รายงานการวิจัยเรื่อง

การรักษาความปลอดภัยของข้อมูล
โดยการเข้ารหัสข้อความลงบนภาพ

โดย
นายธีรวัฒน์ ประกอบผล

โครงการทุนพัฒนานักวิจัยคณะวิทยาศาสตร์ โดยใช้เงินรายได้
ประจำปีงบประมาณ พ.ศ. 2545
คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

การรักษาความปลอดภัยของข้อมูลโดยการเข้ารหัสข้อความลงบนภาพ

บทคัดย่อ

รูปภาพต่างๆ ที่จัดเก็บอยู่ในเครื่องคอมพิวเตอร์นั้นส่วนใหญ่ผู้ที่เป็นเจ้าของไม่สามารถหาหลักฐานอ้างได้ว่าภาพนั้นเป็นของตน และการจะอธิบายรายละเอียดของภาพก็มีข้อจำกัด โดยเฉพาะภาพที่มีรายละเอียดมากๆ หรือภาพที่ต้องการรายละเอียดสำหรับแต่ละส่วน ดังนั้นในงานวิจัยเรื่องการรักษาความปลอดภัยของข้อมูลโดยการเข้ารหัสข้อความลงในภาพนี้จะช่วยในการสร้างคำอธิบายรายละเอียดและจัดเก็บลงในส่วนต่างๆของภาพเพื่อให้ภาพที่จัดเก็บนั้นนำมาใช้ได้อย่างมีประสิทธิภาพมากขึ้น

งานวิจัยนี้ได้ประยุกต์ใช้ความรู้ทางด้านคณิตศาสตร์ในการคำนวณหาจุดของภาพที่จะใช้ในการจัดเก็บข้อมูลหลังจากเลือกส่วนของภาพที่ต้องการจะใส่รายละเอียด นำความรู้ทางด้านการบีบอัดตัวอักษรแบบ Huffman มาใช้ในการลดขนาดข้อมูลที่จะจัดเก็บลงในภาพ และได้ศึกษาโครงสร้างของแฟ้มข้อมูลภาพแบบบิตแมพ (Bitmap) รวมทั้งวิธีการซ่อนข้อมูล (Data hiding) แบบ LSB (Least Significant Bit) ซึ่งอาศัยการซ่อนข้อมูลลงไปในบิตที่มีผลกระทบน้อยที่สุดของข้อมูลที่เป็นสื่อ การศึกษาดังกล่าวก็เพื่อที่จะนำข้อมูลจัดเก็บลงในข้อมูลสีของภาพโดยที่ทำให้ข้อมูลภาพเปลี่ยนแปลงน้อยที่สุด โครงงานนี้ได้ใช้ภาษา C++ และ IDE ของ Microsoft Visual C++ 6.0 เป็นเครื่องมือสำคัญในการพัฒนา การทำงานของโปรแกรมจะแบ่งได้เป็น 2 ส่วนใหญ่ๆ คือ ส่วนซ่อนข้อมูลลงในภาพ และส่วนแสดงข้อมูลที่ซ่อนอยู่ในส่วนต่างๆของภาพ

ABSTRACT

Digital images as stored in a computer, usually has insufficient detail to serve need. Although information about the image can be partly described by a file name and file folder, this is still not the effective way, especially for images with a lot of detail. This special project studies the use of information hiding techniques. The purpose is to allow a user to create a description for an individual part of image. Therefore, pictures can be more effectively and conveniently used.

This special project applies mathematical knowledge to determine which points to use as storage area after selecting the picture segment. Furthermore, Huffman coding method is applied to reduce the size of information. In addition, structure of a bitmap file format and LSB (Least Significant Bit) data hiding technique has been studied and applied to make difference between original and post picture imperceptible. In addition Microsoft Visual C++ version 6.0 were chosen as a main developing tool. The program consists of two main parts, which are the information hiding process and the information display process

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

บทที่ 1
บทนำ (Introduction)

1.1 ความเป็นมาและความสำคัญของปัญหา (STATEMENT AND SIGNIFICANCE OF THE PROBLEMS)

รูปภาพต่างๆที่ถูกจัดเก็บในคอมพิวเตอร์ ไม่ว่าจะเป็น ภาพเหตุการณ์ ภาพวิวทิวทัศน์ หรือภาพเฉพาะทาง เช่น ภาพถ่ายทางการแพทย์ ภาพถ่ายดาวเทียม บางครั้งข้อมูลทั่วไป เช่น ชื่อไฟล์ ขนาดไฟล์ รายละเอียดเกี่ยวกับวัน-เวลา หรือแม้กระทั่งชื่อของโฟลเดอร์ที่ปรากฏอยู่ในคอมพิวเตอร์นั้นอาจจะไม่เพียงพอที่จะอธิบายรายละเอียดของภาพ โดยเฉพาะภาพที่ต้องการการอธิบายรายละเอียดปลีกย่อยภายในภาพ ดังนั้นปัญหาพิเศษนี้จะเป็นการนำความรู้ทางด้านการบีบอัด และการจัดเก็บข้อมูลภาพ มาประยุกต์ใช้ในการทำซอฟต์แวร์เพื่อจัดเก็บคำอธิบายต่างๆ ลงในภาพ โดยไม่มีผลกระทบต่อขนาดของไฟล์ ตัวอย่างเช่น ถ้าหากเรามีภาพแผนที่ประเทศไทย และต้องการบอกรายละเอียดของแต่ละจังหวัดที่ไม่ใช่แค่ชื่อจังหวัดเพียงอย่างเดียว เช่น ต้องการจะบอกถึงจำนวนประชากรที่อาศัยอยู่ สภาพภูมิอากาศ คำขวัญประจำจังหวัด เราก็สามารถที่จะจัดเก็บลงไปในภาพได้ โดยในตัวโปรแกรมจะแบ่งเป็น 2 ส่วน คือ ส่วนที่ให้ผู้ใช้ใส่รายละเอียดของส่วนต่างๆของภาพลงไป กับส่วนที่แสดงภาพซึ่งเมื่อผู้ใช้เลื่อนเมาส์ไปวางบนพื้นที่ที่ได้เก็บรายละเอียด ก็จะปรากฏรายละเอียดขึ้นมา

1.2 ความมุ่งหมายและวัตถุประสงค์ของการวิจัย

1. เพื่อใช้ในการจัดเก็บรายละเอียดลงในส่วนต่างๆ ของภาพที่ต้องการและสามารถจะเรียกดูรายละเอียดในส่วนต่างของภาพได้โดยสะดวก
2. ศึกษาการบีบอัดข้อมูลรวมถึงการจัดเก็บข้อมูลภาพ เพื่อนำมาประยุกต์ใช้ในโครงงาน
3. เพื่อนำไปพัฒนาต่อให้สามารถจัดเก็บข้อมูลในรูปแบบอื่นเช่น เสียง นอกเหนือจากข้อมูลที่เป็นข้อความอย่างเดียว

1.3 สมมติฐานของการศึกษา (HYPOTHESIS TO BE TESTED)

1. การใส่รายละเอียดต่างๆ ลงในจุดสีของภาพ จะทำให้ภาพเกิดการเปลี่ยนแปลงไปจากเดิมเพียงเล็กน้อย ซึ่งมีผลน้อยมากต่อการรับรู้ของมนุษย์
2. การเปลี่ยนแปลงแก้ไขภาพที่ใส่รายละเอียดไปแล้วนั้น จะมีผลกับข้อมูลภาพเฉพาะส่วนที่ถูกทำการเปลี่ยนแปลงแก้ไขเท่านั้น

1.4 ขอบเขตของการวิจัย

ในหัวข้อวิจัยนี้ได้นำเอาความรู้เกี่ยวกับ Image Technology, Compression Technology ที่ได้รับจากการศึกษามาใช้ในการพัฒนาโปรแกรมคอมพิวเตอร์ ซึ่งลักษณะของโปรแกรมที่จะพัฒนาขึ้นมามีลักษณะดังต่อไปนี้

1. สามารถใส่รายละเอียดต่างๆ แล้วจัดเก็บลงในภาพ พร้อมทั้งสามารถแสดงรายละเอียดต่างๆภายในภาพ

2. รายละเอียดที่ผู้ใช้กรอกลงในภาพ และรายละเอียดที่แสดงออกมาจากภาพนั้นจะต้องไม่มีการเปลี่ยนแปลง และคงรูปแบบข้อความไว้ทุกประการ

3. ไฟล์ภาพที่จะนำมาใช้เก็บรายละเอียดต่างๆลงไปนั้นจะต้องเป็นไฟล์ชนิด 24 - bit Bitmap (.bmp) เท่านั้น และหลังจากผ่านการเก็บรายละเอียดต่างๆ ลงในภาพแล้ว ขนาดของไฟล์ภาพจะต้องมีขนาดเท่าเดิมไม่เปลี่ยนแปลง

1.5 ขั้นตอนของการวิจัย

การศึกษาเพื่อทำปัญหาพิเศษนี้ได้แบ่งออกเป็น ส่วนดังนี้

1. ศึกษาการจัดเก็บข้อมูลภาพชนิดต่างๆว่ามีลักษณะการจัดเก็บเป็นอย่างไร เพื่อนำมาพิจารณาถึงความเป็นไปได้ที่จะนำข้อมูลไปฝัง

2. ศึกษาวิธีการต่างๆ ที่ใช้ในการบีบอัดข้อความที่มีใช้ในปัจจุบัน เพื่อนำมาประยุกต์ใช้ในการทำปัญหาพิเศษครั้งนี้

3. ศึกษาการใช้โปรแกรม Microsoft Visual C++ เพื่อใช้ในการพัฒนาโปรแกรม

4. วิเคราะห์และออกแบบการทำงานของโปรแกรม โดยการเอาวิธีการทางคอมพิวเตอร์ที่ได้ศึกษามา เข้ามาช่วยในการวิเคราะห์และออกแบบขั้นตอนและโครงสร้างของโปรแกรม โดยจะทำการแบ่งงานออกเป็นส่วนๆ เช่น ส่วนรับข้อมูล ส่วนประมวลผล และส่วนแสดงผล

5. ทำการพัฒนาโปรแกรมตามที่ได้ออกแบบเอาไว้ในขั้นตอนของการวิเคราะห์และออก - แบบการทำงานของโปรแกรม

6. ทำการทดสอบโปรแกรม และปรับปรุงการทำงานของโปรแกรมให้มีประสิทธิภาพตามที่ต้องการ และบอกถึงความสามารถทั้งหมด รวมถึงข้อจำกัดต่างๆของโปรแกรม

1.6 ข้อตกลงเบื้องต้น (ASSUMPTION)

ในการทำปัญหาพิเศษนี้ ภาพที่จะนำมาจัดเก็บรายละเอียดนั้นจะต้องเป็นไฟล์ภาพที่เป็นชนิด 24-bit Bitmap เท่านั้น (.bmp) และส่วนของภาพที่ได้เลือกไว้และจัดเก็บข้อความไปแล้ว จะไม่สามารถนำมาจัดเก็บข้อความอื่นได้อีก แต่สามารถที่จะแก้ไขข้อควรามหรือยกเลิกการจัดเก็บข้อความในส่วนของภาพนั้นๆได้

1.7 ข้อจำกัดของการวิจัย

1. ในการที่จะพัฒนาโปรแกรมให้ใช้งานกับไฟล์ภาพชนิดอื่น ๆ เช่น jpeg จะต้องใช้เวลาในการศึกษามากกว่านี้ เพื่อที่จะสามารถรักษาวัตถุประสงค์และสมมติฐานของการทำปัญหาพิเศษเอาไว้ได้

2. ภาพที่ได้ใส่รายละเอียดไว้แล้วนั้น ถ้าถูกดำเนินการต่าง ๆ เช่น ปรับแต่งภาพ หรือทำการเปลี่ยนชนิดของไฟล์ภาพ (convert) จะไม่สามารถเรียกดูรายละเอียดที่ได้จัดเก็บเอาไว้

บทที่2
ทฤษฎีที่เกี่ยวข้อง

ทฤษฎีและแนวคิดที่มีความสำคัญกับงานวิจัยนี้ได้แก่ โครงสร้างไฟล์ภาพ Bitmap, ทฤษฎีที่เกี่ยวกับการบีบอัด, แนวคิดในการจัดเก็บข้อความลงในภาพ และทฤษฎีการมองเห็นของมนุษย์

2.1 โครงสร้างไฟล์ภาพ Bitmap (Bitmap–File Formats)

ไฟล์ Windows bitmap จะถูกจัดเก็บอยู่ในรูปแบบของ device–independent bitmap (DIB) ซึ่งการจัดการในลักษณะดังกล่าวจะทำให้ Windows สามารถที่จะแสดงภาพ bitmap ดังกล่าวบน display device แบบใดๆก็ได้ โดยคำว่า "device independent" หมายความว่า bitmap จะระบุ pixel color ในรูปแบบที่ไม่ขึ้นกับวิธีที่ถูกใช้ในการแสดงสีดังกล่าว โดย default filename extension ของ Windows DIB file ก็คือ .BMP

2.1.1 Bitmap–File Structures

โครงสร้างของ bitmap file จะประกอบด้วย

- a bitmap–file header
- a bitmap–information header
- a color table
- an array of bytes that defines the bitmap bits

โดยไฟล์ bitmap จะอยู่ใน form ดังต่อไปนี้

```
BITMAPFILEHEADER  bmfh;
BITMAPINFOHEADER  bmih;
RGB               aColors[];
BYTE              aBitmapBits[];
```

bitmap–file header จะเป็นส่วนของข้อมูลที่เกี่ยวกับ type, size และ layout of a device-independent bitmap file โดย bitmap–file header จะถูกนิยามในรูปของ BITMAPFILEHEADER structure

BITMAPFILEHEADER (Windows Bitmap 3.0)

```
typedef struct tagBITMAPFILEHEADER { /*bmfh*/
    UINT   bfType;
    DWORD  bfSize;
```

```
    UINT   bfReserved1;
    UINT   bfReserved2;
    DWORD  bfOffBits;
} BITMAPFILEHEADER;
```

Member	*Description*
bfType	ระบุชนิดของไฟล์ ซึ่งจะต้องมีค่าเป็น BM
bfSize	ระบุขนาดของไฟล์มีหน่วยเป็น bytes
bfReserved1	สงวนเอาไว้ ต้องมีค่าเป็น 0
bfReserved2	สงวนเอาไว้ ต้องมีค่าเป็น 0
bfOffBits	ระบุ byte offset จาก BITMAPFILEHEADER ไปยัง actual bitmap data
bitmap-information header	จะถูกนิยาม ในรูปแบบของ BITMAPINFOHEADER structure ซึ่งจะระบุ dimensions, compression type และ color format

BITMAPINFOHEADER (Windows Bitmap 3.0)

```
typedef struct tagBITMAPINFOHEADER { /* bmih */
    DWORD  biSize;
    LONG   biWidth;
    LONG   biHeight;
    WORD   biPlanes;
    WORD   biBitCount;
    DWORD  biCompression;
    DWORD  biSizeImage;
    LONG   biXPelsPerMeter;
    LONG   biYPelsPerMeter;
    DWORD  biClrUsed;
    DWORD  biClrImportant;
} BITMAPINFOHEADER;
```

Member	*Description*
biSize	ระบุจำนวน bytes ที่ใช้ใน BITMAPINFOHEADER
biWidth	ระบุความกว้างของภาพ bitmap ในหน่วย pixel

biHeight	ระบุความสูงของภาพ bitmap ในหน่วย pixel
biPlanes	ระบุจำนวน planes สำหรับ target device โดยต้องมีค่าเป็น 1
biBitCount	ระบุจำนวน bits per pixel โดยค่าที่สามารถเป็นไปได้คือ 1, 4, 8 หรือ 24
biCompression	ระบุ type ของการบีบอัดสำหรับ compressed bitmap โดยมีค่าที่เป็นไปได้ดังต่อไปนี้
	BI_RGB ระบุว่าภาพ bitmap นั้นไม่มีการบีบอัด
	BI_RLE8 ระบุว่า มีการใช้ run-length encoded format สำหรับภาพ bitmap แบบ 8 - bits per pixel
	BI_RLE4 ระบุว่ามีการใช้ run-length encoded format สำหรับภาพ bitmap แบบ 4 - bits per pixel
biSizeImage	ระบุขนาดของภาพในหน่วยเป็น bytes โดยถ้าเป็นภาพ bitmap แบบ BI_RGB format สามารถที่จะกำหนดค่า biSizeImage ให้เป็น 0 ได้
biXPelsPerMeter	ระบุ horizontal resolution ของ target device ในหน่วย pixel per meter
biYPelsPerMeter	ระบุ vertical resolution ของ target device ในหน่วย pixel per meter
biClrUsed	ระบุจำนวน color index ที่ถูกใช้จริงๆในภาพ bitmap แต่ถ้าค่าเป็น 0 จะหมายความว่า จะใช้จำนวนสีเท่ากับจำนวนสีสูงสุด
biClrImportant	ระบุจำนวน color index ที่ถูกพิจารณาว่าจะมีความสำคัญสำหรับการแสดงภาพ bitmap ถ้าเป็น 0 แสดงว่าทุกๆสีมีความสำคัญหมด

color table จะถูกนิยามในรูปของ array of RGBQUAD structure ซึ่งจะเป็นส่วนที่จัดเก็บ elements ของสีทั้งหมดที่ใช้ใน bitmap อย่างไรก็ตามในภาพ bitmap แบบ 24 bits จะไม่มีการใช้ตารางสี เพราะว่าทุกๆ pixel ในภาพ จะแทนด้วย 24-bit red-green-blue (RGB) เลย จึงไม่จำเป็นต้องใช้ตารางสี color ที่เก็บอยู่ใน color table ควรจะจัดเก็บโดยเรียงตามลำดับ

ความสำคัญ เพราะจะช่วยให้ display driver สามารถประมวลผลภาพได้ในกรณีที่ device นั้นๆ ไม่สามารถแสดงสีได้เท่ากับจำนวนสีในภาพ bitmap ที่ต้องการเปิด ซึ่งถ้าเป็น Windows DIB version 3.0 หรือ version ต่อๆมา device driver สามารถใช้ biClrImportant ใน BITMAPFILEHEADER ในการตัดสินใจว่า colors ไหนที่มีความสำคัญ

