[19, 1] = 14$ nodes[19, 2] = 19$
nodes[20, 0] = 10$ nodes[20, 1] = 16$ nodes[20, 2] = 15$
nodes[21, 0] = 11$ nodes[21, 1] = 17$ nodes[21, 2] = 16$
nodes[22, 0] = 12$ nodes[22, 1] = 18$ nodes[22, 2] = 17$
nodes[23, 0] = 13$ nodes[23, 1] = 19$ nodes[23, 2] = 18$
nodes[24, 0] = 15$ nodes[24, 1] = 16$ nodes[24, 2] = 21$
nodes[25, 0] = 16$ nodes[25, 1] = 17$ nodes[25, 2] = 22$
nodes[26, 0] = 17$ nodes[26, 1] = 18$ nodes[26, 2] = 23$
nodes[27, 0] = 18$ nodes[27, 1] = 19$ nodes[27, 2] = 24$
nodes[28, 0] = 15$ nodes[28, 1] = 21$ nodes[28, 2] = 20$
nodes[29, 0] = 16$ nodes[29, 1] = 22$ nodes[29, 2] = 21$
nodes[30, 0] = 17$ nodes[30, 1] = 23$ nodes[30, 2] = 22$
nodes[31, 0] = 18$ nodes[31, 1] = 24$ nodes[31, 2] = 23$
//
// ------------------------------------------------
// input data for boundary conditions
// ------------------------------------------------
bcdof[0] = 0$          // first node is constrian
bcval[0] = 0$          // whose describe value is 0
bcdof[1] = 1$          // second node is constrian
bcval[1] = 0$          // whose describe value is 0
bcdof[2] = 2$          // thrid node is constrian
bcval[2] = 0$          // whose describe value is 0
bcdof[3] = 3$          // fourth node is constrian
bcval[3] = 0$          // whose describe value is 0
```

```
bcdof[4] = 4$              // fifth node is constrian
bcval[4] = 0$              // whose describe value is 0
bcdof[5] = 5$              // sixth node is constrian
bcval[5] = 0$              // whose describe value is 0
bcdof[6] = 10$             // 11th node is constrian
bcval[6] = 0$              // whose describe value is 0
bcdof[7] = 15$             // 16th node is constrian
bcval[7] = 0$              // whose describe value is 0
bcdof[8] = 20$             // 21th node is constrian
bcval[8] = 0$              // whose describe value is 0
bcdof[9] = 21$             // 22th node is constrian
bcval[9] = 38.2683$        // whose describe value is 38.2683
bcdof[10] = 22$            // 23th node is constrian
bcval[10] = 70.7107$       // whose describe value is 70.7107
bcdof[11] = 23$            // 24th node is constrian
bcval[11] = 92.3880$       // whose describe value is 92.3880
bcdof[12] = 24$            // 25th node is constrian
bcval[12] = 100$           // whose describe value is 100
//
// ---------------------------------------------------------------
// initialization of matrices and vectors
// ---------------------------------------------------------------
sdofz = sdof - 1$
ff[0] = 0$          // initialization of system force vector
kk[sdofz, sdofz] = 0$       // initialization of index vector
index[0] = 0$   // initialization of index vector
//
// ---------------------------------------------------------------
// computation of element matrices and vectors and their assembly
// ---------------------------------------------------------------
// loop for the total number of elements
for iel = 0 to nel - 1 step 1.0 do
```

```
nd[0] = nodes[iel, 0];     // 1st connected node for [iel]-th element
nd[1] = nodes[iel, 1];     // 2nd connected node for [iel]-nd element
nd[2] = nodes[iel, 2];     // 3rd connected node for [iel]-rd element
// coord values of 1st node
x1 = gcoord[nd[0], 0];
y1 = gcoord[nd[0], 1];
// coord values of 2nd node
x2 = gcoord[nd[1], 0];
y2 = gcoord[nd[1], 1];
// coord values of 3rd node
x3 = gcoord[nd[2], 0];
y3 = gcoord[nd[2], 1];
// extract system dofs for the element
[index] = feeldof(nd, nnel, ndof);
// compute element matrix
[k] = felp2dt3(x1, y1, x2, y2, x3, y3);

// assemble element matrices
[kk] = feasmbll(kk, k, index);
loop$
//
// ---------------------------------------
// apply boundary conditions
// ---------------------------------------
[kk, ff] = feaplyc2(kk, ff, bcdof, bcval)$
saveMatrix(kk, "c:/kk.txt")$
mff = arrayToMat(ff)$
saveMatrix(mff, "c:/ff.txt")$
//
// ---------------------------------------
// solve the matrix equation
// ---------------------------------------
```

