2110427 : การบ้านครั้งที่ 2

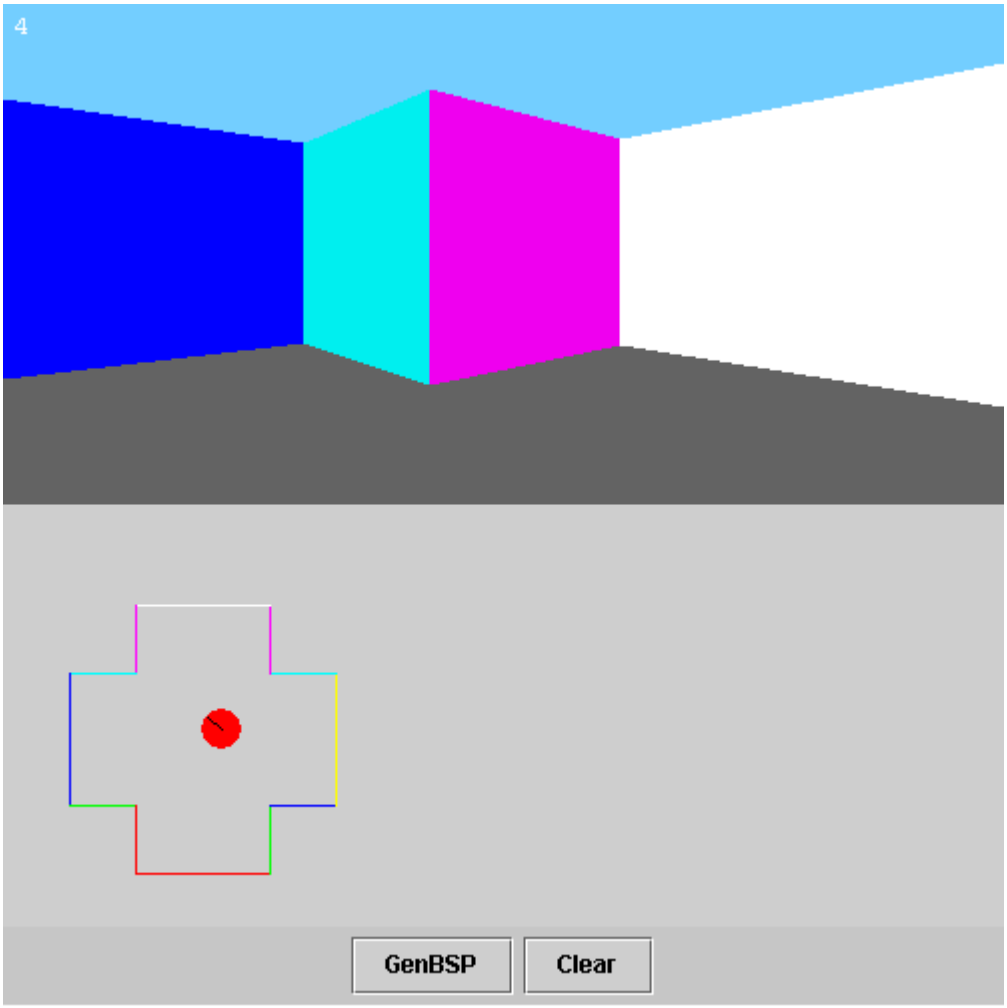
จุดประสงค์

- ศึกษาโครงสร้าง การใช้งาน และการสร้าง binary space partition (BSP) tree
- พัฒนาคลังคำส่งที่ใช้ในการสร้าง ต้นไม้ BSP ที่แทนกำแพงต่างๆในห้อง