RGBQUAD (Windows Bitmap 3.0)

```
typedef struct tagRGBQUAD { /* rgbq */
        BYTE   rgbBlue;
        BYTE   rgbGreen;
        BYTE   rgbRed;
        BYTE   rgbReserved;
} RGBQUAD;
```

Member	*Description*
rgbBlue	ระบุความเข้มของสีน้ำเงิน
rgbGreen	ระบุความเข้มของสีเขียว
rgbRed	ระบุความเข้มของสีแดง
rgbReserved	ส่วนนี้ไม่ถูกใช้ และจะต้องกำหนดให้มีค่าเป็น 0

เราสามารถใช้ BITMAPINFO structure แทน ในส่วนของ bitmap-information header และ color table ได้

BITMAPINFO (Windows Bitmap 3.0)

```
typedef struct tagBITMAPINFO { /* bmi */
        BITMAPINFOHEADER  bmiHeader;
        RGBQUAD           bmiColors[ ];
} BITMAPINFO;
```

Member	*Description*
bmiHeader	เป็น BITMAPINFOHEADER structure ที่เก็บข้อมูลเกี่ยวกับ dimensions, compression type และ color format ของ DIB
bmiColors	เป็น array ของ RGBQUADstructure ซึ่งกำหนดสีใน bitmap

bitmap bits จะเป็นส่วนของ byte array ที่แสดงข้อมูลภาพซึ่งจะจัดเก็บในลักษณะแบบ consecutive rows หรือ “scan line” scan line ของภาพ bitmap จะถูกจัดเก็บจากล่างขึ้นบน โดยในแต่ละ scan line จะประกอบด้วย consecutive bytes ซึ่งเป็น pixel ใน scan line ซึ่งถูกจัดเก็บในลำดับจากซ้ายไปขวา ซึ่งหมายความว่า byte แรกใน array จะเป็น lower-left corner pixel ของ bitmap และ byte สุดท้ายใน array จะเป็น upper-right corner pixel โดยจำนวนของ bytes ใน scan line จะขึ้นอยู่กับ color format และ width ของภาพ

ใน BITMAPINFOHEADER จะมี biBitCount ที่ใช้ในการระบุว่าจะต้องใช้ข้อมูลจำนวนกี่ bit ในการกำหนดจุด pixel ในภาพ และเป็นการบอกจำนวนสีที่มากที่สุดของ bitmap โดย biBitCount จะมีค่าที่สามารถที่เป็นไปได้ดังต่อไปนี้

Value	*Meaning*
1	จะเป็น Bitmap แบบ Monochrome และ color table จะมี 2 entries โดยทุกๆ bit ใน bitmap array จะแทนแต่ละ pixel
4	Bitmap จะมี maximum color เป็น 16 สี แต่ละ pixel ใน bitmap จะถูกแทนโดยข้อมูล 4-bit ที่ชี้ไปยัง color table
8	Bitmap จะมี maximum color เป็น 256 สี แต่ละ pixel ใน bitmap จะถูกแทนโดยข้อมูล 8-bit (1 byte) ที่ชี้ไปยัง color table
24	Bitmap จะมี maximum color เป็น 2^{24} สี และทุกๆ 3-byte sequence ใน bitmap array จะแทนความเข้มสัมพัทธ์ของสีแดง สีเขียว และสีน้ำเงินของแต่ละ pixel

2.1.2 Bitmap Compression

Windows Bitmap version 3.0 และ version ต่อๆมาจะสนับสนุน run-length encoded (RLE) formats เพื่อใช้สำหรับในการบีบอัด bitmap ที่เป็นแบบ 4-bits per pixel และแบบ 8-bits per pixel ซึ่งการบีบอัดจะทำให้ลดการใช้ disk และ memory storage

2.1.2.1 Compression of 8-Bits-per-Pixel Bitmaps

ถ้าค่า biCompression ใน BITMAPINFOHEADER มีค่าเป็น BI_RLE8 แสดงว่ามีการบีบอัดโดยใช้ run-length encoded format สำหรับภาพ bitmap แบบ 256 สี โดยใน format นี้จะมีอยู่ 2 โหมด คือ encoded mode และ absolute mode ซึ่งทั้ง 2 โหมดดังกล่าวจะปรากฏในส่วนใดของภาพ bitmap หนึ่งๆก็ได้

encoded mode

จะใช้ข้อมูลทีละ 2 byte โดย byte แรก จะระบุจำนวน consecutive pixel ที่จะถูกวาดโดยใช้ color index ใน byte ที่สอง อย่างไรก็ตามใน byte ที่หนึ่งนั้นสามารถมีค่าเป็น 0x00 ซึ่งระบุว่าเป็นตัว escape โดยค่าใน byte ที่สองที่ตามมาจะต้องมีค่าอยู่ระหว่าง 0x00 และ 0x02 โดยถ้ามีค่าเป็น 0x00 จะหมายถึง end of line ถ้ามีค่าเป็น 0x01 จะหมายถึง end of bitmap และถ้ามีค่าเป็น 0x02 จะหมายถึง delta นั่นคือ อีก 2 byte มี่ตามมาจะเป็นค่าที่จะระบุ horizontal และ vertical offset จากตำแหน่งของ pixel ถัดไปที่จะวาด

absolute mode

จะใช้ข้อมูลทีละ 2 byte โดย byte แรก จะต้องมีค่าเป็น 0x00 และใน byte ที่สองที่ตามมาจะต้องมีค่าอยู่ระหว่าง 0x03 และ 0xFF โดยตัวเลขใน byte ที่สองนี้จะบอกถึงจำนวนของ bytes ที่ตามมาซึ่งค่าของแต่ละ byte จะเป็น color index ของ 1 pixel ที่จะวาด โดยจำนวนข้อมูลในส่วนนี้จะต้องมีจำนวนที่เป็นจำนวนเท่าของ word boundary

ตัวอย่างของ 8-bit RLE bitmap (โดยค่าตัวเลขฐาน 16 สองหลัก ที่อยู่ในคอลัมน์ที่สองจะเป็น color index สำหรับ 1 pixel)

Compressed data	*Expanded data*
03 04	04 04 04
05 06	06 06 06 06 06
00 03 45 56 67 00	45 56 67
02 78	78 78
00 02 05 01	Moves 5 right and 1 down
02 78	78 78
00 00	End of line
09 1E	1E 1E 1E 1E 1E 1E 1E 1E 1E
00 01	End of RLE bitmap

2.1.2.2 Compression of 4-Bits-per-Pixel Bitmaps

ถ้าค่า biCompression ใน BITMAPINFOHEADER มีค่าเป็น BI_RLE4 แสดงว่ามีการบีบอัดโดยใช้ run-length encoded format สำหรับภาพ bitmap แบบ 16 สี โดยใน format นี้จะมีอยู่ 2 โหมด คือ encoded mode และ absolute mode ซึ่งทั้ง 2 โหมดดังกล่าวจะปรากฏในส่วนใดของภาพ bitmap หนึ่งๆก็ได้

encoded mode

จะใช้ข้อมูลทีละ 2 byte โดย byte แรก จะระบุจำนวน pixel ที่จะถูกวาดโดยใช้ color index ใน byte ที่สอง ซึ่งใน byte ที่สองจะมี color index อยู่ 2 index โดย color index แรกจะอยู่ที่ high-order nibble ส่วนอีก color index จะอยู่ที่ low-order nibble

pixel แรกจะถูกวาดโดยใช้สีที่มี color index อยู่ในส่วนของ high-order nibble, pixel ที่สองจะถูกวาดโดยใช้สีที่มี color index อยู่ในส่วนของ low-order nibble, pixel ที่สามจะถูกวาดโดยใช้สีที่มี color index อยู่ในส่วนของ high-order nibble เป็นเช่นนี้เรื่อยไปจนกระทั่งจำนวน pixel ที่ถูกวาดครบตามที่ระบุไว้ในข้อมูล byte แรก

อย่างไรก็ตามใน byte ที่หนึ่งนั้นสามารถมีค่าเป็น 0x00 ซึ่งระบุว่าเป็นตัว escape โดยค่าใน byte ที่สองที่ตามมาจะต้องมีค่าอยู่ระหว่าง 0x00 และ 0x02 โดยถ้ามีค่าเป็น 0x00 จะหมายถึง end of line ถ้ามีค่าเป็น 0x01 จะหมายถึง end of bitmap และถ้ามีค่าเป็น 0x02 จะหมายถึง delta นั่นคือ อีก 2 byte มี่ตามมาจะเป็นค่าที่จะระบุ horizontal และ vertical offset จากตำแหน่งของ pixel ถัดไปที่จะวาด

absolute mode

จะใช้ข้อมูลทีละ 2 byte โดย byte แรก จะต้องมีค่าเป็น 0x00 และใน byte ที่สองที่ตามมาจะต้องมีค่าอยู่ระหว่าง 0x03 และ 0xFF โดยตัวเลขใน byte ที่สองนี้จะบอกถึงจำนวนของ color index ที่จะตามมา ซึ่งค่าของแต่ละ byte ที่ตามมาจะมี 2 color index คือ high-order nibble และ low-order nibble โดย 1 color index ก็จะสำหรับ 1 pixel และจำนวนข้อมูลในส่วนนี้จะต้องมีจำนวนที่เป็นจำนวนเท่าของ word boundary

ตัวอย่างของ 8-bit RLE bitmap (โดยค่าตัวเลขฐาน 16 หนึ่งหลัก ที่อยู่ในคอลัมน์ที่สองจะเป็น color index สำหรับ 1 pixel)

Compressed data	*Expanded data*
03 04	0 4 0
05 06	0 6 0 6 0
00 06 45 56 67 00	4 5 5 6 6 7
04 78	7 8 7 8
00 02 05 01	Moves 5 right and 1 down
04 78	7 8 7 8
00 00	End of line
09 1E	1 E 1 E 1 E 1 E 1
00 01	End of RLE bitmap

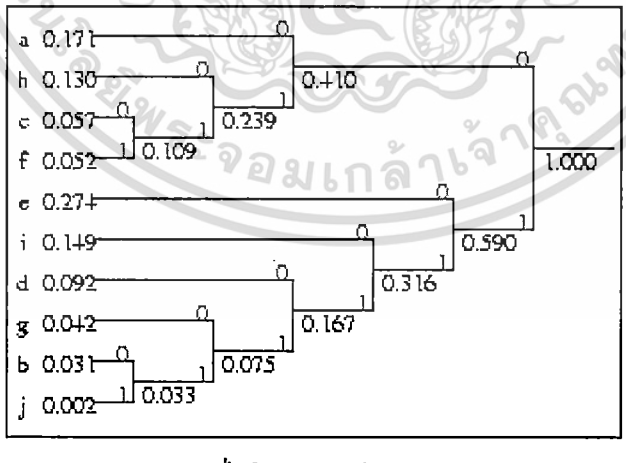
2.2 ทฤษฎีที่เกี่ยวกับการเข้ารหัสแบบ Huffman coding

2.2.1 นิยาม

การเข้ารหัสแบบ Huffman coding นั้นเป็นการบีบอัดแบบไม่มีการสูญหายของข้อมูล (Lossless compression) ซึ่งตั้งชื่อตามผู้คิดค้น David Huffman หลักการก็คือแทนที่ตัวอักษรที่ปรากฏบ่อยครั้งที่สุด (Most occurrence) ด้วยบิตข้อมูลที่สั้นที่สุดเท่าที่จะเป็นไปได้

2.2.2 วิธีการเข้ารหัสแบบ Huffman coding

วิธีการของการเข้ารหัสแบบนี้จะเริ่มต้นด้วยการนำตัวอักษรทั้งหมดที่ปรากฏอยู่ในข้อความมาหาค่า Occurrence สมมติว่า เรามีข้อมูลที่ประกอบขึ้นจาก Set ของตัวอักษร a,b,c,d,e,f,g,h,i และ j จากนั้นเราก็ทำการหาค่า Occurrence ของตัวอักษรแต่ละตัวในที่นี้คือ P(x) และทำการเรียงลำดับ (Sort) จากตัวอักษรที่มีค่า P(x) สูงสุดไปยังตัวอักษรที่มีค่า P(x) ต่ำสุด จากนั้นก็ทำการสร้างแผนผังต้นไม้แบบ Huffman (Huffman tree) โดยการสร้างนั้นทำได้โดยนำค่า P(x) ของตัวอักษรที่มีค่า P(x) น้อยที่สุด 2 ตัวมาบวกกัน และทำเครื่องหมายว่าตัวอักษรดังกล่าวได้เลือกไปแล้ว จากนั้นทำการตรวจสอบหาค่าตัวอักษร 2 ตัวที่มีค่า P(x) น้อยที่สุดและยังไม่ได้เลือก ทำไปจนเลือกครบทุกตัวอักษรจากนั้นทำการกำหนดเลข 0 และ 1 ให้แต่ละเส้นทางลงใน Tree โดยในที่นี้จะให้เส้นบนเป็น 0 เส้นล่างเป็น 1 ตัวเลข 0 หรือ 1 ที่กำหนดนี้จะเป็นตัวกำหนดว่า ตัวอักษรแต่ละตัวจะเข้ารหัสในรูปเลขฐานสองอย่างไร



รูปที่ 2.2.2.1 แสดง Huffman Tree ที่ได้สำหรับข้อมูลข้างต้น

ถึงตรงนี้เราก็จะได้ผลลัพธ์ของการเข้ารหัสแบบ Huffman coding โดยดูเส้นทางจากส่วนปลายด้านขวาที่จะเรียกกันว่า Root ไปยังส่วนปลายด้านซ้ายที่เป็นตัวอักษรก็จะได้รหัสสำหรับตัวอักษรแต่ละตัว ซึ่งแสดงได้ดังนี้

X	Code
a	00
b	111110
c	0110
d	1110
e	10
f	0111
g	11110
h	010
i	110
j	111111

รูปที่ 2.2.2.2 **แสดงผลลัพธ์ที่ได้จาก** Huffman Tree

จากนั้นก็แทนตัวอักษรในข้อมูลด้วยบิตข้อมูลที่ได้ เช่น ตัวอักษร a นั้นตามมาตรฐาน ASCII คือ 97 (ฐาน 10) หรือเขียนในรูปแบบเลขฐานสองคือ 01100001 แต่หลังจากที่เข้ารหัสตามผลลัพธ์ที่ได้ก็จะมีขนาดเพียง 2 บิตคือ 00 การเข้ารหัสแบบนี้จะเป็นการรับประกันว่าจะไม่มีรหัสของตัวอักษรใดๆ ซ้ำกัน และจะไม่มีรหัสของตัวอักษรใดเป็น Prefix (เช่น ถ้าตัวอักษร a เข้ารหัสเป็น 00 แล้วจะไม่มีตัวอักษรใดขึ้นด้วย 00)ของตัวอักษรอื่น

อย่างไรก็ตามจะเห็นได้ว่าการเข้ารหัสแบบ Huffman coding นั้นต้องมีการจัดเก็บผลลัพธ์ที่ได้จากการสร้าง Huffman Tree ในการที่จะถอดรหัส ซึ่งถือว่าเป็น Overhead ของข้อมูล และเนื่องจากข้อมูลที่มีขนาดเล็กซึ่งหมายถึงประโยคสั้นๆ (โดยทั่วไปข้อมูลที่มีขนาดเล็กจำนวนของตัวอักษรที่ใช้ก็มีโอกาสซ้ำกันน้อย) นั้นเมื่อใช้วิธีการบีบอัดแบบ Huffman coding จะมีส่วนที่เป็น Overhead อยู่มากเมื่อเทียบกับตัวข้อมูล และเนื่องจากว่าในโครงงานนี้ข้อความที่จัดเก็บลงในส่วนต่างๆ มีความเป็นไปได้สูงที่ผู้ใช้จะใส่ข้อความที่มีขนาดเล็กๆ เช่นประโยคสั้นๆ ที่มีขนาดไม่เกิน 3 บรรทัด ดังนั้นด้วยเหตุดังกล่าวการกำจัดส่วนที่เป็น Overhead นั้นจึงมีความสำคัญ

2.2.3 วิธีการเข้ารหัสที่นำมาใช้ในโครงงาน

ดังนั้นในการที่โครงงานนี้จะทำให้ข้อมูลที่เป็น Overhead หมดไปก็โดยการทำให้ส่วนที่เป็น Overhead นั้นอยู่ในรูปข้อมูลที่ไม่แปรผันไปตามชุดข้อมูล กล่าวคือไม่ว่าจะมีข้อมูลอย่างไรก็ไม่จำเป็นต้องใส่ตารางผลลัพธ์ไปกับชุดข้อมูล แต่การจะกำหนดผลลัพธ์ให้เหมาะสมกับข้อมูลทุกรูปแบบนั้นทำได้ยาก ยกตัวอย่างเช่น ถ้าเกิดเรานำผลลัพธ์ที่ได้จากตัวอย่างที่แล้วไปใช้เข้ารหัส

กับข้อมูลที่มีค่า P(x) ของตัวอักษรตรงกันข้ามกับตัวอย่างข้างต้น ก็จะทำให้ประสิทธิภาพของการเข้ารหัสแบบ Huffman Coding นั้นลดลงไป

แต่ถ้าเรามีตัวแบบที่ใกล้เคียงกับข้อมูลหลายๆชนิด แล้วเลือกได้ว่าข้อมูลแบบใดควรจะเข้ารหัสตามตัวแบบใด แม้จะมีผลต่อประสิทธิภาพการบีบอัดให้ลดน้อยลงไปแต่การกำจัด Overhead ตามวิธีนี้ก็เป็นการรับประกันว่าจะสามารถลดขนาดข้อมูลที่มีขนาดเล็กได้ และในขณะเดียวกันก็ลดขนาดข้อมูลที่มีขนาดใหญ่ได้ด้วย