```
fsol = ludecompos(kk, mff)$
fsol2 = gauss_elim(kk, mff)$
laplace_functions.mvs
//
// ------------------------------
//  analytical solution
// ------------------------------
for i = 0 to nnode - 1 do
   x = gcoord[i, 0]; y = gcoord[i, 1];
   esol[i, 0] = 100 * sinh(0.31415927 * y) * sin(0.31415927 * x) / sinh(3.1415927);
loop$
//
// ------------------------------------
//  print both exact and fem solutions
// ------------------------------------
//mfsol = arrayToMat(fsol)$
saveMatrix(fsol, "c:/fsol.txt")$
saveMatrix(esol, "c:/esol.txt")$

saveMatrix(mfsol, "c:/fsol.txt")$
saveMatrix(esol, "c:/esol.txt")$

j=0$
for i = 0 to nx - 1 do
 for j = 0 to ny - 1 do
  pos = nx * i + j;
  mat[pos, 0] = gcoord[pos, 0];
  mat[pos, 1] = fsol[pos, 0];
  mat[pos, 2] = gcoord[pos, 1];
 loop;
loop$
saveMatrix(mat, "c:/gfx.txt")$
```

```
plot3d(mat, nx, ny)$

//
// ---------------------------------------------------------------------
function [rkk] = feasmbll(kk, kv, idx)
// ---------------------------------------------------------------------
// Purpose:
// Assenbly of element matrices into the system matrix
//
// Synopsis:
// rkk = feasmbl1[kk, k, index]
//
// Variable Description:
// kk - system matrix
// k - element matrix
// index - dof vector associated with an element
// ---------------------------------------------------------------------
//
edof = arrayLen(idx) - 1;
ii = 0;
jj = 0;
rkk = kk;
for i = 0 to edof do
   ii = idx[i];
   for j = 0 to edof do
      jj = idx[j];
      rkk[ii, jj] = rkk[ii, jj] + kv[i, j];
   loop;
loop;
end function$
// ---------------------------------------------------------------------
```

```
function [idx] = feeldof(nd, nnel, ndof)
// ------------------------------------------------------------------------
// Purpose:
// Compute system dofs associated with each element
//
// Synopsis:
// [index]=feeldof(nd,nnel,ndof)
//
// Variable Description:
// index - system dof vector associated with element iel
// nd - element node numbers whose system dofs are to be determine
// nnel - number of nodes per element
// ndof - number of dofs per node
// ------------------------------------------------------------------------
//
edof = nnel * ndof;
k = 0;
idx[0] = 0;
for i=0 to 2 do
   start = ( nd[i] - 1 ) * ndof;
   for j=1 to ndof do
      idx[k] = start + j;
      k = k + 1;
   loop;
loop;
end function$
// ------------------------------------------------------------------------

function [rk] = felp2dt3(x1, y1, x2, y2, x3, y3)
// ------------------------------------------------------------------------
// Purpose:
// element matrix for two-dimensional Laplace's equation
```

```
// using three-node linear triangular element
//
// Synopsis:
// [k] = felp2dt3(x1, y1, x2, y2, x3, y3)
//
// Variable Description:
// rk - element stiffness matrix [size of 3*3)
// x1,y1 - x and y coordinate values of the first node of element
// x2,y2 - x and y coordinate values of the second node of element
// x3,y3 - x and y coordinate values of the third node of element
// ---------------------------------------------------------------------

// element matrix
A = 0.5 * ((x2* y3) + (x1 * y2) + (x3 * y1) - (x2 * y1) - (x1 * y3) -(x3 * y2));
rk[2, 2] = 0;
// area of the triangle
rk[0, 0] = ((x3 - x2) * (x3 - x2) + (y2 - y3) * (y2 - y3)) / (4 * A);
rk[0, 1] = ((x3 - x2) * (x1 - x3) + (y2 - y3) * (y3 - y1)) / (4 * A);
rk[0, 2] = ((x3 - x2) * (x2 - x1) + (y2 - y3) * (y1 - y2)) / (4 * A);
rk[1, 0] = rk[0, 1];
rk[1, 1] = ((x1 - x3) * (x1 - x3) + (y3 - y1) * (y3 - y1)) / (4 * A);
rk[1, 2] = ((x1 - x3) * (x2 - x1) + (y3 - y1) * (y1 - y2)) / (4 * A);
rk[2, 0] = rk[0, 2];
rk[2, 1] = rk[1, 2];
rk[2, 2] = ((x2 - x1) * (x2 - x1) + (y1 - y2) * (y1 - y2)) / (4 * A);
end function$

function [rkk, rff] = feaplyc2(kk, ff, bcdof, bcval)
// ---------------------------------------------------------------------
// Purpose:
// Apply constraints to matrix equation [kk]x=ff
//
```