ความเป็นมา

ต้นไม้ BSP เป็นโครงสร้างข้อมูลแบบต้นไม้ทวิภาคประเภทหนึ่งที่มีไว้เก็บข้อมูล polytopes ต่างๆ ใน n-dimensional space เพื่อให้อธิบายได้ง่ายขึ้น จะขออธิบายเฉพาะกรณีการเก็บ polygons ใน 3-dimensional space และให้ง่ายขึ้นไปอีก (สำหรับการบ้านนี้) จะเป็นเฉพาะกรณีการเก็บ polygons ที่เป็นสี่เหลี่ยมผืนผ้าความสูงคงที่ ในสามมิติ ซึ่งสี่เหลี่ยมแต่ละอันที่ว่านี้ก็คือกำแพงของห้องนั่นเอง

Applet ข้างล่างนี้ แสดงกำแพงต่างๆ ในห้องหนึ่งเป็นสองมุมมอง ข้างบนแสดงมุมมอง perspective จากตาของคนที่เดินเข้าไปในห้อง ส่วนด้านล่างแสดงมุมมอง top view โดยรูปกลมแดง แทนคนที่กำลังเดินในห้อง เส้นสั้นๆ สีดำในวงแดง แทนทิศทางการมอง ขอให้นักเรียนลองเอาเมาส์ไปกดที่ แล้วลากไปยังบริเวณต่างๆ ของห้องดูว่ามีการเปลี่ยนแปลงอย่างไร (applet นี้อนุญาตในเราเดินทะลุกำแพงได้)



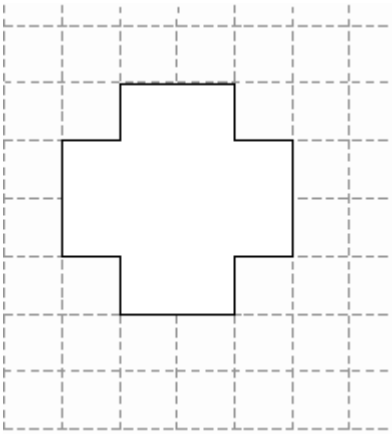
สิ่งที่ต้องทำ

ให้นักเรียนเขียน method **build** ซึ่งอยู่ในแฟ้ม Hw2.java ให้สมบูรณ์ โดย method นี้รับ Vector ของ กำแพงต่างๆ

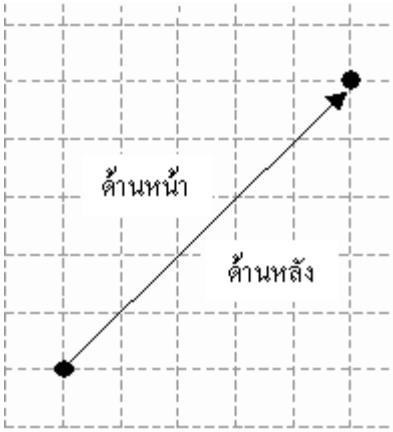
ข้อกำหนดของกำแพง

เนื่องจากกำแพงในห้องเป็นสี่เหลี่ยมที่มีความสูงคงตัว ดังนั้น เราสามารถแทนกำแพงต่างๆ ในห้องด้วยเส้นตรง ซึ่งแทนความยาวและตำแหน่งของกำแพง ตัวอย่างเช่นรายการของเส้นตรงต่างๆ ข้างล่างนี้ทางซ้าย แทนกำแพงต่างๆ ในห้องที่จัดเรียงตามลักษณะข้างล่างนี้ทางขวา

(200, 200) (400, 200)
(400, 200) (400, 300)
(400, 300) (500, 300)
(500, 300) (500, 500)
(500, 500) (400, 500)
(400, 500) (400, 600)
(400, 600) (200, 600)
(200, 600) (200, 500)
(200, 500) (100, 500)
(100, 500) (100, 300)
(100, 300) (200, 300)
(200, 300) (200, 200)