โดยในโครงงานนี้จะทำการสร้างตัวแบบในการเข้ารหัสที่ใช้วิธีเดียวกันกับ Huffman Coding จำนวน 3 แบบคือ ตัวแบบสำหรับข้อมูลที่เป็นภาษาไทย ตัวแบบสำหรับข้อมูลที่เป็นภาษาอังกฤษ และตัวแบบสำหรับข้อมูลที่มีทั้งภาษาไทยและภาษาอังกฤษ โดยการสร้างตัวแบบนั้นจะใช้การหาค่า Occurrence ของตัวอักษรแต่ละตัวจากข้อมูลที่มีการใช้ตัวอักษรอย่างเป็นปกติ (ในโครงงานนี้จะนำข้อมูลที่เป็นภาษาไทยอย่างเดียว ภาษาอังกฤษอย่างเดียว และข้อมูลที่มีทั้งภาษาไทย และอังกฤษมาใช้เป็นตัวแบบ) และทำการกำหนดค่า P(x) ให้กับตัวอักษรแต่ละตัวในแต่ละตัวแบบ จากนั้นก็ใช้วิธีของ Huffman coding เพื่อหารหัสของตัวอักษรแต่ละตัวในแต่ละตัวแบบ

2.3 ทฤษฎี Steganography

2.3.1 นิยาม

เป็นศาสตร์ที่ว่าด้วยการซ่อนข้อมูลหนึ่งไว้ในอีกข้อมูลหนึ่งโดยมีวัตถุประสงค์ที่จะปิดบังไม่ให้ผู้อื่นสงสัยได้ว่ามีการซ่อนข้อมูลที่ต้องการจะส่งเอาไว้ โดยปกติแล้วเราจะทำการเข้ารหัสลับ (encrypt)ข้อมูลที่ต้องการจะซ่อนเสียก่อน ก่อนที่จะนำลงไปฝังในข้อมูลอื่น แต่ก็ไม่จำเป็นว่า Steganography จะต้องมีการเข้ารหัสลับอยู่ด้วยเสมอ เพราะการเข้ารหัสลับจะทำเพื่อเป็นการเพิ่มระดับของความปลอดภัยให้กับข้อความที่เราต้องการจะซ่อนเท่านั้น

ข้อดีของ Steganography ก็คือ เราสามารถรถส่งข้อมูลที่เป็นความลับได้โดยที่เป้าหมายของการส่งข้อมูลนั้นจะไม่ถูกเปิดเผยเนื่องจากเราซ่อนข้อมูลนั้นๆเอาไว้ในอีกข้อมูลหนึ่ง

ส่วนข้อด้อยของ Steganography ก็คือ เกิด overhead ในการที่จะซ่อนข้อมูลเพียงเล็กน้อยเอาไว้ในตัวข้อมูลอื่น

2.3.2 คำเฉพาะในเรื่อง Steganography

- Cover carrier คือ สื่อหรือข้อมูลที่จะใช้สำหรับจัดเก็บข้อมูลที่ต้องการจะซ่อน ซึ่ง Cover carrier อาจจะเป็น text , image, audio, video หรือสื่ออื่นก็ได้
- Embedded message คือข้อมูลที่เราต้องกาจะซ่อนเอาไว้ใน Cover carrier ซึ่งสามารถที่จะเป็น plaintext, ciphertext, image หรืออื่นๆที่มีการจัดเก็บในลักษณะ Bit stream
- Stego carrier คือผลลัพธ์ที่ได้จากการนำ embedded message มาฝังลงใน Cover carrier
- stegokey คือ เป็นข้อมูลที่เป็นความลับ เช่น password ที่เราเอาเข้ามาใช้เพื่อทำให้การซ่อนข้อมูลของเรามีความปลอดภัยมากยิ่งขึ้น

2.3.3 ข้อควรพิจารณาในการทำ Steganography

1. Cover carrier ไม่ควรที่จะเปลี่ยนแปลงเสียหาย (degrade) จนทำให้สามารถจะรับรู้ได้ว่ามีการซ่อน Embedded message เอาไว้
2. Embedded message ควรจะ encode อยู่ที่ตัวข้อมูลจริงๆของ cover carrier ไม่ควรจะอยู่ที่ header หรือ wrapper ของ cover carrier เพื่อรักษาความสอดคล้องของ format ของ Cover carrier ให้คงอยู่
3. ควรระวังในเรื่องความผิดเพี้ยนเปลี่ยนแปลงของ Embedded message เนื่องจากการที่ Cover carrier ถูกปรับเปลี่ยน (modify)

2.3.4 Steganography in Images

Image Steganography ถูกนำเอามาใช้ประโยชน์ เนื่องจากว่าความสามารถในการรับรู้ด้วยสายตาของมนุษย์ (Human visual system) นั้นมีขีดจำกัด โดยระดับความเข้มที่แตกต่างกันเพียงเล็กน้อยของสีสีเดียวกันจะไม่แตกต่างกันเลยในการรับรู้ของมนุษย์ ด้วยเหตุนี้เราจึงสามารถที่จะนำข้อมูลที่จัดเก็บอยู่ในรูปแบบ bit stream เช่น text หรือแม้แต่ image เองมาซ่อนไว้ใน image ที่เก็บอยู่ในคอมพิวเตอร์ได้

2.3.4.1 Image Encoding Technique

วิธีการส่วนใหญ่ที่ใช้ในการซ่อนข้อมูลลงในภาพมีอยู่ 3 วิธี คือ

1. Least Significant Bit (LSB) insertion
2. Masking and Filtering Technique
3. Algorithms and Transformations

แต่ในที่นี้จะกล่าวถึงเฉพาะวิธีที่จะนำไปใช้ในโครงงาน

Least Significant Bit (LSB) insertion

เป็น Image Steganography Technique ที่ถูกรู้จักมากที่สุดวิธีหนึ่ง หลักการทำงานของมันก็คือ จะนำข้อมูลแต่ละ bit ของ embedded message เข้าไปเก็บที่ least significant bit ของแต่ละ byte ข้อมูลใน cover image ข้อดีหลักของ LSB insertion คือ ข้อมูลสามารถแทรกเข้าไปใน least bit ได้โดยไม่มีผลกระทบต่อการรับรู้ของมนุษย์

โดยถ้าเป็น image แบบ 24-bits-per-pixel ก็แสดงว่าใน 1 pixel เราสามารถที่จะ encode ส่วนของข้อมูลที่เป็น embedded message เข้าไปได้ 3 bit นั่นคือถ้าเราต้องการ embed ตัวอักษร A เข้าไปเก็บใน 24-bit image เราก็ต้องใช้ทั้งหมด 3 pixel ดังแสดงข้างล่างนี้

(00100111 11101001 11001000) (00100111 11001000 11101001)

(11001000 00100111 11101001)

ถ้าเรานำ binary value ของตัวอักษร A คือ (01000001) แทรกเข้าไปใน 3 pixel ดังกล่าวโดยเริ่มจาก top left byte จะได้ดังนี้

(00100110 11101001 11001000) (00100110 11001000 11101000)

(11001000 00100111 11101001)

จะพบว่ามีเพียงข้อมูลใน bit ที่เป็นตัวเอียงเท่านั้นที่มีการเปลี่ยนแปลง

ถ้าเป็นภาพแบบที่เป็นการใช้ตารางสี ซึ่งก็คือ 1 bit image, 4 bit image และ 8 bit image การใช้วิธี LSB insertion กับข้อมูลภาพที่เป็น index ที่ชี้ไปยังตารางสี ควรจะระลึกเสมอว่าการเปลี่ยนค่าข้อมูลนั้นแม้เพียง 1 bit อาจหมายถึงการเปลี่ยนจากเฉดสีหนึ่งไปยังอีกเฉดสีหนึ่งเลยทีเดียว ซึ่งการเปลี่ยนแปลงในลักษณะนี้ มนุษย์สามารถรับรู้ถึงความผิดปกติได้อย่างแน่นอน ด้วยเหตุนี้วิธี LSB insertion จึงมักจะไม่นำไปใช้กับภาพที่มีลักษณะเป็นภาพ Grayscale

2.4 ทฤษฎีการมองเห็นของมนุษย์

การมองเห็นของมนุษย์นั้นมีส่วนหลักๆ 2 ส่วนนั่นคือ ส่วนที่ดวงตารับแสง และอีกส่วนคือการส่งไปยังสมองส่วนที่ทำหน้าที่ตีความ โดยส่วนที่เราสนใจศึกษา คือส่วนที่ดวงตารับแสง ซึ่งเริ่มต้นจาก retina ที่อยู่ภายในดวงตานั้นจะประกอบไปด้วยส่วนที่ไวต่อการรับรู้ข้อมูลภาพที่เราเรียกว่า Photo receptors ซึ่งเป็น cell ที่มีอยู่ มากใน retina แบ่งได้ 2 ชนิดคือ rods และ cones โดยส่วนที่เป็น rods นั้นจะไวต่อการรับรู้ความเข้มแสง ทำให้เราแยกความแตกต่างของสีขาว และดำ ในขณะที่ cones นั้นมีความไวต่อความแตกต่างสี ซึ่ง cones มีอยู่ด้วยกัน 3 ชนิด คือ cones ชนิดที่ไวต่อแสงสีแดง,สีเขียว และสีน้ำเงิน ซึ่งจำนวนของ cones ชนิดต่างๆที่เป็นส่วนประกอบของ Photo receptors นั้นมีจำนวนที่ต่างกัน โดย cones ชนิดที่ไวต่อแสงสีเขียวมีอยู่มากที่สุด และ

cones ที่ไวต่อแสงสีน้ำเงินมีจำนวนอยู่น้อยที่สุด จำนวนของ cones ที่แตกต่างกันนี้เองมีผลต่อการมองเห็นของมนุษย์นั่นคือ มนุษย์สามารถรับรู้ความแตกต่างของสีเขียวได้ดีกว่าสีอื่นๆ และรับรู้ความแตกต่างของสีน้ำเงินน้อยกว่าสีอื่นๆ และจากเหตุผลนี้เราจึงใช้ประโยชน์จากข้อมูลภาพ bitmap ที่ใช้ RGB Color space ซึ่ง R,G และ B เป็นข้อมูลที่แสดงระดับความแตกต่างของสีแดง, เขียว และน้ำเงินตามลำดับ โดยการแทรก bit ข้อมูลของรายละเอียดที่เราต้องการใส่ไปในจุดภาพในส่วนที่เป็น B component ของ RGB Color space และเพื่อให้การเปลี่ยนแปลงนั้นมีผลต่อคุณภาพของภาพน้อยที่สุด เราก็จะทำการเปลี่ยนแปลงสีไปจากสีเดิมน้อยที่สุด

บทที่ 3
การวิจัย และการดำเนินการ

3.1 ระบบงานของโปรแกรม

งานวิจัยนี้คือโปรแกรมซ่อนข้อมูลลงในภาพเพื่อช่วยอธิบายรายละเอียดของภาพ ซึ่งลักษณะการทำงานของโปรแกรมจะแบ่งเป็น 2 ส่วนดังนี้

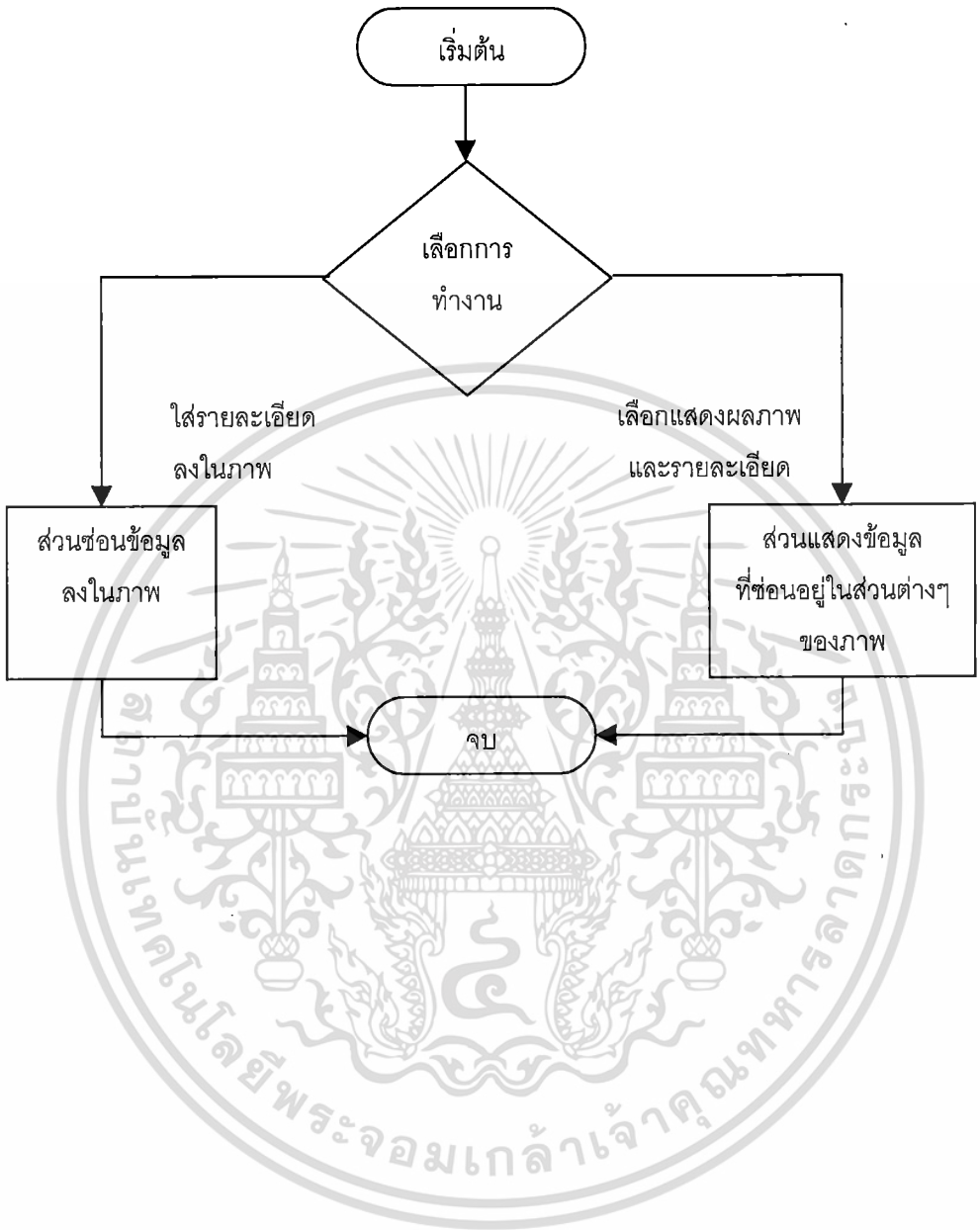
1. ส่วนซ่อนข้อมูลลงในภาพ

ให้ผู้ใช้เลือกภาพที่ต้องการ จากนั้นเลือกส่วนที่ต้องการจะบอกรายละเอียดภายในภาพนั้น หลังจากนั้นก็พิมพ์รายละเอียดที่ต้องการอธิบายภาพในส่วนนั้น โดยถ้าหากผู้ใช้ต้องการจะอธิบายส่วนอื่นๆก็สามารถกลับไปเลือกส่วนที่ต้องการ แล้วพิมพ์รายละเอียดของส่วนนั้นๆ แต่จะต้องไม่ไปซ้ำซ้อนกับส่วนที่ได้ทำการเลือกไว้ก่อนหน้านี้ หลังจากที่ผู้ใช้เสร็จสิ้นขั้นตอนในการเลือกส่วนของภาพ และพิมพ์รายละเอียดของส่วนต่างๆแล้ว โปรแกรมจะทำการบีบอัดข้อมูลที่เป็นรายละเอียดที่ผู้ใช้พิมพ์เพื่อให้สามารถจัดเก็บข้อมูลรายละเอียดได้มากขึ้น จากนั้นก็ทำการจัดเก็บข้อความที่ผ่านการบีบอัดแล้วลงในภาพ ซึ่งจะทำโดยการแทรกลงในจุดสีของภาพ โดยให้ขนาดของไฟล์ภาพไม่เปลี่ยนแปลงแต่จะมีผลกระทบกับคุณภาพของภาพซึ่งเมื่อมองด้วยสายตาของผู้ใช้แล้วอาจไม่เห็นการเปลี่ยนแปลงได้ชัดเจน

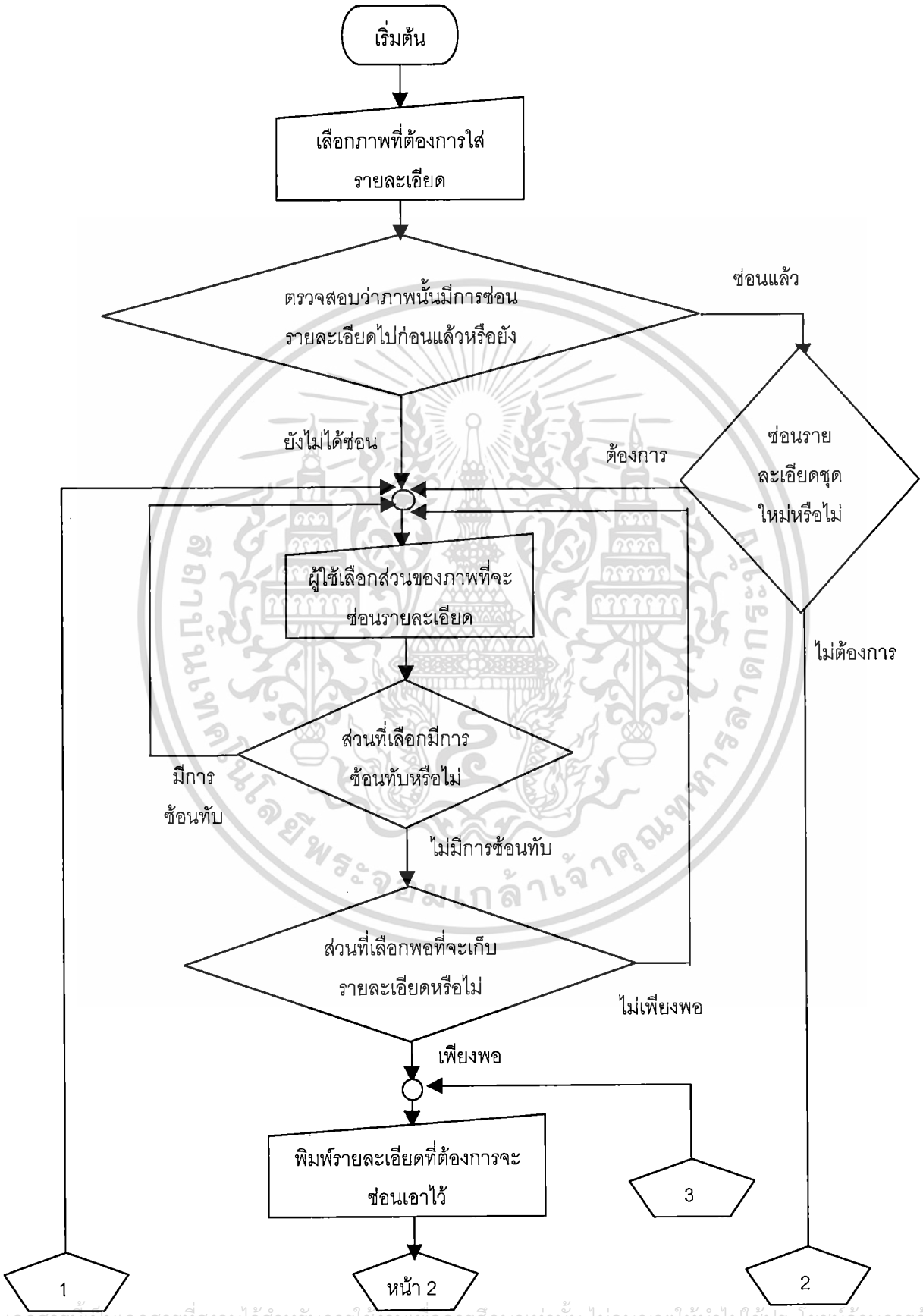
2. ส่วนแสดงข้อมูลที่ซ่อนอยู่ในส่วนต่างๆของภาพ