```
// Synopsis:
// [rkk,rff]=feaplyc2(kk,ff,bcdof,bcval)
//
// Variable Description:
// kk - system matrix before applying constraints
// ff - system vector before applying constraints
// bcdof - a vector containing constrained dof
// bcval - a vector containing constrained value
//
// For example, there are constraints at dof=2 and 10
// and their constrained values are 0.0 and 2.5,
// respectively. Then, bcdof(1)=2 and bcdof(2)=10; and
// bcval(1)=1.0 and bcval(2)=2.5.
// -----------------------------------------------------------------------
//
n = arrayLen(bcdof) - 1;
sdof = getRows(kk) - 1;
rkk = kk;
rff = ff;
for i=0 to n do
   c = bcdof[i];
   for j=0 to sdof do
      rkk[c, j] = 0;
   loop;

   rkk[c, c] = 1;
   rff[c] = bcval[i];
loop;
end function$
```

ภาคผนวก ง.

กฎการตัดคำและไวยากรณ์ของภาษา

ง.1 โครงสร้างของกฎในการตัดทำ

```
import java_cup.runtime.Symbol;

%%

%public
%cup
%implements sym

%unicode

%{
  StringBuffer string = new StringBuffer();
  SymbolTable symbolTable;        // External symbol table

  public void setSymbolTable(SymbolTable symbolTable) {
    this.symbolTable = symbolTable;
  }

  private Symbol symbol(int symbolID) {
    return new Symbol(symbolID);
  }

  private Symbol symbol(int symbolID, Object value) {
    return new Symbol(symbolID, value);
  }
%}

/* Identifiers */
Identifier = [:jletter:][:jletterdigit:]*
```

```
/* Integer literals */
IntegerLiteral = 0 | [1-9][0-9]*

/* Real number literals */
Sign = [-]
RealLiteral = {Sign}? ({FloatLiteral1}|{FloatLiteral2}) {Exponent}?
FloatLiteral1 = {IntegerLiteral} \. [0-9]*
FloatLiteral2 = [0-9]+
Exponent = [eE] [+-]? [0-9]+

/* Complex number literals */
ComplexLiteral = {RealLiteral} i

/* String literals */
StringCharacter = [^\r\n\"\\]

/* Line terminator character */
LineTerminator = \r|\n|\r\n

%yylexthrow Exception
%state STRING_STATE

%%
<YYINITIAL> {
  /* Keywords */
  "while"                       { return symbol(WHILE); }
  "do"                  { return symbol(DO); }
  "loop"                { return symbol(LOOP); }
  "end"                 { return symbol(END); }
  "if"                  { return symbol(IF); }
  "then"                { return symbol(THEN); }
  "else"                { return symbol(ELSE); }
```

```
"function"          { return symbol(FUNCTION); }
"for"               { return symbol(FOR); }
"to"                { return symbol(TO); }
"step"              { return symbol(STEP); }

{Identifier}        { symbolTable.enter(yytext(), new SymbolTableEntry(yytext()));
                      return symbol(ID, yytext()); }
{RealLiteral}       { return symbol(REAL, yytext()); }
{ComplexLiteral}    { return symbol(COMPLEX, yytext()); }

/* Separators */
","                 { return symbol(COMMA); }
"("                 { return symbol(LPAR); }
")"                 { return symbol(RPAR); }
"{"                 { return symbol(LBRC); }
"}"                 { return symbol(RBRC); }
"["                 { return symbol(LBRK); }
"]"                 { return symbol(RBRK); }

/* Operators */
"="                 { return symbol(ASSIGN); }
"=="                { return symbol(EQ); }
"!="                { return symbol(NEQ); }
"<"                 { return symbol(LE); }
"<="                { return symbol(LEQ); }
">"                 { return symbol(GE); }
">="                { return symbol(GEQ); }
"-"                 { return symbol(MINUS); }
"+"                 { return symbol(PLUS); }
"*"                 { return symbol(TIMES); }
"/"                 { return symbol(DIV); }
"%"                 { return symbol(MOD); }
```