ถ้าพูดถึงกำแพง ก็ต้องมีสองด้าน เส้นตรงที่แทนกำแพงนั้นแทนกำแพงเพียงด้านเดียว โดยเราจะเห็นกำแพงก็ต่อเมื่อเราอยู่ที่**ด้านหน้า**ของ**เส้น** (ขอเน้นว่าของ**เส้น**) ถ้าเราอยู่ด้านหลังเส้น เราจะมองไม่เห็นกำแพง (เสมือนมองทะลุกำแพงได้ อย่าเพิ่งสงสัยว่าทำไมเป็นเช่นนี้) แล้วเราจะรู้ได้อย่างไรว่าเป็นด้านหน้าหรือด้านหลังเส้น เราอาศัยกฎการลากเส้นเป็นตัวกำหนด ดูรูปข้างล่างประกอบ

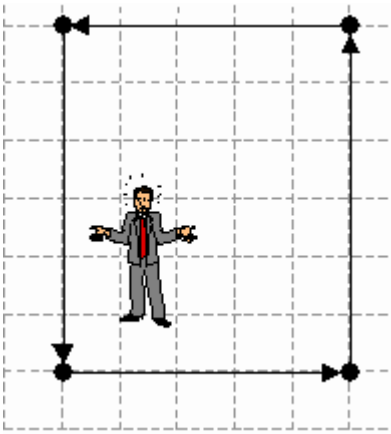


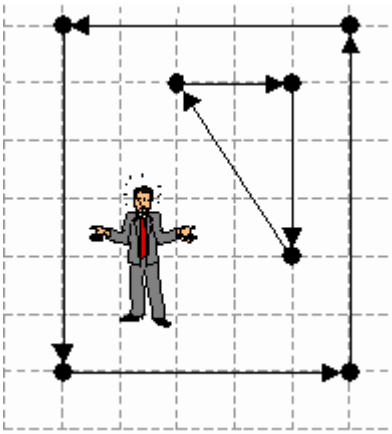
อธิบายง่ายๆ ก็คือ ถ้าอยากรู้ว่าด้านไหนหน้าหรือหลัง ก็ให้คว่ำมือแล้วเอานิ้วชี้วางในแนวเดียวกับเส้น โดยเล็บของนิ้วชี้อยู่ทางจุดปลายของการลากเส้น (หัวลูกศรในรูป) จากนั้นกางนิ้วโป้งให้ตั้งฉากกับนิ้วชี้ ก็จะได้ว่าทิศที่นิ้วโป้งชี้ไปคือด้านหน้า (ฝั่งตรงข้ามก็เป็นด้านหลัง)

หรือจะอธิบายเชิงคณิตศาสตร์จะได้ว่า ถ้าเราต่อเส้นที่เริ่มลากจากจุด (x1,y1) ไปยัง (x2,y2) ให้เป็นเส้นที่มีความยาวอนันต์ จะแบ่งระนาบออกเป็นสองส่วน สำหรับจุด (x, y) ใดๆ บนระนาบขอกำหนดให้ c(x, y) = (x1*dy - y1*dx) + (y*dx - x*dy) โดยที่ dx = (x2 - x1) และ dy = (y2 - y1) จะได้ว่า

ถ้า c(x,y) > 0	(x, y) อยู่ด้านหน้าของเส้น
ถ้า c(x,y) < 0	(x, y) อยู่ด้านหลังของเส้น
ถ้า c(x,y) = 0	(x, y) อยู่แนวเดียวกับเส้น

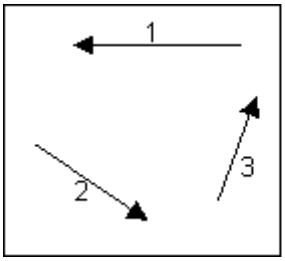
ดังนั้นถ้าเราต้องวาดห้องสี่เหลี่ยมจัตุรัสมีกำแพงสี่ด้าน ก็ต้องลากเส้นทั้งสี่ต่อกันเป็นแบบทวนเข็มนาฬิกา (ดังรูปซ้ายมือข้างล่างนี้) และถ้าภายในห้องนี้มีกล่องสามด้านสูงเท่ากำแพงวางอยู่ ก็ต้องประกอบด้วยเส้นสามเส้นต่อกันตามเข็มนาฬิกา (ดังรูปขวามือข้างล่างนี้)