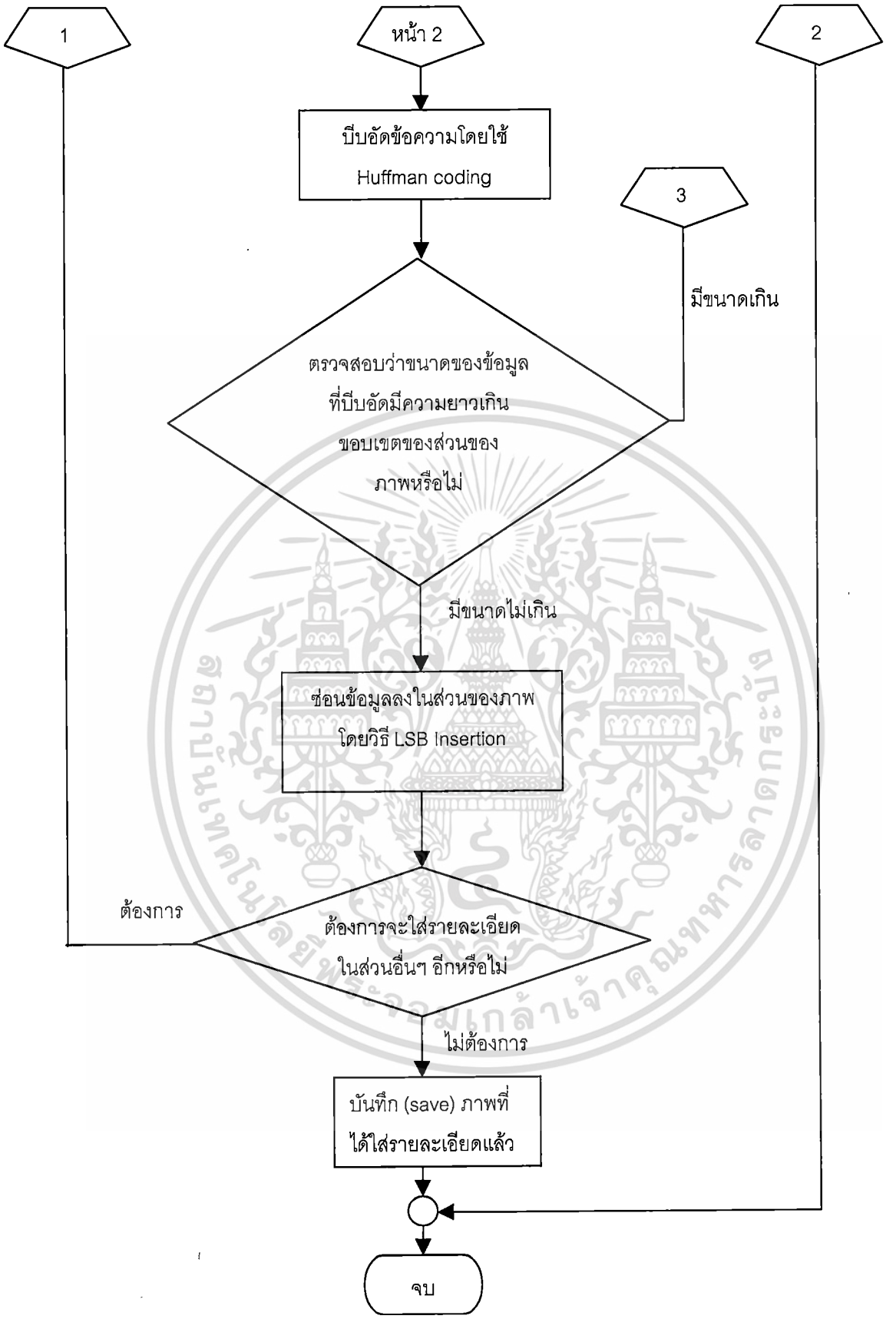
เมื่อผู้ใช้เลือกภาพที่ผ่านการจัดเก็บรายละเอียดไปแล้วนำมาเปิดในโปรแกรม ตัวโปรแกรมจะทำการขยายข้อความที่บีบอัดเอาไว้ และแสดงรายละเอียดของส่วนต่างๆในภาพที่มีการจัดเก็บรายละเอียดไว้นั้น โดยจะแสดงรายละเอียดเฉพาะส่วนที่ผู้ใช้นำเมาส์ไปวาง

ขั้นตอนของความสัมพันธ์ในการทำงานแสดงเป็น System flow chart ดังนี้

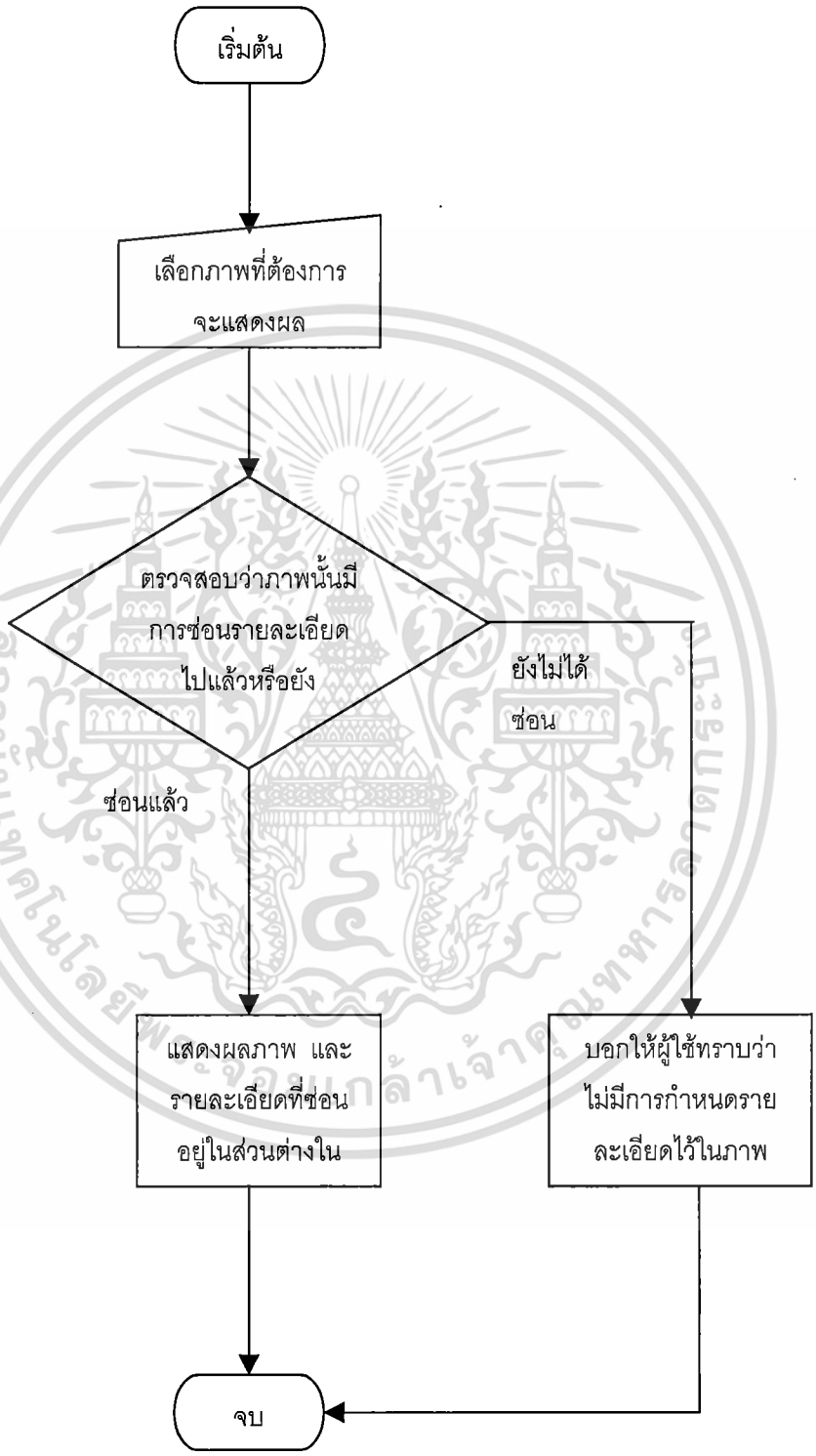


ส่วนซ่อนข้อมูลลงในภาพ





ส่วนแสดงข้อมูลที่ซ่อนอยู่ในส่วนต่าง ๆของภาพ



3.2 รายละเอียดการออกแบบ

ปัญหาพิเศษนี้คือโปรแกรมซ่อนข้อมูลลงในภาพเพื่อช่วยอธิบายรายละเอียดของภาพ ซึ่งลักษณะการทำงานของโปรแกรมจะแบ่งเป็น 2 ส่วนดังนี้

3.2.1 ส่วนซ่อนข้อมูลลงในภาพ

3.2.1.1 การรับภาพ

ภาพที่ต้องการจะซ่อนรายละเอียดลงไป จะต้องนำมาตรวจสอบก่อนว่า ภาพนั้นเป็นภาพ 24-bit Bitmap เท่านั้น และถ้าภาพที่เลือกนั้นมีการแทรกรายละเอียดเอาไว้แล้ว โปรแกรมจะทำการสอบถามผู้ใช้ว่าจะทำการแทรกรายละเอียดลงในภาพนั้นๆอีกหรือไม่ ถ้าผู้ใช้ต้องการแทรกรายละเอียดลงในส่วนของภาพก็จะมีทางเลือกให้ผู้ใช้เลือก 2 ทางเลือก คือ ผู้ใช้ต้องการที่จะแทรกรายละเอียดในภาพนั้นเพิ่มเติมหรือผู้ใช้ต้องการที่จะยกเลิกการแทรกรายละเอียดในภาพนั้นทิ้งไปแล้วทำการแทรกรายละเอียดในภาพนั้นๆใหม่

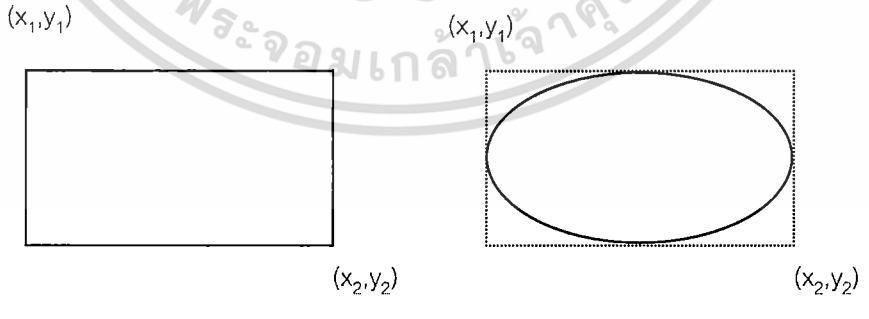
3.2.1.2 การเลือกส่วนของภาพที่ต้องการใส่รายละเอียด

ในการเลือกส่วนของภาพผู้ใช้สามารถเลือกรูปแบบส่วนที่ต้องการได้ 2 รูปแบบคือ

- รูปสี่เหลี่ยม
- รูปวงรีหรือวงกลม

โดยเมื่อผู้ใช้เลือกส่วนของภาพแล้ว โปรแกรมจะตรวจสอบว่าส่วนที่เลือกนั้นมีการซ้อนทับกับส่วนที่ได้ใส่รายละเอียดไปแล้วหรือไม่ โดยมีวิธีการดังนี้

- โปรแกรมจะทำการเก็บจุด (x_1,y_1) และ (x_2,y_2) ดังรูปสำหรับทุกๆส่วนของภาพที่ได้ทำการเลือก



ส่วนของภาพที่เลือกเป็นรูปสี่เหลี่ยม ส่วนของภาพที่เลือกเป็นรูปวงรี

รูปที่ 3.2.1.2.1 แสดงจุดที่โปรแกรมเก็บไว้สำหรับแต่ละส่วนของภาพ

- โปรแกรมจะทำการคำนวณ เพื่อตรวจสอบว่าส่วนของภาพที่เลือกมีการซ้อนทับกับส่วนของภาพที่เลือกไว้แล้วก่อนหน้าหรือไม่ โดยอาศัยความสัมพันธ์จากสมการทางคณิตศาสตร์ของสี่เหลี่ยม และวงรี โดยแยกตรวจสอบเป็น 3 กรณี

กำหนด : มีส่วนของภาพอยู่ 2 ส่วนที่ต้องการตรวจสอบว่าซ้อนทับกันหรือไม่ คือ ส่วนที่ 1 กับ ส่วนที่ 2 โดยให้

x11 แทน จุด x_1 ของส่วนของภาพที่ 1

x12 แทน จุด x_1 ของส่วนของภาพที่ 2

x21 แทน จุด x_2 ของส่วนของภาพที่ 1

x22 แทน จุด x_2 ของส่วนของภาพที่ 2

y11 แทน จุด y_1 ของส่วนของภาพที่ 1

y12 แทน จุด y_1 ของส่วนของภาพที่ 2

y21 แทน จุด y_2 ของส่วนของภาพที่ 1

y22 แทน จุด y_2 ของส่วนของภาพที่ 2

1. สี่เหลี่ยมกับสี่เหลี่ยม

```
if ( (x11 > x22) || (y11 > y22) || (x21 < x12) || (y21 < y12) )
{
   return NotOverlap;
}
else
{
    return Overlap;
}
```

2. สี่เหลี่ยมกับวงรี

ให้ส่วนของภาพที่ 1 เป็นรูปสี่เหลี่ยม และ ส่วนของภาพที่ 2 เป็นรูปวงรี

```
if( (x11 <= x12) && (y11 <= y12) && (x21 >= x22) && (y21 >= y22) )
{
    return Overlap;
}
double   a,b,h,k,x,y;
a = (x22 – x12) / 2;
```

```
b = (y22 – y12) / 2;
h = (x22 + x12) / 2;
k = (y22 + y12) / 2;
x = a^2 * (1 - ((y11 - k)^2 /  b^2));
if (x >= 0)
{
    double x1,x2;
    x1 = h - sqrt(x);
    x2 = sqrt(x) + h;
    if( (x1 >= x11) && (x1 <= x21) )
    {
            return Overlap;
    }
    else if( (x2 >= x11) && (x2 <= x21) )
    {
            return Overlap;
    }
    else if ( (x1 <= x11)&&(x2 >= x21))
    {
            return Overlap;
    }
}
x = a^2 * (1 - ((y21 - k)^2 /  b^2));
if (x >= 0)
{
    double x1,x2;
    x1=h-sqrt(x);
    x2=sqrt(x)+h;
    if( (x1 >= x11) && (x1 <= x21) )
    {
            return Overlap;
```

```
        }
        else if( (x2 >= x11) && (x2 <= x21))
        {
                return Overlap;
        }
        else if ((x1<=beginPoint.x)&&(x2>=stopPoint.x))
        {
                return Overlap;
        }
}
y =  b^2 * (1 - ((x11 - h)^2 /  a^2));
if (y >= 0)
{
        double y1,y2;
        y1 = k - sqrt(y);
        y2 = sqrt(y) + k;
        if( (y1 >= y11) && (y1 <= y21) )
        {
                return Overlap;
        }
        else if( (y2 >= y11) && (y2 <= y21) )
        {
                return Overlap;
        }
        else if ( (y1 <= y11) && (y2 >= y21))
        {
                return Overlap;
        }
}
y =  b^2 * (1 - ((x21 - h)^2 /  a^2));
if (y >= 0)
```

```
{
    double y1,y2;
    y1=k-sqrt(y);
    y2=sqrt(y)+k;

    if ( (y1 >= y11) && (y1 <= y21) )
    {
            return Overlap;
    }
    else if ( (y2 >= y11) && (y2 <= y21) )
    {
            return Overlap;
    }
    else if ( (y1 <= y11) && (y2 >= y21) )
    {
            return Overlap;
    }
}
retrun NotOverlap;
```

3. วงรีกับวงรี

```
double a1,b1,h1,k1,a2,b2,h2,k2;
a1 = (x21 - x11) / 2;
b1 = (y21 - y11) / 2;
h1 = (x21 + x11) / 2;
k1 = (y21 + y11) / 2;
a2 = (x22 - x12) / 2;
b2 = (y22 - y12) / 2;
h2 = (x22 + x12) / 2;
k2 = (y22 + y12) / 2;
if( (x11 <= x12) && (y11 <= y12)&&(x21 >= x22) && (y21 >= y22) )
```