```
"&&"                    { return symbol(AND); }
"^"                     { return symbol(POW); }
"||"                    { return symbol(OR); }
";"                     { return symbol(SEMICOL); }

[\ \t\b\f\r\n]+         { /* skip whitespace */ }
"//"                    { /* end-of-line comment */ }

/* string literal */
\"                  { yybegin(STRING_STATE); string.setLength(0); }
}

<STRING_STATE> {
\"                  { yybegin(YYINITIAL); return symbol(STRING, string.toString()); }

{StringCharacter}+          { string.append( yytext() ); }

/* escape sequences */
"\\b"               { string.append( '\b' ); }
"\\t"               { string.append( '\t' ); }
"\\n"               { string.append( '\n' ); }
"\\f"               { string.append( '\f' ); }
"\\r"               { string.append( '\r' ); }
"\\\""              { string.append( '\"' ); }
"\\'"               { string.append( '\'' ); }
"\\\\"              { string.append( '\\' ); }

/* error cases */
\\.                 { throw new RuntimeException("Illegal escape sequence \""+yytext()+"\""); }
{LineTerminator}        { throw new RuntimeException("Unterminated string at end of line"); }
}
```

```
/* error fallback */
.                       { throw new Exception(); }
<<EOF>>                         { return symbol(EOF); }
```

ง.2 หลักไวยากรณ์โครงสร้างของภาษา

```
// definition of tokens, if applicable with token type
terminal WHILE, DO, LOOP, END, IF, THEN, ELSE, FUNCTION, TO, FOR, STEP;
terminal SEMICOL, COMMA;
terminal LPAR, RPAR, LBRC, RBRC, LBRK, RBRK;
terminal ASSIGN, EQ, NEQ, LE, LEQ, GE, GEQ;
terminal MINUS, PLUS, TIMES, DIV, MOD, POW;
terminal AND, OR;
terminal UMINUS;

terminal String ID, REAL, COMPLEX;
terminal String STRING;

non terminal TStatement statement;
non terminal TStatementSequence statement_sequence;
non terminal TWhileStatement while_statement;
non terminal TForStatement for_statement;
non terminal TIfStatement if_statement;
non terminal TFunction function_declaration;
non terminal TExpressionList expression_list;
non terminal TNumberList number_list;
non terminal TIdentifierList identifier_list;
non terminal TExpression expression_statement;
non terminal TBooleanExpression boolean_expression;
non terminal TCollectionInitializer collection_initializer;
non terminal TArrayInitializer array_initializer;
non terminal TArrayInitializerList array_initializer_list;
non terminal TMatrixInitializer matrix_initializer;
non terminal TMatrixInitializerList matrix_initializer_list;
non terminal TCubeInitializer cube_initializer;
non terminal TAFunctionCall function_call;
```

```
non terminal TIdentifier identifier;
non terminal TNumber number;
non terminal TString string;

// precedences, left associativity
precedence left ASSIGN;
precedence left AND, OR;
precedence left EQ, NEQ;
precedence left LE, LEQ, GE, GEQ;
precedence left MINUS, PLUS;
precedence left TIMES, DIV, MOD;
precedence left UMINUS;
precedence right POW;

// grammar rules
statement ::= while_statement:whileStmt
              {: RESULT = new TStatement(whileStmt); :}
              | for_statement:forStmt
              {: RESULT = new TStatement(forStmt); :}
              | if_statement:ifStmt
              {: RESULT = new TStatement(ifStmt); :}
              | expression_statement:expStmt
              {: RESULT = new TStatement(expStmt); :}
              | function_declaration:funcDecl
              {: RESULT = new TStatement(funcDecl); :}
              ;

statement_sequence ::= statement_sequence:stmtSeq statement:stmt SEMICOL
              {: RESULT = new TStatementSequence(stmtSeq, stmt); :}
              | statement:stmt SEMICOL
              {: RESULT = new TStatementSequence(stmt); :}
              ;
```