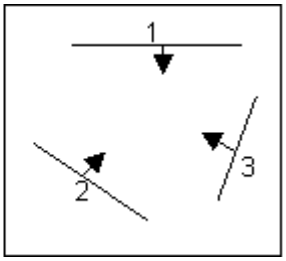




ดังนั้นถ้าเราต้องการให้แทนกำแพงทั้งสองด้าน ก็ต้องมีสองเส้น

ก่อนอื่นขอเขียนเส้นในอีกแบบจะเห็นเรื่องด้านหน้าด้านหลังได้ง่ายกว่า โดยเขียนเส้นที่มีลูกศรตั้งฉากกับกำแพง มีทิศชี้ไปยังด้านหน้ากำแพง (เขาเรียงเส้นที่มีลูกศรในภาษาทาง computer graphics ว่า normal vector) ดังรูปข้างล่างนี้





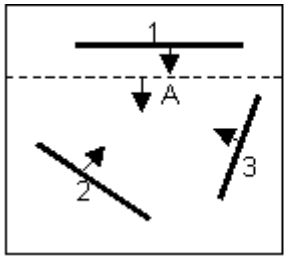
เราแทนกำแพงด้วยเส้น เราแทนเส้นหนึ่งเส้นด้วย array ขนาด 4 ช่องแบบ int ถ้าเส้นๆ หนึ่งลากจากจุด (x1, y1) ไปยัง (x2, y2) จะเก็บ int ต่างๆ ใน array เรียงดังนี้ (x1, y1, x2, y2)

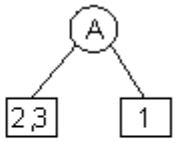
ดังนั้นเราแทนกำแพงต่างๆ ด้วย Vector ของเส้น (Vector เป็น class มาตรฐานที่มีให้ใช้ใน java.util.Vector)

BSP Tree

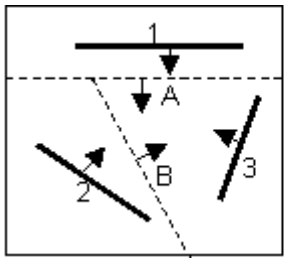
ต้นไม้ BSP เป็นต้นไม้ทวิภาคที่เกิดมาจากการแบ่งระนาบออกเป็นส่วนๆ ตามลำดับของเส้นแบ่งส่วนระนาบ (partition line) เส้นแบ่งส่วนนี้เป็นเส้นตรงความยาวอนันต์ ซึ่งแบ่งระนาบออกเป็นสองส่วน ส่วนด้านหน้าเส้นแบ่ง และส่วนด้านหลังเส้นแบ่ง จะขอแสดงให้เห็นเป็นตัวอย่างกันเลยจะดีกว่า

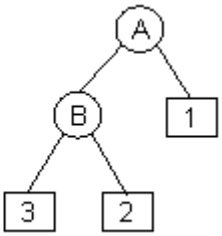
ในตัวอย่างข้างล่างนี้ เรามีกำแพง 3 เส้น (จะขอใช้คำว่ากำแพง แทนคำว่าเส้นตรง เพราะจะได้ไม่สับสนกับเส้นแบ่งส่วนระนาบ ซึ่งก็เป็นเส้นตรงเหมือนกัน) ถ้าเราลากเส้นตรงแบ่งส่วนระนาบดังเส้น A ในรูป จะแบ่งกำแพงต่างๆ ในระนาบออกเป็นสองส่วนคือกำแพงที่อยู่ด้านหน้าของ A (ซึ่งคือกำแพง 2 และ 3) และกำแพงที่อยู่ด้านหลังของ A (ซึ่งคือกำแพง 1) เขียนแทนด้วยต้นไม้ได้ดังรูป โดยที่กิ่งซ้ายแทนด้านหน้า กิ่งขวาแทนด้านหลัง ที่รากเก็บเส้นแบ่งส่วนระนาบ