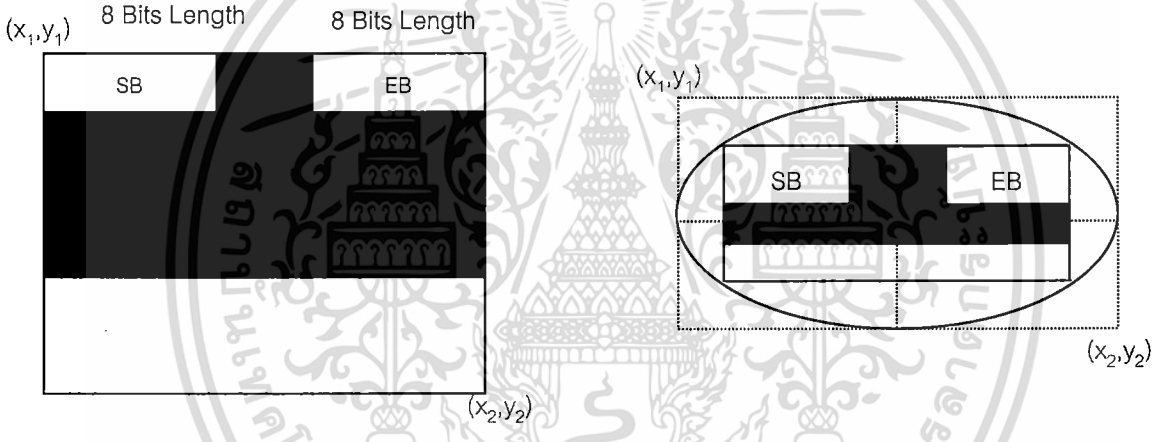
```
{
        return Overlap;
}
int startX,endX;
BOOL flag=FALSE;
If ( (x11 >= x12) && (x11 <= x22) )
{
        flag = TRUE;
        startX = x11;
        if (x21 <= x22)
        {
                endX = x21;
        }
        else
        {
                endX = x22;
}
else if( (x12 >= x11) && (x12 <= x21) )
{
        flag = TRUE;
        startX = x12;
        if (x21 <= x22)
        {
                endX = x21;
        }
        else
        {
                endX = x22;
        }
}
if(flag)
```

```
{
        int i = startX;
        while( (I <= endX) && (!result) )
        {
                double y,y11,y12,y21,y22;
                y =  b1^2 * (1 - (( i - h1)^2 /  (a1)^2));
                y11 =k1 - sqrt(y);
                y12 = sqrt(y) + k1;
                y =  (b2)^2 * (1 - ((I - h2)^2 /  (a2)^2));
                y21 = k2 - sqrt(y);
                y22 = sqrt(y) + k2;
                if ( (y11 >= y21) && (y11 <= y22) )
                {
                        return Overlap;
                }
                else if( (y12 >= y21) && (y12 <= y22))
                {
                        return Overlap;
                }
                i++;
        }
}
return NotOverlap;
```

3.2.1.3 การตรวจสอบว่าส่วนของภาพที่เลือกมีเนื้อที่เพียงพอหรือไม่ที่จะเก็บข้อมูลเพื่อการตรวจสอบว่าส่วนของภาพนี้มีการแทรกรายละเอียดเอาไว้แล้ว

ข้อมูลเพื่อการตรวจสอบว่าส่วนของภาพนี้จะมีการแทรกรายละเอียดเอาไว้ ก็คือรายละเอียดที่จะบอกได้ว่ามีส่วนของภาพที่ได้ซ่อนข้อความเอาไว้ตรงส่วนไหนบ้าง โดยข้อมูลที่เก็บจะประกอบด้วย จุด x_1, y_1, x_2, y_2 ซึ่งแสดงเอาไว้ดังรูปที่ 3.2.1.2.1 นอกจากนี้ยังต้องเก็บด้วยว่าส่วนของภาพที่เลือกนั้นเป็นรูปอะไร (สี่เหลี่ยมหรือวงรี), ใช้ Huffman Model ใดในการบีบอัดข้อความและข้อความที่บีบอัดมีขนาดกี่ Byte อีกด้วย โดยข้อมูลเหล่านี้จะถูกจัดเก็บอยู่ใน Red Channel ของข้อมูลภาพ

ในการเก็บตำแหน่งเริ่มต้นของส่วนของภาพที่จะใช้เก็บข้อมูลดังที่กล่าวไปข้างต้นจะต้องอาศัยข้อมูลที่เป็น Start of Block (11111110) และ End of Block (11111100) โดยมี รายละเอียดดังรูป



ส่วนของภาพที่เลือกเป็นรูปสี่เหลี่ยม ส่วนของภาพที่เลือกเป็นรูปวงรี

รูปที่ 3.2.1.3.1 **แสดงส่วนที่ใช้เก็บข้อมูลเพื่อการตรวจสอบว่าส่วนของภาพนี้มีการแทรกรายละเอียดเอาไว้**

จากรูป

- ในกรณีที่เป็นสี่เหลี่ยมเราจะเริ่มเก็บ SB ในตำแหน่งเริ่มต้น (x_1,y_1)
- ในกรณีที่เป็นวงรีเราจะเริ่มเก็บ SB ที่ตำแหน่ง
  - วงรีที่มีแกนเอกอยู่บนแกน X

    $x_1 = (h - c)$

    $y_1 = (k - (b^2 / a))$

    โดย $c = sqrt(a^2 - b^2)$
  - วงรีที่มีแกนเอกอยู่บนแกน X

$x_1 = (h - (a^2 / b))$

$y_1 = (k - c)$

โดย $c = sqrt(b^2 - a^2)$

โดย h คือจุดศูนย์กลางวงรีในแนวแกน X

k คือจุดศูนย์กลางวงรีในแนวแกน Y

a คือความยาววงรีในแนวแกน X ที่ผ่านจุดศูนย์กลาง /2

b คือความยาววงรีในแนวแกน Y ที่ผ่านจุดศูนย์กลาง /2

นอกจากนี้ ส่วนที่ใช้ในการเก็บข้อมูล เพื่อการตรวจสอบว่าส่วนของภาพนี้ได้มีการแทรกรายละเอียดเอาไว้แล้ว จะต้องมีขนาดตามแกนนอน (แกน X) อย่างน้อย 16 Pixel (Size of SB + Size of EB) และต้องมีจำนวน Pixel ที่เป็นพื้นที่สีดำอย่างน้อยเท่ากับ RealBitUsed (x_1, y_1, x_2, y_2) โดยFunction RealBitUsed เป็นดังนี้

```
int    RealBitUsed ( int x1, int y1, int x2, int y2 )
{
        int    realBitUsed = 87;       // เป็นขนาดต่ำสุดที่สามารถเก็บรายละเอียดได้
// Size of x1=16 Bit, Size of x2=16 Bit, Size of y1=16 Bit, Size of y2=16 Bit, Size
// of Huffman Model= 2 Bit,
// Size of Compress Data = 16 Bit + 4 Bit for Stuffing in Size of Compress Data
// and Size of draw typeDraw=1Bit
        int    countOne = 0,i;
        for ( i=1 ; i <= 16 ; i++)
        {
                if ( ( x1 % 2) == 1 )
                        countOne++;
                else
                        countOne = 0;
                x1 = x1 >> 1;
                if ( countOne == 5 )            // ต้อง set Bit Stuffing
                {
                        realBitUsed++;
                        countOne = 0;
```

```
        }
}
for ( i=1 ; i <= 16 ; i++ )
{
        if ( ( y1 % 2) == 1 )
                countOne++;
        else
                countOne = 0;
        y1 = y1 >> 1;
        if ( countOne == 5 )            // ต้อง set Bit Stuffing
        {
                realBitUsed++;
                countOne = 0;
        }
}
for( i=1 ; i <= 16 ; i++)
{
        if ( ( x2 % 2) == 1 )
                countOne++;
        else
                countOne = 0;
        x2 = x2 >> 1;
        if ( countOne == 5 )            // ต้อง set Bit Stuffing
        {
                realBitUsed++;
                countOne = 0;
        }
}
for( i=1 ;  i <= 16 ; i++)
{
        if ( ( y2 % 2 ) == 1 )
```

```
                countOne++;
            else
                countOne = 0;
            y2 = y2 >> 1;
            if ( countOne == 5 )            // ต้อง set Bit Stuffing
            {
                realBitUsed++;
                countOne = 0;
            }
        }
        return realBitUsed;
    }
```

3.2.1.4 **การบีบอัดข้อความโดยวิธี** Huffman Coding

โปรแกรมจะทำการรับข้อความจากผู้ใช้ จากนั้นจะนำข้อความดังกล่าวมาทำการบีบอัดโดยวิธี Huffman Coding โดยมีรายละเอียดของ model ที่ใช้ดังนี้

โปรแกรมได้สร้างตัวแบบ (Huffman Model) ขึ้นมา 3 ตัวแบบ คือ

1. ตัวแบบสำหรับภาษาไทย : ซึ่งประกอบด้วย พยัญชนะไทย , < >,<.>,<:>, <?>,<'>,<+>,<->,<*>,<=>,<!>, <$>,</>,<\>,<฿>,<%>,<&>,<(>,<)>,<">,<,>,<{>,<}>, <<>,<>>,<;>,<_>,<[>,<]>,<~>,<@>,<#>,<^>,<|>, 0 to 9 , ๐ - ๙

2. ตัวแบบสำหรับภาษาอังกฤษ : ซึ่งประกอบด้วย A to Z, a to z , < >,<.>, <:>,<?>,<'>,<+>,<->,<*>,<=>,<!>, <$>,</>,<\>,<%>,<&>,<(>,<)>,<">,<,>,<{>,<}>, <<>,<>>,<;>,<_>,<[>,<]>,<~>,<@>,<#>,<^><|>, 0 to 9

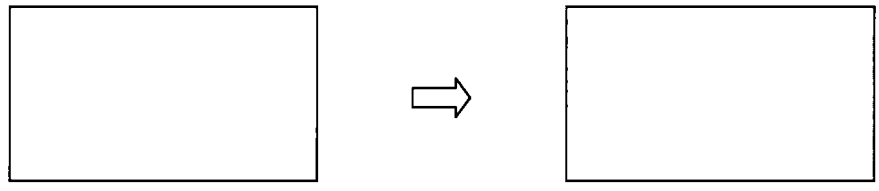
3. ตัวแบบสำหรับภาษาไทยและภาษาอังกฤษ : จะประกอบด้วยตัวอักขระในตัวแบบสำหรับภาษาไทยและตัวแบบสำหรับภาษาอังกฤษทั้งหมด

สำหรับรายละเอียดของวิธี Huffman coding ได้อธิบายไปแล้วในหัวข้อ 2.2.3

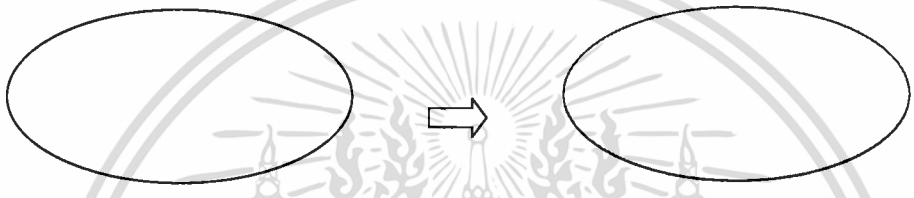
3.2.1.5 การตรวจสอบว่าส่วนของภาพที่เลือกนี้มีเนื้อที่เพียงพอที่จะเก็บข้อความของส่วนนั้น ๆหรือไม่

โปรแกรมจะนำข้อความที่ผ่านการบีบอัดแล้ว มาตรวจสอบว่าสามารถใส่ลงในส่วนของภาพที่เลือกมาหรือไม่ ดังนี้

- ทุกๆครั้งที่มีการเลือกส่วนของภาพ โปรแกรมจะทำการคำนวณจุดบนเส้นรอบ-รูปที่เลือกและทำการเก็บเอาไว้ดังรูป โดยโปรแกรมจะเก็บเฉพาะจุดบนเส้นสีแดง



ส่วนของภาพที่เลือกเป็นรูปสี่เหลี่ยม



ส่วนของภาพที่เลือกเป็นรูปวงรี

รูปที่ 3.2.1.5.1 **แสดงจุดของส่วนของภาพที่โปรแกรมเก็บเอาไว้**

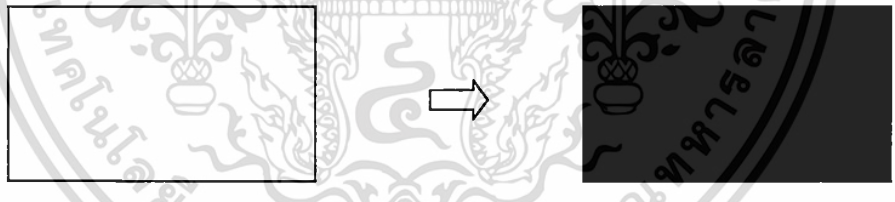
- โปรแกรมจะทำการนำจุดต่างๆมาคำนวณว่ามีเนื้อที่สำหรับใส่ข้อความที่บีบอัดทั้งหมดกี่ Pixel หลังจากนั้นจะนำมาเปรียบเทียบกับขนาดของข้อมูลที่บีบอัดแล้ว ดังนี้

```
CPoint    start,stop;
int    bitUsed = ( compress->GetCount() ) *  8; // ขนาดข้อมูลที่ทำการบีบอัด
int    I = 0;
int    sizeArray = ( m_pTempDraw->m_Points ).GetSize();
//  m_pTempDraw->m_Points  เก็บจุดในส่วนของภาพตามที่กล่าวไว้ในหัวข้อนี้
int    bitBlockHave = 0;                    // จำนวน pixel ที่มีในส่วนของภาพ
while ( i < sizeArray )
{
        start = ( m_pTempDraw->m_Points ).GetAt( i );
        stop = ( m_pTempDraw->m_Points ).GetAt( ++i );
        int    y = start.y;
        for ( int x = start.x ; x <= stop.x ; x++)
        {
```

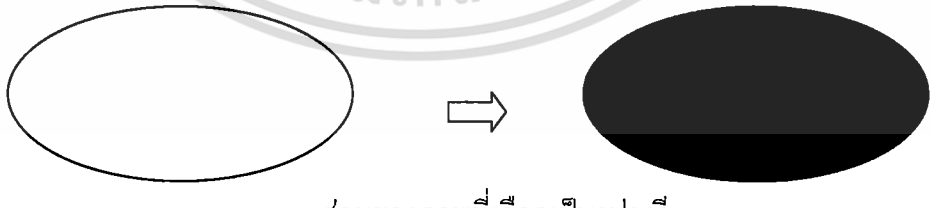
```
                bitBlockHave++;
            }
            i++;
        }
        if ( bitBlockHave >= bitUsed )
        {
            return Enough;
        }
        return NotEnough;
```

3.2.1.6 การเก็บข้อมูลที่บีบอัดลงในจุดสีของภาพ

หลังจากที่เราได้ทำการตรวจสอบแล้วว่า ขนาดของข้อมูลที่ผ่านการบีบอัดไปแล้วนั้น สามารถแทรกลงในจุดสีของส่วนที่เลือกแล้ว โปรแกรมจะทำการนำข้อมูลแทรกลงในจุดสีโดยแทรกลงใน Blue Channel ตามหลักการที่ได้อธิบายไว้ในหัวข้อ 2.3.4.1 Least Significant Bit (LSB) insertion โดยจุดที่จะทำการเก็บข้อมูลที่บีบอัดจะเป็นจุดที่อยู่ในส่วนของภาพที่ผู้ใช้เลือกดังรูป ซึ่งก็คือจุดในบริเวณพื้นที่สีน้ำเงินทั้งหมด โดยข้อมูลที่บีบอัดเหล่านี้จะถูกจัดเก็บอยู่ใน Blue Channel ของข้อมูลภาพ



ส่วนของภาพที่เลือกเป็นรูปสี่เหลี่ยม



ส่วนของภาพที่เลือกเป็นรูปวงรี

รูปที่ 3.2.1.6.1 **แสดงจุดของส่วนของภาพที่สามารถเก็บข้อมูลที่บีบอัดได้**

3.2.2 ส่วนแสดงข้อมูลที่ซ่อนอยู่ในส่วนต่าง ๆของภาพ

3.2.2.1 เลือกภาพที่ต้องการจะแสดงผล

ภาพที่ต้องการจะแสดงรายละเอียดที่ได้ซ่อนเอาไว้นั้น จะต้องถูกตรวจสอบก่อนว่าเป็นภาพ 24-bit Bitmap

3.2.2.2 ตรวจสอบว่าภาพนั้นมีการซ่อนรายละเอียดไปแล้วหรือยัง

ในการตรวจสอบว่าภาพนั้นมีการซ่อนรายละเอียดไปแล้วหรือยัง โปรแกรมจะอ่านข้อมูลภาพไปเรื่อยๆจนเจอ Start of Block (11111110) และ End of Block (11111100) จากนั้นโปรแกรมก็จะอ่านรายละเอียดในส่วนของภาพนั้นออกมา ซึ่งก็คือข้อมูลเพื่อการตรวจสอบว่าส่วนของภาพนี้มีการแทรกรายละเอียดเอาไว้แล้วซึ่งอธิบายเอาไว้ในหัวข้อ 3.2.1.3 (ประกอบด้วยค่า x_1, y_1, x_2, y_2, ค่าที่บอกว่าส่วนการใส่รายละเอียดนั้นเป็นส่วนที่มีรูปร่างเป็นสี่เหลี่ยมหรือวงรี, Huffman Model ที่ใช้ในการบีบอัดข้อมูล และขนาดของข้อมูลที่บีบอัด)