```
while_statement ::= WHILE boolean_expression:booleanExp DO statement_sequence:stmtSeq LOOP
                    {: RESULT = new TWhileStatement(booleanExp, stmtSeq); :}
                    ;

for_statement ::= FOR identifier:id ASSIGN expression_statement:expStmt TO
expression_statement:expStmt2 DO statement_sequence:stmtSeq LOOP
                    {: RESULT = new TForStatement(id, expStmt, expStmt2, stmtSeq); :}
                    | FOR identifier:id ASSIGN expression_statement:expStmt TO
expression_statement:expStmt2 STEP expression_statement:expStmt3 DO
statement_sequence:stmtSeq LOOP
                    {: RESULT = new TForStatement(id, expStmt, expStmt2, expStmt3, stmtSeq); :}
                    ;

if_statement ::= IF boolean_expression:booleanExp THEN statement_sequence:stmt END IF
                    {: RESULT = new TIfStatement(booleanExp, stmt); :}
                    | IF boolean_expression:booleanExp THEN statement_sequence:stmt1 ELSE
statement_sequence:stmt2 END IF
                    {: RESULT = new TIfStatement(booleanExp, stmt1, stmt2); :}
                    ;

function_declaration ::= FUNCTION identifier:funcName LPAR RPAR ASSIGN
expression_statement:expStmt END FUNCTION
                    {: RESULT = new TFunction(funcName, null, expStmt); :}

                    | FUNCTION identifier:funcName LPAR identifier_list:idList RPAR ASSIGN
expression_statement:expStmt END FUNCTION
                    {: RESULT = new TFunction(funcName, idList, expStmt); :}

                    | FUNCTION identifier:returnName ASSIGN identifier:funcName LPAR RPAR
statement_sequence:stmtSeq END FUNCTION
                    {: RESULT = new TFunction(funcName, returnName, null, stmtSeq); :}
```

```
                | FUNCTION identifier:returnName ASSIGN identifier:funcName LPAR
identifier_list:idList RPAR statement_sequence:stmtSeq END FUNCTION
                {: RESULT = new TFunction(funcName, returnName, idList, stmtSeq); :}

                | FUNCTION LBRK identifier_list:idList RBRK ASSIGN identifier:funcName
LPAR RPAR statement_sequence:stmtSeq END FUNCTION
                {: RESULT = new TFunction(idList, funcName, null, stmtSeq); :}

                | FUNCTION LBRK identifier_list:idList RBRK ASSIGN identifier:funcName
LPAR identifier_list:idList2 RPAR statement_sequence:stmtSeq END FUNCTION
                {: RESULT = new TFunction(idList, funcName, idList2, stmtSeq); :}
                ;

expression_list ::= expression_statement:expStmt
                {: RESULT = new TExpressionList(expStmt); :}
            | expression_list:expList COMMA expression_statement:expStmt
                {: RESULT = new TExpressionList(expList, expStmt); :}
            ;

number_list ::= number:num
                {: RESULT = new TNumberList(num); :}
                | number_list:numList COMMA number:num
                {: RESULT = new TNumberList(numList, num); :}
                ;

identifier_list ::= identifier:id
                {: RESULT = new TIdentifierList(id); :}
                | identifier_list:idList COMMA identifier:id
                {: RESULT = new TIdentifierList(idList, id); :}
                ;
```

```
function_call ::= identifier:id LPAR RPAR
                  {: RESULT = new TFunctionCall(id); :}
                  | identifier:id LPAR expression_list:expList RPAR
                  {: RESULT = new TFunctionCall(id, expList); :}
                  ;

expression_statement ::= number:num
                  {: RESULT = num; :}
                  | identifier:id
                  {: RESULT = id; :}
                  | string:str
                  {: RESULT = str; :}
                  | function_call:functionCall
                  {: RESULT = functionCall; :}
                  | LPAR expression_statement:expStmt RPAR
                  {: RESULT = expStmt; :}
                  | MINUS expression_statement:expStmt
                  {: RESULT = new TExpressionPrefix(expStmt, "-"); :} %prec UMINUS
                  | expression_statement:leftExp PLUS expression_statement:rightExp
                  {: RESULT = new TExpressionInfix(leftExp, "+", rightExp); :}
                  | expression_statement:leftExp MINUS expression_statement:rightExp
                  {: RESULT = new TExpressionInfix(leftExp, "-", rightExp); :}
                  | expression_statement:leftExp TIMES expression_statement:rightExp
                  {: RESULT = new TExpressionInfix(leftExp, "*", rightExp); :}
                  | expression_statement:leftExp DIV expression_statement:rightExp
                  {: RESULT = new TExpressionInfix(leftExp, "/", rightExp); :}
                  | expression_statement:leftExp MOD expression_statement:rightExp
                  {: RESULT = new TExpressionInfix(leftExp, "%", rightExp); :}
                  | expression_statement:leftExp POW expression_statement:rightExp
                  {: RESULT = new TExpressionInfix(leftExp, "^", rightExp); :}
                  | identifier:id ASSIGN expression_statement:expStmt
                  {: RESULT = new TAssignment(id, expStmt); :}
```