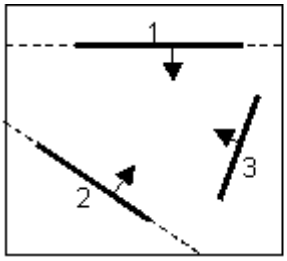


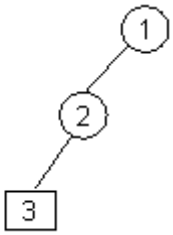
เนื่องจากลูกซ้ายของ A มีกำแพงมากกว่าหนึ่ง ก็จะทำการแบ่งส่วนต่ออีก คราวนี้ขอลากเส้นแบ่งส่วนระนาบ B ดังรูปข้างล่างนี้ ก็เป็นในทำนองเดียวกัน คือ B แบ่งกำแพง 2 และ 3 ออกเป็นส่วนที่อยู่หน้า B คือกำแพง 3 และส่วนที่อยู่หลัง C คือกำแพง 2



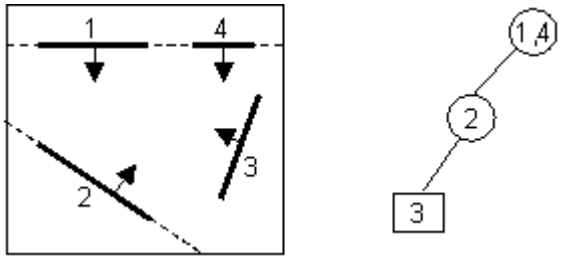


ปัญหาที่น่าสนใจก็คือว่าจะลากเส้นแบ่งส่วนระนาบอย่างไรดี ? วิธีง่ายๆ ก็คือ นำกำแพงที่มีอยู่มาเป็นตัวกำหนดเส้นแบ่งส่วนระนาบเลย สมมติว่าเราเริ่มด้วยกำแพงทั้งสามเหมือนเดิม คราวนี้เราเลือกกำแพง 1 เป็นตัวกำหนดแนวการแบ่งส่วนระนาบครั้งแรก และตามด้วยการเลือกกำแพง 2 เป็นตัวแบ่งส่วนระนาบสำหรับลูกทางซ้ายของ 1 จะได้ดังรูปข้างล่างนี้

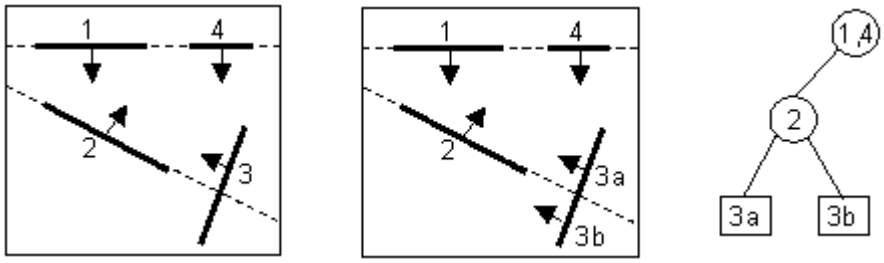




ดังนั้นที่ปมต่างๆ ของต้นไม้ ก็ต้องเก็บกำแพงซึ่งเป็นตัวกำหนดแนวการแบ่งส่วนระนาบไว้ด้วยดังแสดงในรูปข้างบนนี้ และถ้ามีหลายๆ กำแพงอยู่บนเส้นแบ่งส่วนระนาบ ก็ต้องเก็บกำแพงเหล่านั้นในปมด้วย ดังตัวอย่างข้างล่างนี้ กำแพง 1 และ 4 อยู่บนเส้นแบ่งส่วนระนาบ ก็จะได้ต้นไม้ BSP ดังรูปขวา