3.2.2.3 นำส่วนของภาพที่ได้ซ่อนคำอธิบายเอาไว้มาแสดงรายละเอียด

โปรแกรมจะนำค่า x_1, y_1, x_2, y_2 และข้อมูลที่บอกว่าส่วนของภาพมีรูปร่างเป็นรูปอะไรมาทำการคำนวณเพื่อทราบว่าส่วนของภาพนั้นประกอบด้วยจุดใดบนภาพบ้าง จากนั้นจะเริ่มอ่านข้อมูลที่ส่วนของภาพจนได้ข้อมูลครบตามที่ระบุไว้ในค่าขนาดของข้อมูลที่ทำการบีบอัดที่ได้จากหัวข้อ 3.2.2.2 และเมื่อได้ข้อมูลที่บีบอัดมา ก็จะใช้ค่า Huffman Model ที่ได้จากหัวข้อ 3.2.2.2 เช่นกัน เพื่อนำมาทำการขยายข้อความที่บีบอัดออกมา และเก็บเอาไว้เพื่อนำมาแสดงผลต่อไป

3.2.2.4 แสดงผลภาพ และรายละเอียดที่ซ่อนอยู่ในส่วนต่าง ๆของภาพนั้น

หลังจากที่โปรแกรมนำข้อมูลมาขยายการบีบอัดแล้ว โปรแกรมก็จะทำการแสดงรายละเอียดที่ได้ซ่อนเอาไว้ในส่วนของภาพ โดยการแสดงรายละเอียดนี้จะเกิดขึ้นเมื่อผู้ใช้นำเมาส์ไป over หรือทำให้เกิด event mouse over บนส่วนของภาพต่างๆที่ได้ทำการซ่อนข้อความเอาไว้ และโปรแกรมจะแสดงรายละเอียดโดยขึ้นเป็น Tooltip ให้ผู้ใช้ได้ทราบต่อไป

3.2.2.5 ตรวจสอบส่วนของภาพที่มีการใส่รายละเอียดลงไปในกรณีที่มีการ CROP ภาพ

ในกรณีที่มีการ Crop ภาพ แล้วบันทึกเป็นภาพแบบ 24 - bit Bitmap เมื่อนำมาเปิดเพื่อดูรายละเอียดนั้น โปรแกรมจะทำตามขั้นตอนในหัวข้อ 3.2.2.1 ถึงหัวข้อ 3.2.2.2 เพียง

แต่ว่าในกรณีที่เมื่ออ่านค่า x_1, y_1, x_2 และ y_2 ได้มานั้นจะพบว่าไม่สัมพันธ์กับตำแหน่งที่อ่านรายละเอียดเหล่านั้นมา ดังนั้นเราจะต้องหาก่อนว่าค่า x_1, y_1, x_2 และ y_2 ที่เป็นค่าจริงๆหลังจากที่มีการ Crop ภาพ เป็นค่าอะไร โดยอาศัยความสัมพันธ์ที่ว่า

xStart คือ ตำแหน่ง X เริ่มต้นในภาพ ที่พบ SB

yStart คือ ตำแหน่ง y เริ่มต้นในภาพ ที่พบ SB

x1 คือ ค่า x_1 ที่อ่านได้จากภาพ

y1 คือ ค่า y_1 ที่อ่านได้จากภาพ

x2 คือ ค่า x_2 ที่อ่านได้จากภาพ

y2 คือ ค่า y_2 ที่อ่านได้จากภาพ

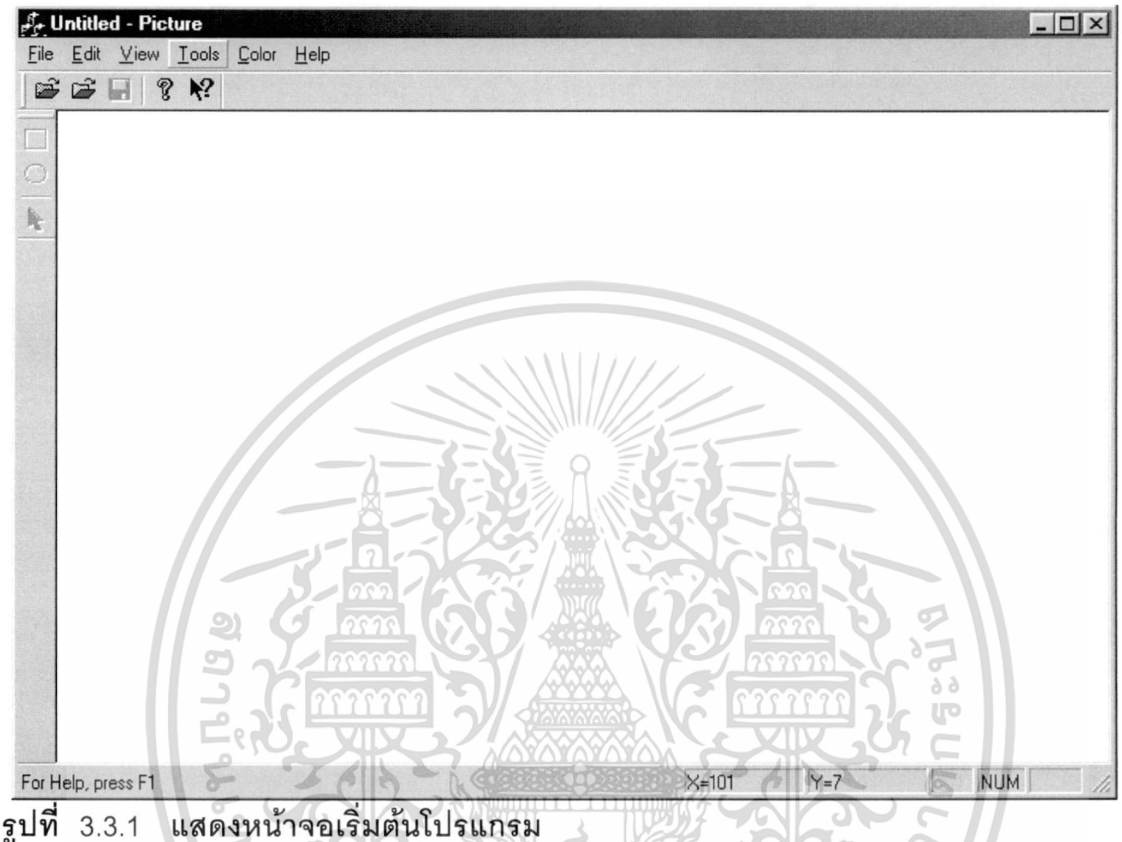
```
switch ( typeDraw )     //    typeDraw ตัวแปรบอกว่าเป็นรูปสี่เหลี่ยมหรือวงรี
{
        case 0:         //Rectangle
                if ( ( xStart != x1) || (yStart != y1) )
                {
                        int   deltaX = x1 - xStart;
                        int   deltaY = y1 - yStart;
                        x1 = x1 - deltaX;
                        y1 = y1 - deltaY;
                        x2 = x2 - deltaX;
                        y2 = y2 - deltaY;
                }
                break;
        case 1:         //Ellipse
                double    a,b,h,k,c;
                int    xC,yC;
                a = ( x2 - x1 ) / 2;
                b = ( y2 - y1 ) / 2;
                h = ( x1 + x2 ) / 2;
                k = ( y1 + y2 ) / 2;
                if ( a >= b )
                {       c = sqrt( pow(a,2) - pow(b,2) );
```

```
                    xC = ( int ) ( h – c );
                    yC = ( int ) ( k - ( pow(b,2) / a ) );
                    if ( ( xStart != xC ) || ( yStart != yC ) )
                    {
                        int   deltaX = xC - xStart;
                        int   deltaY = yC - yStart;
                        x1 = x1 - deltaX;
                        y1 = y1 - deltaY;
                        x2 = x2 - deltaX;
                        y2 = y2 - deltaY;
                    }
                }
                else
                {       c = sqrt( pow(b,2) – pow(a,2) );
                    xC = ( int ) ( h - ( pow(a,2) / b ) );
                    yC = ( int ) ( k – c );
                    if ( ( xStart != xC ) || ( yStart != yC ) )
                    {
                        int   deltaX = xC - xStart;
                        int   deltaY = yC - yStart;
                        x1 = x1 - deltaX;
                        y1 = y1 - deltaY;
                        x2 = x2 - deltaX;
                        y2 = y2 - deltaY;
                    }
                }
            break;
        }
```

เมื่อได้ค่า x_1, y_1, x_2 และ y_2 จากความสัมพันธ์ข้างต้นมาแล้ว ก็ดำเนินการตามขั้นตอนในหัวข้อ 3.2.2.3 และ 3.2.2.4 ต่อไป

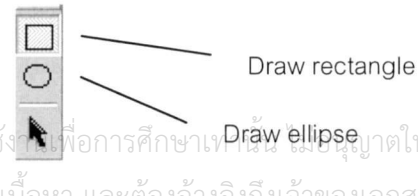
3.3 ลักษณะของการใช้โปรแกรม

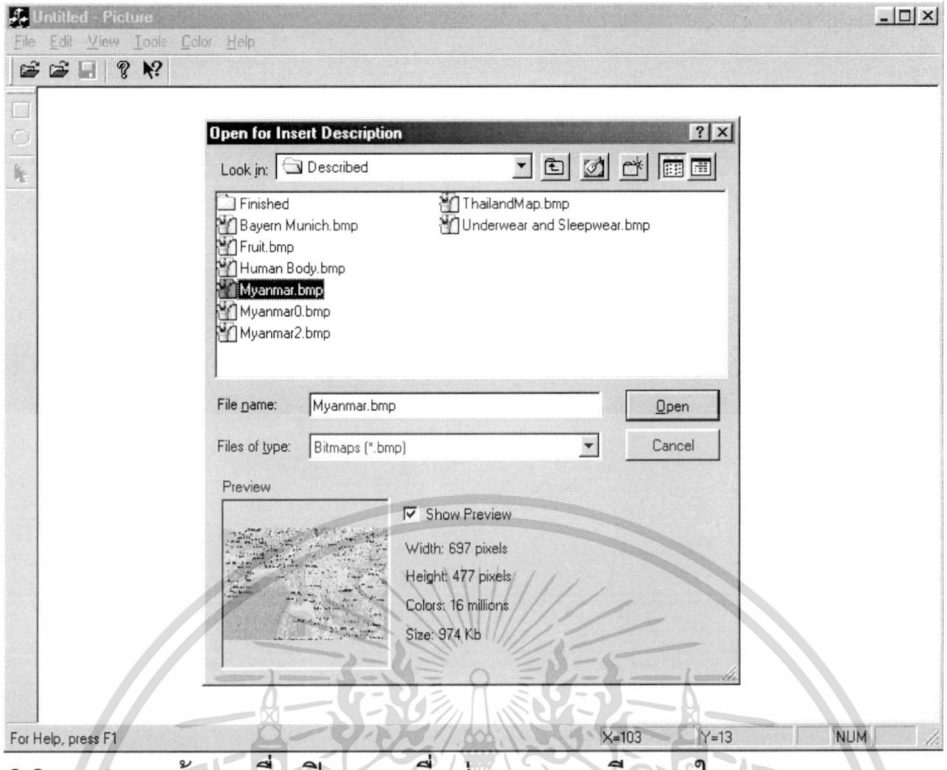
โปรแกรมซ่อนข้อมูลลงในภาพเพื่อช่วยอธิบายรายละเอียดของภาพ เป็นแอปพลิเคชั่นที่ทำงานบนระบบปฏิบัติการ Windows ซึ่งเมื่อทำการติดตั้งแล้วเรียกใช้งานจะปรากฏหน้าจอดังรูป



รูปที่ 3.3.1 **แสดงหน้าจอเริ่มต้นโปรแกรม**

ต่อมาให้เลือกไปที่เมนู "File" เพื่อทำการเปิดภาพขึ้นมา โดยการเปิดภาพนั้นต้องเลือกการเปิดภาพเพื่อซ่อนรายละเอียดลงในภาพ โดยไปที่คำสั่ง "Open for Insert Description..." (หรืออาจ click ที่รูป  ที่อยู่บริเวณ toolbar ทางด้านบน) จากนั้นจะปรากฏหน้าจอดังรูปที่ 3.3.2 โดยจะมีการแสดงตัวอย่าง (preview) ภาพที่เลือกอยู่ (สังเกตว่าในขณะที่ทำการ preview นั้น ภาพ bitmap ที่มีรายละเอียดของ colors ไม่เป็นแบบ 16 millions จะไม่สามารเปิดเข้ามาใช้ในโปรแกรมได้) จากนั้นเมื่อได้ภาพที่ต้องการแล้ว click ที่ปุ่ม open จะปรากฏภาพนั้นในโปรแกรมดังรูปที่ 3.3.3 เมื่อเปิดภาพเข้ามาในโปรแกรมแล้ว จากนั้นก็เป็นขั้นตอนของการซ่อนรายละเอียดใน ส่วนต่างๆของภาพซึ่งการเลือกส่วนต่างๆของภาพนั้นมีเครื่องมือที่ใช้ในการเลือกส่วนของภาพอยู่ 2 แบบคือ เครื่องมือรูปสี่เหลี่ยม (Draw rectangle) และเครื่องมือรูปวงรี (Draw ellipse) ซึ่งจะปรากฏอยู่ใน toolbar ทางด้านซ้าย





รูปที่ 3.3.2 **แสดงหน้าจอเพื่อเปิดภาพเพื่อซ่อนรายละเอียดลงในภาพ**

รูปที่ 3.3.3 **แสดงหน้าจอเมื่อทำการเปิดภาพเข้ามาในโปรแกรม (สังเกตว่าจะมีคำว่า <insert>> ปรากฏอยู่ที่ Title bar เพื่อแสดงว่าเป็นการเปิดภาพเพื่อซ่อนรายละเอียดลงในภาพ)**

โดยในที่นี้จะแสดงการเลือกส่วนของภาพโดยใช้เครื่องมือทั้ง 2 แบบคือ Draw rectangle และ Draw ellipse

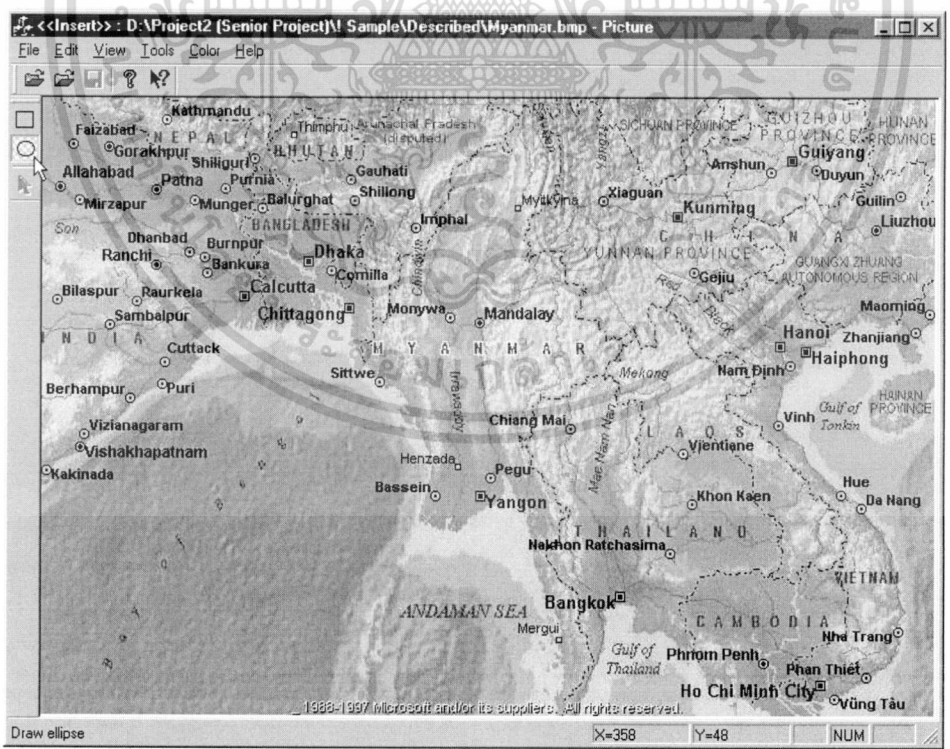
สมมติว่าเราต้องการจะซ่อนข้อมูลลงในส่วนต่างๆ ดังนี้

- บริเวณที่เป็น ANDAMAN SEA ต้องการใส่คำอธิบายว่า “ทะเลอันดามัน : เป็นส่วนหนึ่งของมหาสมุทรอินเดีย ซึ่งแยกออกมาจากอ่าวเบงกอลที่อยู่ทางตอนใต้ของประเทศอินเดีย” (จะเลือกส่วนของภาพโดยใช้ Draw ellipse)

- บริเวณ MYANMAR ต้องการใส่คำอธิบายว่า ” สหภาพพม่า : มีจำนวนประชากรประมาณ 48 ล้านคน ประกอบไปด้วยชนพื้นเมืองกว่า 130 กลุ่ม ที่มีภาษาและขนบธรรมเนียมเป็นของตน โดยคำว่า ” เมียนม่าร์ นั้นหมายความถึงชนชาติกลุ่มใหญ่ๆคือ บาร์ม่าร์ , ชิน , คาร์ชิน , คาร์ยา , คาร์ยิน , หม่อน , ราไคน์ และ ชาน ซึ่งบาร์มาร์มีถึง 69 เปอร์เซ็นต์ของประชากรทั้งประเทศ” (จะเลือกส่วนของภาพโดยใช้ Draw rectangle)