```
                | identifier:id ASSIGN collection_initializer:collectionInit
                {: RESULT = new TAssignment(id, collectionInit); :}
                | LBRK identifier_list:idList RBRK ASSIGN function_call:functionCall
                {: RESULT = new TIdentifierCollectionAssignment(idList, functionCall); :}
                ;

boolean_expression ::= LPAR boolean_expression:boolExp RPAR
                {: RESULT = boolExp; :}
                | boolean_expression:leftBoolExp AND boolean_expression:rightBoolExp
                {: RESULT = new TLogicalBoolean(leftBoolExp, "&&", rightBoolExp); :}
                | boolean_expression:leftBoolExp OR boolean_expression:rightBoolExp
                {: RESULT = new TLogicalBoolean(leftBoolExp, "||", rightBoolExp); :}
                | expression_statement:leftExp EQ expression_statement:rightExp
                {: RESULT = new TRelationalBoolean(leftExp, "==", rightExp); :}
                | expression_statement:leftExp NEQ expression_statement:rightExp
                {: RESULT = new TRelationalBoolean(leftExp, "!=", rightExp); :}
                | expression_statement:leftExp GE expression_statement:rightExp
                {: RESULT = new TRelationalBoolean(leftExp, ">", rightExp); :}
                | expression_statement:leftExp LE expression_statement:rightExp
                {: RESULT = new TRelationalBoolean(leftExp, "<", rightExp); :}
                | expression_statement:leftExp GEQ expression_statement:rightExp
                {: RESULT = new TRelationalBoolean(leftExp, ">=", rightExp); :}
                | expression_statement:leftExp LEQ expression_statement:rightExp
                {: RESULT = new TRelationalBoolean(leftExp, "<=", rightExp); :}
                ;

collection_initializer ::= array_initializer:arrayInit
                {: RESULT = arrayInit; :}
                | matrix_initializer:matInit
                {: RESULT = matInit; :}
                | cube_initializer:cubeInit
                {: RESULT = cubeInit; :}
```

```
                ;

array_initializer ::= LBRC number_list:numList RBRC
                {: RESULT = new TArrayInitializer(numList); :}
                ;

array_initializer_list ::= array_initializer:arrayInit
                {: RESULT = new TArrayInitializerList(arrayInit); :}
                | array_initializer_list:arrayInitList COMMA array_initializer:arrayInit
                {: RESULT = new TArrayInitializerList(arrayInitList, arrayInit); :}
                ;

matrix_initializer ::= LBRC array_initializer_list:arrayInitList RBRC
                {: RESULT = new TMatrixInitializer(arrayInitList); :}
                ;

matrix_initializer_list ::= LBRC matrix_initializer:matInit RBRC
                {: RESULT = new TMatrixInitializerList(matInit); :}
                | LBRC matrix_initializer_list:matInitList COMMA matrix_initializer:matInit RBRC
                {: RESULT = new TMatrixInitializerList(matInitList, matInit); :}
                ;

cube_initializer ::= LBRC matrix_initializer_list:matInitList RBRC
                {: RESULT = new TCubeInitializer(matInitList); :}
                ;

identifier ::= ID:id
                {: RESULT = new TIdentifier(id); :}
                | ID:id LBRK expression_statement:expStmt RBRK
                {: RESULT = new TIdentifier(id, expStmt); :}
                | ID:id LBRK expression_statement:expStmt1 COMMA
expression_statement:expStmt2 RBRK
```

```
                {: RESULT = new TIdentifier(id, expStmt1, expStmt2); :}
                | ID:id LBRK expression_statement:expStmt1 COMMA
expression_statement:expStmt2 COMMA expression_statement:expStmt3 RBRK
                {: RESULT = new TIdentifier(id, expStmt1, expStmt2, expStmt3); :}
                ;

number ::= REAL:real
                {: RESULT = new TNumber(real, false); :}
                | COMPLEX:complex
                {: RESULT = new TNumber(complex, true); :}
                ;

string ::= STRING:str
                {: RESULT = new TString(str); :}
                ;
```