แล้วถ้าเส้นแบ่งส่วนระนาบที่ลากนั้น ไปตัดกำแพงที่มีอยู่ จะทำอย่างไร วิธีง่ายๆ ก็คือก็แบ่งกำแพงที่ถูกตัดออกเป็นสองกำแพง (ที่ความจริงต่อกัน) เลย กำแพงย่อยส่วนหนึ่งก็ต้องอยู่ด้านหน้าเส้นแบ่งส่วน กำแพงย่อยอีกส่วนก็ต้องอยู่ด้านหลังเส้นแบ่งส่วน รูปข้างล่างนี้ กำแพง 3 ถูกเส้นแบ่งส่วนระนาบ (ที่อยู่บนแนวเดียวของกำแพง 2) ตัด กำแพง 3 จึงถูกแบ่งออกเป็นสองส่วนคือ 3a และ 3b ได้ต้นไม้ BSP ดังรูปขวา



จากตัวอย่างข้างต้นนี้ สรุปได้ว่าต้นไม้ BSP แบ่งกำแพงต่างๆ ที่มีอยู่ในระนาบออกเป็นกลุ่มๆ ซึ่งถูกกำหนดโดยเส้นแบ่งส่วนระนาบ ถ้าพิจารณาที่ต้นไม้ย่อย T ใดๆ ในต้นไม้ BSP จะพบว่า ที่รากของ T จะเก็บกำแพงต่างๆ ที่อยู่บนแนวเดียวกับเส้นแบ่งส่วนระนาบ กำแพงต่างๆ ที่อยู่ในต้นไม้ย่อยทางซ้ายของ T จะอยู่ด้านหน้าเส้นแบ่งส่วนที่ราก T และกำแพงต่างๆ ที่อยู่ในต้นไม้ย่อยทางขวาของ T จะอยู่ด้านหลังเส้นแบ่งส่วนที่ราก T

การใช้ต้นไม้ BSP

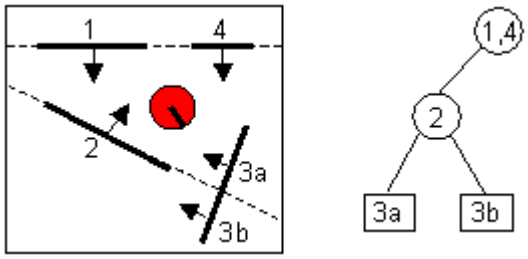
เราใช้ต้นไม้ BSP ได้มากมาย (อ่าน BSPfaq) แต่ที่เราใช้ใน applet ข้างบนก็คือใช้สำหรับการทำ hidden surface removal (HSR) ในการวาดกำแพงต่างๆ แบบ Painter's algorithm (นั่นคือวาดของที่อยู่ไกลกว่าก่อน แล้ววาดของที่อยู่ใกล้กว่าทับของที่อยู่ไกลกว่า) และลดจำนวนการวาดกำแพงที่ไม่จำเป็นต้องวาด ด้วยการเปรียบเทียบตำแหน่งของคนกับเส้นแบ่งส่วนระนาบจะได้ลำดับการวาดดังนี้

ถ้าคนอยู่ด้านหน้าเส้นแบ่งส่วนระนาบ ก็ให้วาดกำแพงต่างๆ ที่อยู่ด้าน**หลัง**เส้นแบ่งส่วน ตามด้วยกำแพงต่าง ที่อยู่บน**แนวเดียว**กับเส้นแบ่งส่วน แล้วจึงค่อยวาดกำแพงต่างๆ ที่อยู่**หน้า**เส้นแบ่งส่วน

ถ้าคนอยู่ด้านหลังเส้นแบ่งส่วนระนาบ ก็ให้วาดกำแพงต่างๆ ที่อยู่ด้าน**หน้า**เส้นแบ่งส่วน ตามด้วยกำแพงต่าง ที่อยู่บน**แนวเดียว**กับเส้นแบ่งส่วน แล้วจึงค่อยวาดกำแพงต่างๆ ที่อยู่**หลัง**เส้นแบ่งส่วน