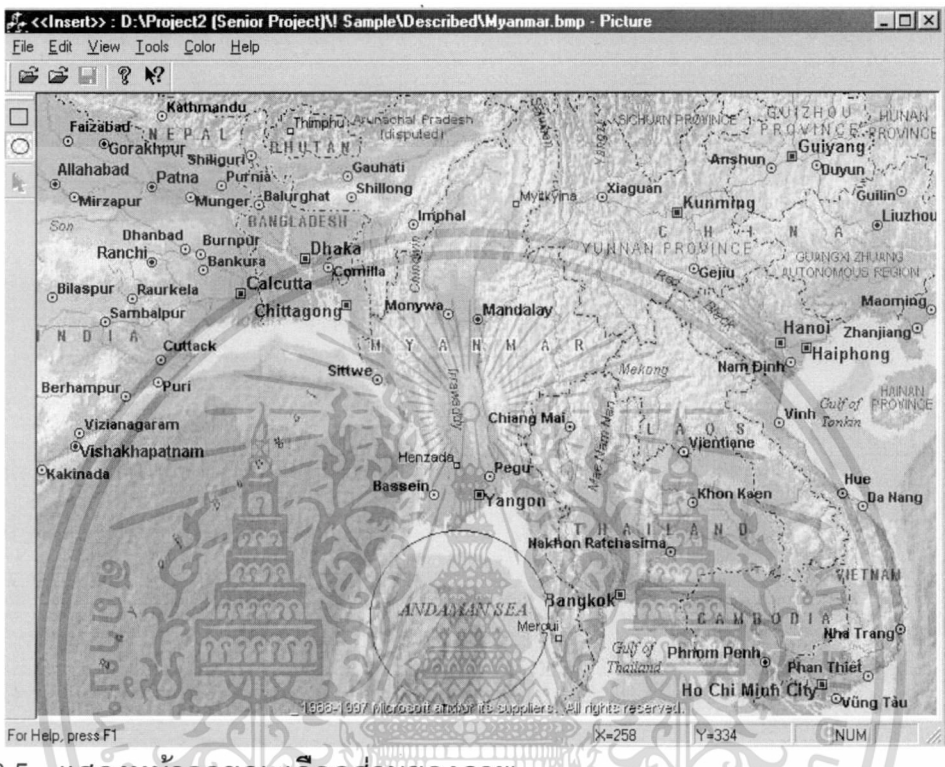
- บริเวณ Yangon ต้องการใส่คำอธิบายว่า “กรุงย่างกุ้ง : เป็นเมืองหลวงของสหภาพพม่า มีประชากรประมาณ 2 ล้านคน” (จะเลือกส่วนของภาพโดยใช้ Draw ellipse)

เราจะเริ่มจากการเลือกส่วนของภาพส่วนที่เป็น ANDAMAN SEA โดยการไปที่ toolbar ที่อยู่ทางด้านซ้าย click ที่เครื่องมือ Draw ellipse ดังรูปที่ 3.3.4



รูปที่ 3.3.4 แสดงหน้าจอขณะเลือกเครื่องมือ Draw ellipse

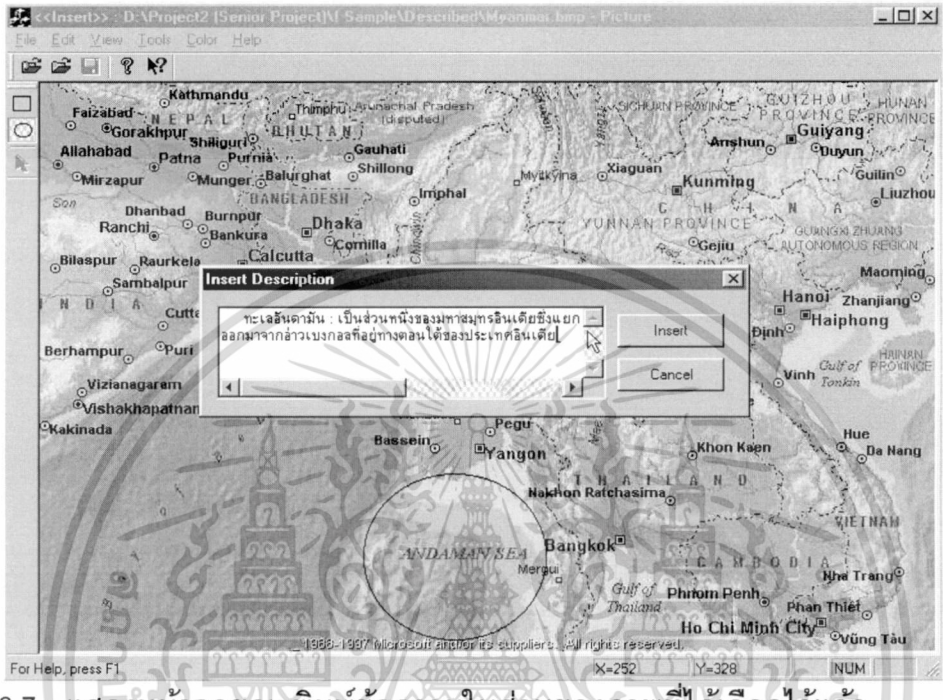
จากนั้นให้เลือกส่วนของภาพส่วนที่เป็น ANDAMAN SEA โดยการ click ที่ mouse ค้างเอาไว้(Drag) และลากให้เป็นรูปวงรีล้อมส่วน ANDAMAN SEA จากนั้นปล่อย mouse ก็จะปรากฏวงรีล้อมรอบส่วนที่เป็น ANDAMAN SEA ดังรูปที่ 3.3.5 และโปรแกรมจะแสดงหน้าจอดังรูปที่ 3.3.6 เพื่อให้ใส่ข้อความในส่วนที่เลือก



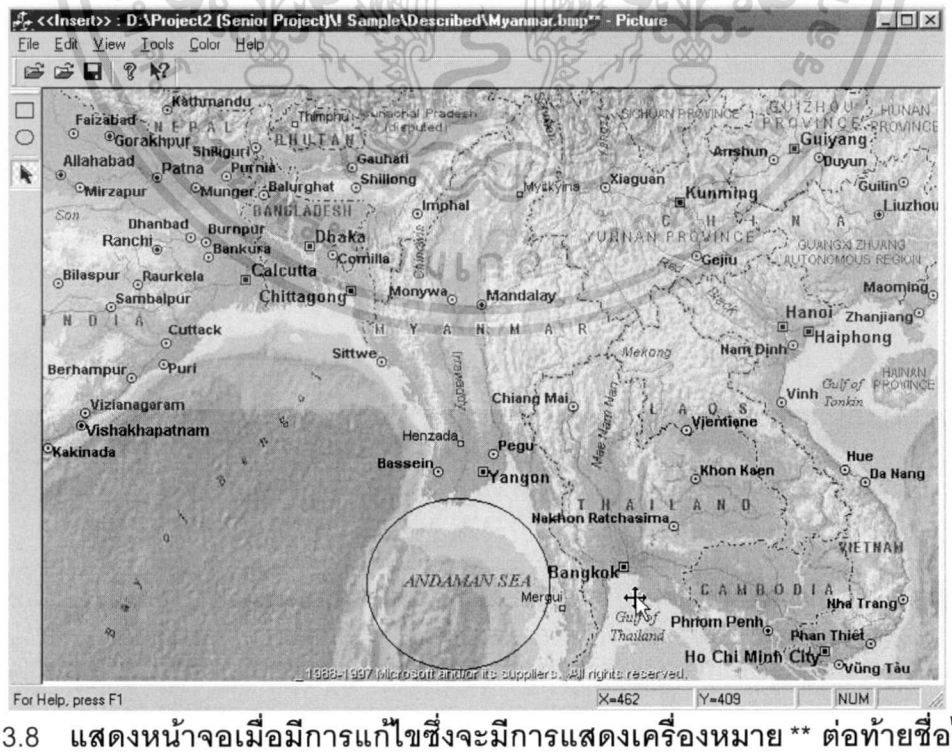
รูปที่ 3.3.5 **แสดงหน้าจอขณะเลือกส่วนของภาพ**

รูปที่ 3.3.6 **แสดงหน้าจอให้ใส่ข้อความที่ต้องการจะซ่อนลงในภาพ**

ในที่นี้ให้ใส่คำอธิบายว่า “ทะเลอันดามัน : เป็นส่วนหนึ่งของมหาสมุทรอินเดียซึ่งแยกออกมาจากอ่าวเบงกอลที่อยู่ทางตอนใต้ของประเทศอินเดีย” ดังรูปที่ 3.3.7 และ click ที่ Insert ก็เป็นอันเสร็จสิ้นการซ่อนข้อความในส่วนนั้น สังเกตที่ Title bar จะพบว่ามีเครื่องหมาย ** ต่อจากชื่อภาพเพื่อแสดงว่ามีการแก้ไข รูปที่ 3.3.8



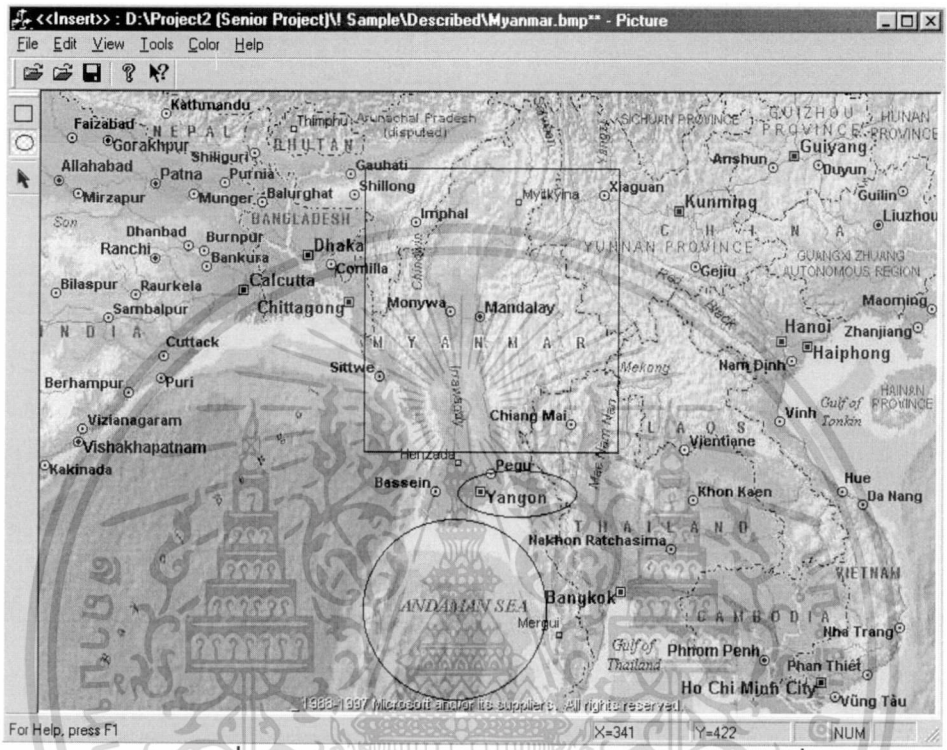
รูปที่ 3.3.7 แสดงหน้าจอขณะพิมพ์ข้อความในส่วนของภาพที่ได้เลือกไว้แล้ว



รูปที่ 3.3.8 แสดงหน้าจอเมื่อมีการแก้ไขซึ่งจะมีการแสดงเครื่องหมาย ** ต่อท้ายชื่อไฟล์

สำหรับส่วนอื่นๆนั้น ก็ให้ทำแบบเดียวกันนั่นคือเลือกส่วนของภาพโดยใช้เครื่องมือ Draw rectangle หรือ Draw ellipse จากนั้นก็เลือกส่วนของภาพ และพิมพ์ข้อความ

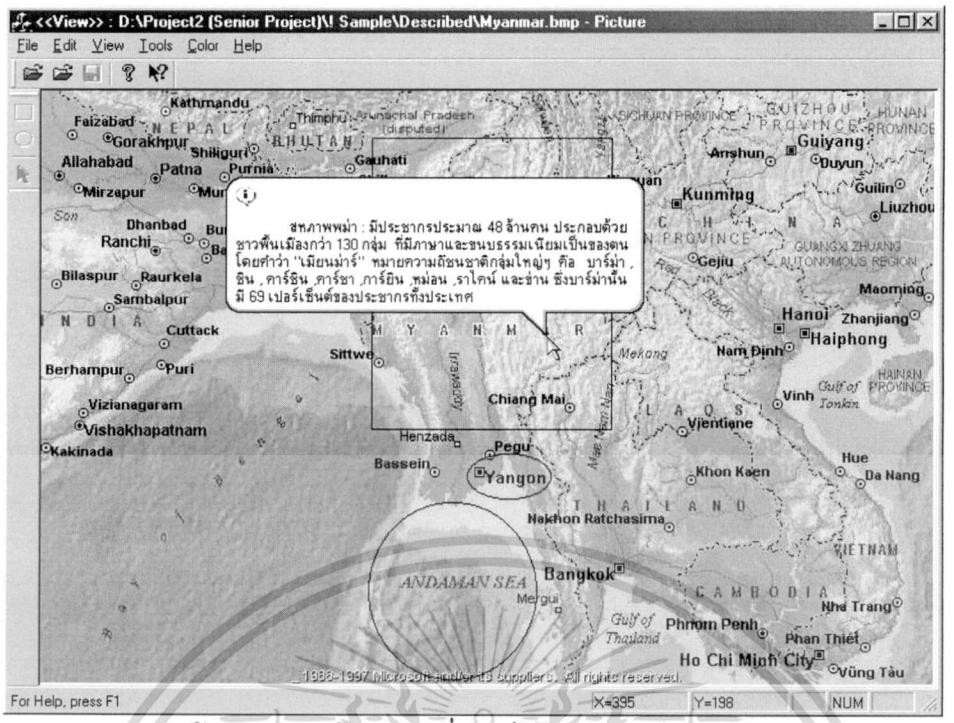
เมื่อทำการซ่อนข้อความลงในส่วนต่าง ๆ เสร็จสิ้นครบทั้ง 3 ส่วนตามที่ได้กล่าวไปแล้วจะปรากฏหน้าจอดังรูป 3.3.9



รูปที่ 3.3.9 **แสดงหน้าจอเมื่อใส่ข้อความลงในส่วนของภาพครบทุกส่วนที่ต้องการ**

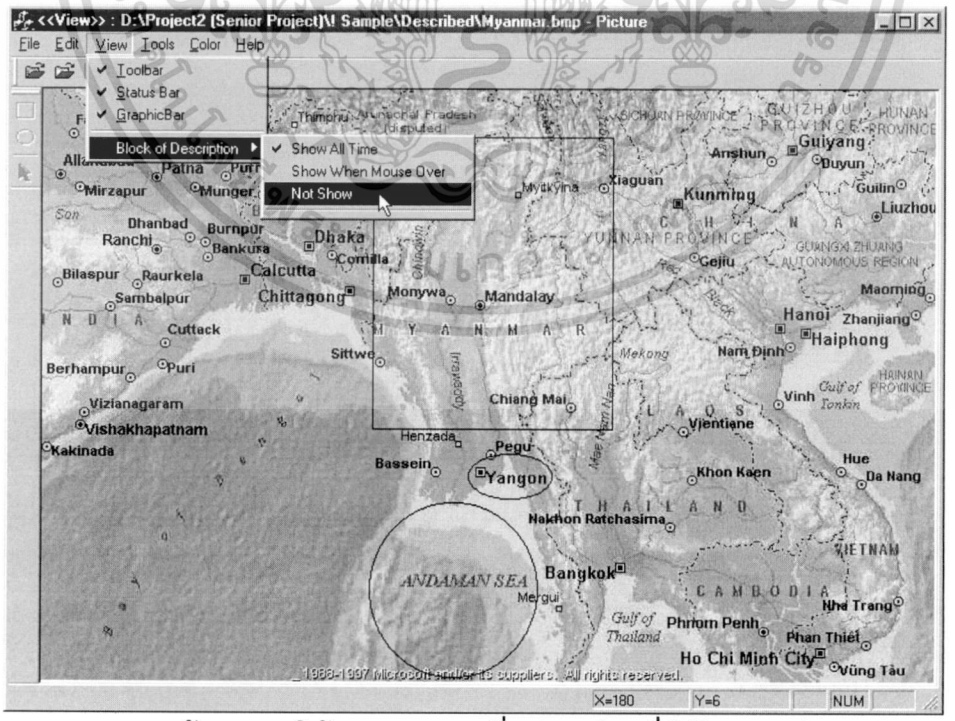
จากนั้นให้ทำการบันทึกโดยไปที่เมนู "File" และเลือกคำสั่ง "Save as" หรือ "Save" เพื่อทำการบันทึกเป็น หรือบันทึก (หรืออาจ click ที่รูป  ที่อยู่บริเวณ toolbar ทางด้านบนก็จะเป็นการ Save) หากต้องการจะแก้ไข เคลื่อนย้ายหรือลบส่วนที่ได้ซ่อนข้อความไปแล้วก็ทำได้โดยไปที่ toolbar ที่อยู่ทางด้านซ้าย click ที่เครื่องมือ Selection ที่เป็นรูป  แล้ว double click หากต้องการจะแก้ไข หรือ drag หากต้องการเคลื่อนย้ายส่วนของ block ที่มีข้อความซ่อนอยู่ แต่ถ้าต้องการลบ block นั้นก็ทำได้โดยไปที่ Menu bar เลือก "Edit" และเลือก ที่ Submenu ที่เขียนว่า "Clear selection" โปรแกรมก็จะทำการลบ block และข้อความที่ซ่อนอยู่ใน block นั้น

โดยหลังจากที่ทำการแก้ไข และบันทึกเสร็จสิ้นแล้วหากต้องการเปิดภาพเพื่อแสดงข้อความที่ซ่อนไว้ในส่วนของภาพขึ้นมาก็สามารถทำได้โดยไปที่คำสั่ง "Open for View Description..." (หรืออาจ click ที่รูป  ที่อยู่บริเวณ toolbar ทางด้านบน) จากนั้นจะปรากฏหน้าจอดังรูปที่ 3.3.10

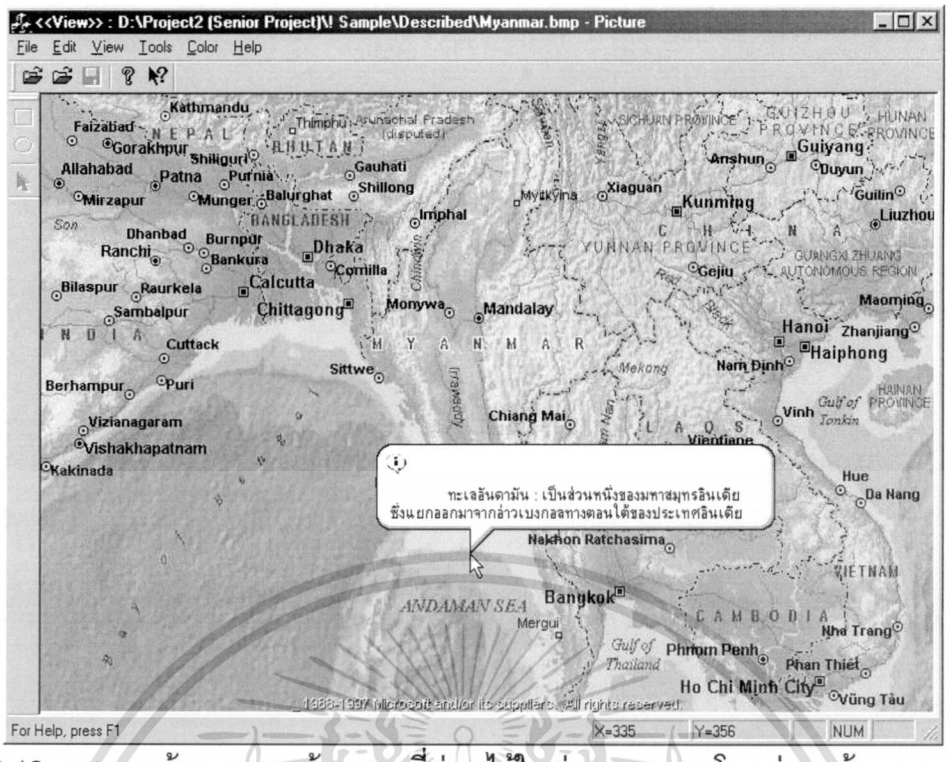


รูปที่ 3.3.10 **แสดงหน้าจอแสดงข้อความที่ซ่อนไว้ในส่วนของภาพ**

หากไม่ต้องการให้โปรแกรมแสดงเส้นขอบของพื้นที่ที่มีข้อความซ่อนอยู่ก็ทำได้ไปที่ "View" ที่อยู่บน Menu bar จากนั้นไปที่ "Block of Description" ดังรูปที่ 3.3.11 จากนั้นจะได้เป็นดังรูป 3.3.12



รูปที่ 3.3.11 **แสดงหน้าจอการใช้เมนู** "View" **เพื่อซ่อนส่วนที่มีข้อความซ่อนอยู่**



รูปที่ 3.3.12 **แสดงหน้าจอแสดงข้อความที่ซ่อนไว้ในส่วนของภาพโดยซ่อนเส้นขอบของพื้นที่ที่มีข้อความซ่อนอยู่**

สำหรับการใช้งานโปรแกรมในส่วนอื่นๆ สามารถดูได้โดยไปที่ "Help" ที่อยู่บน Menu bar จากนั้นไปที่คำสั่ง "Help Topics" และเลือกหัวข้อที่ต้องการความช่วยเหลือดังรูปที่ 3.3.13

รูปที่ 3.3.13 **แสดงหน้าจอขอความช่วยเหลือ**

บทที่ 4
การประเมินผลระบบ

4.1 การประเมินผล

โปรแกรมนี้เป็นโปรแกรมที่พัฒนาขึ้นมาเพื่ออธิบายรายละเอียดของส่วนต่างๆในภาพ โดยให้ผู้ใช้เลือกส่วนต่างๆของภาพที่ต้องการจะใส่รายละเอียด ซึ่งส่วนที่เลือกนั้นจะต้องมีขนาดใหญ่พอที่จะเก็บรายละเอียดที่จำเป็นในการตรวจสอบว่ามีข้อความซ่อนอยู่ในส่วนนี้ และเมื่อผู้ใช้เลือกส่วนของภาพได้แล้ว ผู้ใช้ก็จะสามารถพิมพ์ข้อความที่ต้องการอธิบายในส่วนนั้นๆได้ และหลังจากนั้นข้อความต่างๆจะถูกนำมาบีบอัดเพื่อให้ใช้เนื้อที่ในการเก็บน้อยลง และจะนำข้อมูลดังกล่าวไปฝังลงในภาพ โดยขนาดของข้อความที่ผ่านการบีบอัดแล้วนั้นก็ต้องมีจำนวน bit น้อยกว่าจำนวน pixel ของสีน้ำเงิน (blue channel) ของส่วนของภาพที่เลือกมา โดยถ้ามีจำนวนมากกว่าจำนวน pixel ของสีน้ำเงิน (blue channel) ของส่วนของภาพที่เลือกมา ผู้ใช้ก็จะต้องทำการเลือกส่วนของภาพให้ใหญ่ขึ้น หรือลดปริมาณข้อความที่ต้องการจะใส่ลงในภาพ

จากการทดลองใช้โปรแกรมทำให้พบว่า การที่ผู้ใช้จะเลือกส่วนของภาพแล้วมีขนาดเล็กจนไม่สามารถเก็บรายละเอียดที่จำเป็นในการตรวจสอบว่ามีข้อความซ่อนอยู่นั้น เกิดขึ้นน้อยครั้งมาก

ส่วนการที่จะใส่คำอธิบายแล้วทำให้ขนาดของข้อความที่ผ่านการบีบอัดแล้วมีจำนวน bit มากกว่าจำนวน pixel ของสีน้ำเงิน (blue channel) ของส่วนของภาพที่เลือกมา มีโอกาสเกิดขึ้นได้พอสมควร ขึ้นอยู่กับลักษณะของภาพและข้อความที่ผู้ใช้ต้องการจะอธิบาย และโดยส่วนใหญ่การบีบอัดข้อความในโปรแกรมได้ช่วยลดขนาดของข้อมูลได้ประมาณ 15%-35%

หลังจากที่ผู้ใช้ใส่รายละเอียดลงในภาพและทำการบันทึกภาพแล้วนั้น ภาพดังกล่าวสามารถนำไปเปิดกับโปรแกรมดูภาพอื่นๆได้เหมือนภาพทั่วไป แต่ถ้านำมาเปิดในโปรแกรมนี้ก็จะสามารถแสดงรายละเอียดต่างๆ ที่ได้ใส่ไว้

4.2 ข้อควรปรับปรุงแก้ไข

1. โปรแกรมนี้ไม่สามารถใช้กับภาพ 24 - bit bitmap บางภาพได้ และยังไม่สามารถระบุได้ว่าเป็นภาพไหนที่ไม่สามารถนำมาใช้ได้จนกว่าจะนำมาใช้ในโปรแกรม ซึ่งเมื่อนำภาพดังกล่าวมาใส่ข้อมูลแล้วทำการบันทึกภาพ หลังจากนั้นเมื่อเปิดภาพขึ้นมาอีกครั้ง ภาพที่ออกมาจะเกิดความผิดเพี้ยนไป ซึ่งจากลักษณะของภาพที่เป็นปัญหานี้ไม่น่าจะเป็นผลมาจากการใช้วิธี Least Significant Bit Insertion แต่อย่างใด โดยในขณะทำการพัฒนาโปรแกรมนี้อยู่ยังไม่สามารถหาคำตอบได้ว่าเกิดจากสาเหตุอะไร จึงหวังว่าถ้ามีผู้พัฒนาโปรแกรมนี้ต่อจะช่วยหาคำตอบให้ด้วย

2. ในการซ่อนรายละเอียดลงในภาพนั้น มีข้อจำกัดว่าสามารถที่จะใช้ได้กับภาษาไทยและภาษาอังกฤษเท่านั้น

3. ควรพัฒนาโปรแกรมให้สามารถใช้กับไฟล์ภาพในรูปแบบอื่น ๆ ได้ด้วย หรือไม่ก็ควรที่จะมีการเพิ่มส่วนที่จะทำการแปลงภาพจากไฟล์รูปแบบต่าง ๆ ให้เป็นไฟล์รูปภาพสำหรับใช้ได้ในโปรแกรม นั่นคือ 24-bit bitmap

4. ในการซ่อนรายละเอียดต่าง ๆ ลงในภาพนั้น ถ้าหากว่าขนาดของส่วนของภาพที่เลือกมีขนาดเล็กและข้อความที่ต้องการนำมาใส่เพื่ออธิบายมีขนาดใหญ่ เมื่อนำข้อความที่บีบอัดแล้วไปซ่อนในภาพนั้น อาจจะไม่สามารถซ่อนได้ ดังนั้นจึงควรพัฒนาส่วนที่บีบอัดข้อมูลให้สามารถบีบอัดได้มากกว่านี้ เพื่อให้สามารถใส่รายละเอียดที่ต้องการได้มากขึ้น

5. ในการเลือกส่วนของภาพเพื่อที่จะใส่รายละเอียดลงไปนั้น รูปแบบที่สามารถเลือกได้มี 2 รูปแบบ คือรูปสี่เหลี่ยมกับรูปวงรีหรือวงกลม ควรจะมีการทำรูปแบบอื่นๆ เพื่อที่ผู้ใช้จะสามารถเลือกรูปแบบให้เหมาะกับส่วนที่ต้องการอธิบายรายละเอียดของภาพมากยิ่งขึ้น

6. หากภาพที่เราได้ใส่รายละเอียดลงไปแล้ว ถูกนำไปตัด (crop) และในการตัด (crop) ออกไปนั้นไม่ได้โดนส่วนของภาพที่ได้เลือกและใส่รายละเอียดลงไป โปรแกรมก็ยังคงสามารถแสดงรายละเอียดได้ แต่ถ้าหากว่าส่วนที่ถูกตัด (crop) ออกไปเป็นส่วนหนึ่งของส่วนที่ใส่ข้อความก็จะทำให้การแสดงผลนั้นไม่สมบูรณ์ ดังนั้นในโปรแกรมนี้จะไม่แสดงรายละเอียดในส่วนของภาพที่ไม่สมบูรณ์ คือถูกตัด (crop) ออกไปบางส่วน

บทที่ 5
บทสรุปและข้อเสนอแนะ

5.1 สรุปผล

ผลที่ได้จากการพัฒนาโปรแกรม โดยใช้เทคนิคและหลักการต่างๆที่ได้กล่าวมาแล้ว ในบทที่ 2 จะทำให้เราสามารถสร้างโปรแกรมเพื่อเก็บรายละเอียดเกี่ยวกับไฟล์ภาพได้ดีขึ้น นั่นคือทำให้เราสามารถเก็บรายละเอียดในส่วนต่างๆภายในภาพได้ โดยรายละเอียดนี้จะติดอยู่กับตัวภาพ ทำให้สามารถแสดงคำอธิบายที่ถูกเก็บอยู่ในภาพออกมาได้

โดยโปรแกรมมีความสามารถต่างๆดังนี้

1. โปรแกรมสามารถใส่รายละเอียดเกี่ยวกับส่วนต่าง ๆ ของภาพ โดยที่ไม่ทำให้ภาพนั้นมีขนาดใหญ่ขึ้น เพราะโปรแกรมได้แทรกรายละเอียดต่างๆลงในจุดสีของภาพ ซึ่งคุณภาพของภาพที่เปลี่ยนไปไม่ได้มีผลกับสายตาของมนุษย์ นั่นคือมนุษย์ไม่สามารถสังเกตเห็นสีที่เปลี่ยนไป

2. ก่อนที่จะเก็บคำอธิบายลงในส่วนต่าง ๆ ของภาพ โปรแกรมจะทำการบีบอัดก่อนเพื่อให้สามารถเก็บคำอธิบายต่างๆได้มากขึ้น

3. เนื่องจากโปรแกรมนี้ ทำการจัดเก็บคำอธิบายต่างๆลงในจุดสีของภาพตรงส่วนที่เลือกโดยตรง ดังนั้นถ้าหากภาพนี้โดนตัด (crop) ออกไป โดยส่วนที่เราได้เลือกและใส่คำอธิบายยังอยู่ ภาพก็ยังสามารถแสดงรายละเอียดได้เหมือนเดิม

4. ภาพที่ผ่านการซ่อนข้อมูลในโปรแกรมนี้ สามารถนำไปใช้กับโปรแกรมดูภาพทั่วไปได้

โปรแกรมนี้ถูกพัฒนาขึ้นบน Window 98 และต้องใช้กับภาพ 24-bit bitmap และมีการติดต่อกับผู้ใช้โดยใช้เมาส์

5.2 ข้อเสนอแนะ

1. โปรแกรมควรสามารถใช้กับไฟล์รูปภาพแบบอื่น ๆ ได้นอกเหนือจาก 24 – bit bitmap ซึ่งจะต้องมีการศึกษารูปแบบในการจัดเก็บสำหรับไฟล์ภาพในแต่ละรูปแบบเพื่อจะได้นำไปคิดต่อว่าจะแทรกข้อมูลที่ต้องการเข้าไปในรูปภาพนั้นได้อย่างไร ที่จะไม่ทำให้คุณภาพของภาพนั้นๆ เปลี่ยนแปลงไปจนสามารถที่จะสังเกตได้

2. ในการเลือกส่วนของภาพเพื่อใส่รายละเอียดลงไปนั้น ควรมีการเพิ่มรูปแบบที่สามารถเลือกได้ โดยนอกจากสี่เหลี่ยมและวงรีหรือวงกลมแล้ว อาจจะมีรูปทรงเรขาคณิตอื่นๆ ให้เลือกใช้เพิ่มเติมได้ หรือควรที่จะเลือกส่วนของภาพได้ในลักษณะ Free - Form นั่นคือสามารถลากเมาส์เพื่อเลือกส่วนของภาพเป็นรูปแบบต่างๆที่ต้องการ แต่การเลือกในลักษณะ Free – Form จะไม่สามารถเก็บขอบเขตของส่วนของภาพที่เลือกในรูปแบบโคออร์ดิเนต (x_1, y_1) และ (x_2, y_2) เหมือน

กับสี่เหลี่ยมและวงรีหรือวงกลม เพราะไม่สามารถคำนวณเส้นรอบรูปจากโคออร์ดิเนต (x_1, y_1) และ (x_2, y_2) ได้

3. เนื่องจากในโปรแกรมนี้สามารถใช้กับไฟล์รูปภาพชนิด 24-bit bitmap ซึ่งเป็นแบบ RGB model โดยจะจัดเก็บรายละเอียดที่บอกขอบเขตของส่วนของภาพที่เลือกเอาไว้ใน Red Channel และเก็บข้อมูลที่ใช้อธิบายส่วนของภาพนั้นไว้ใน Blue Channel ดังนั้นไฟล์รูปภาพที่ไม่เป็นแบบ RGB model เช่น ภาพ Monochrome ซึ่งมีเพียง channel เดียว จะไม่สามารถจัดเก็บด้วยวิธีดังกล่าวข้างต้นได้ จึงควรออกแบบโดยจัดเก็บส่วนที่บอกขอบเขตของส่วนของภาพและส่วนของข้อมูลที่จะใช้อธิบายส่วนของภาพนั้นเอาไว้ใน channel เดียวกัน

4. ควรมีการปรับปรุงให้โปรแกรมสามารถใช้ได้กับภาษาอื่น ๆ นอกเหนือจากภาษาไทยและภาษาอังกฤษ

5. ควรมีการปรับปรุงส่วนบีบอัดข้อมูล ให้สามารถบีบอัดข้อมูลได้มีประสิทธิภาพมากขึ้น เพื่อให้ใส่ข้อความได้มากขึ้นในส่วนของภาพที่เลือก โดยอาจจะทำการเพิ่ม huffman model อื่นๆ เข้าไปในโปรแกรม หรืออาจจะเปลี่ยนวิธีการบีบอัดข้อมูลที่มีประสิทธิภาพมากกว่านี้

6. อาจจะพัฒนาโปรแกรมต่อไปโดยการเก็บข้อมูลเสียงไว้ในไฟล์รูปภาพ แต่ต้องคำนึงถึงขนาดของข้อมูลเสียง เช่นเดียวกันกับการคำนึงถึงขนาดของข้อความที่เคยกล่าวไปแล้ว

7. อาจจะมีคำสั่งส่วนที่ใช้ในการ Scan รูปภาพจาก Scanner เข้ามาใช้ในโปรแกรม

8. อาจจะพัฒนาโปรแกรมไปเป็น plug in ฝังไว้ใน Web Browser

9. อาจจะนำโปรแกรมไปพัฒนาต่อ เพื่อการประยุกต์ใช้ในอนาคต โดยนำไฟล์รูปภาพที่มีการแทรกคำอธิบายต่างๆ หรือแทรกส่วนที่เชื่อมโยงไปยัง web page ต่างๆ เอาไว้มาพิมพ์ออกทางกระดาษ แล้วแจกจ่ายไปให้บุคคลต่าง ๆ โดยผู้ที่ได้รับรูปภาพนั้นไปสามารถที่จะนำไป Scan เป็นไฟล์ภาพแล้วนำมาใช้ในโปรแกรมหรือ plug in ใน web browser เพื่อดูคำอธิบายต่างๆที่แทรกเอาไว้หรือจะเชื่อมโยงไปยัง web page ตามที่ได้แทรกเอาไว้ต่อไป

10. การที่จะทำการพัฒนาโปรแกรมนี้ต่อๆไป เพื่อเป็นการเพิ่มประสิทธิภาพของโปรแกรม ผู้พัฒนาควรที่จะต้องเน้นการศึกษาเกี่ยวกับเรื่อง Steganography, Huffman Coding, Graphics File Format

บรรณานุกรม

นิรุธ อำนวยศิลป์. 2544. **คู่มือการเขียนโปรแกรม** Microsoft Visual C++ Version 6.0. กรุงเทพฯ : ซัคเซส มีเดีย.

ยุทธนา ลีลาศวัฒนกุล. 2544. **คู่มือการเขียนโปรแกรม และใช้งาน** Visual C++ 6.0 **ฉบับโปรแกรมเมอร์**. กรุงเทพฯ : อินโฟเพรส.

วุฒิพงศ์ พงษ์สุวรรณ ,น.ต.ดร.. 2543. **เจาะรหัสคอมพิวเตอร์**. กรุงเทพฯ : ซอฟต์แวร์ ปาร์ค.

Bruce Schneier. 2543. The First Book Of Compression & Encryption. California : Hifn publishing.

Duncan Sellars. An Introduction to Steganography. [Online]. Available : http://www.jjtc.com/stegdoc.

Mark Nelson 2539 . The Data Compression Book. New York : Prentic Hall.

http://www.codeguru.com

http://www.codeproject.com

http://www.cwinapp.com

http://www.thaidev.com