โดยขณะที่จะวาดกำแพงใด ก็ให้คำนวณมุมมองก่อนว่า อยู่ในด้านหน้าตาคนมองหรือไม่เสียก่อนจะวาดด้วย จะได้ไม่ต้องวาดกำแพงที่อยู่ด้านหลังตาของคนมอง

ตัวอย่างเช่นรูปข้างล่างนี้แสดงคนด้วยวงกลมแดงอยู่ในห้องหันหน้ามองตามแนวขีดดำในวงกลม การวาดกำแพงต่างๆ ที่คนๆ นี้มองเห็น จะเริ่มที่ ราก 1,4 พบว่าคนนี้อยู่ด้านหน้าเส้นแบ่งส่วนที่ราก ดังนั้นจะขอวาดกลุ่มของกำแพงด้านหลังเส้นแบ่งส่วนก่อน (ซึ่งในที่นี้ไม่มี) หลังจากนั้นจึงวาดกำแพง 1 และ 4 (แต่พออยากจะวาด 1 และ 4 ก็พบว่าอยู่ด้านหลังตาของคนมอง ก็ไม่มีอะไรวาดอีก) แล้วจึงตามด้วยกลุ่มของกำแพงด้านหน้าของเส้นแบ่งส่วน (ซึ่งคือต้นไม้ย่อยทางซ้ายของราก 1,4) ก็ลงไปต้นไม้ย่อยนี้ต่อ พบว่าคนอยู่หน้าเส้นแบ่งส่วนก็ต้องวาดกำแพงหลังเส้นแบ่งส่วนก่อน (ซึ่งในที่นี้คือ 3b) ตามด้วยการวาดกำแพงในบนแนวเดียวกับเส้นแบ่งส่วน (ซึ่งคือ 2) แล้วปิดท้ายด้วยกำแพงที่อยู่หน้าเส้นแบ่งส่วน (ซึ่งคือ 3a)



หมายเหตุ นักเรียนไม่ต้องเขียนโปรแกรมการวดกำแพงนี้หรอก เพียงแต่ต้องการให้เข้าใจการประยุกต์ต้นไม้ BSP เท่านั้นเอง

คำแนะนำ

- กำแพงแทนด้วยเส้น
- เส้นแทนด้วยจุดสองจุด
- จุดแทนด้วย int สองตัว
- ดังนั้นกำแพงแทนด้วย array ของ int (ขนาด 4 ช่อง ช่องที่ 0 ถึง 3) กำแพงที่แทนด้วยเส้นซึ่งลากจากจุด (x1, y1) ไปยัง (x2, y2) จะเก็บใน array เรียงดังนี้ (x1, y1, x2, y2)
- กำแพงต่างๆ ในห้องแทนด้วย Vector ของกำแพง
- เราแทนเส้นแบ่งส่วนระนาบซึ่งผ่านจุด (x1, y1) และ (x2, y2) ด้วยสมการ y = (dy/dx)x + b โดยที่ dy = (y2-y1) dx = (x2 - x1) และ b คือ (y1*dx - x1*dy) ดังนั้นจะขอแทนเส้นแบ่งส่วนระนาบด้วย array ของ float ขนาดสามช่อง ช่องที่ 1 เก็บ dy ช่องที่สองเก็บ dx และช่องที่สามเก็บ (y1*dx - x1*dy)
- แต่ละปมในต้นไม้ BSP เป็น object ของ class BspNode

```
class BspNode{
  public float[] partition = null;
  public Object[] lines = null;
  public BspNode front = null;
  public BspNode back = null;
}
```

partition คือเส้นแบ่งส่วนระนาบกำกับปมนี้
lines คือ array ซึ่งเก็บเส้น (กำแพง) ต่างๆ ที่อยู่บนแนวเดียวกับ partition
front คือ รากของต้นไม้ย่อย BSP ที่เก็บกำแพงต่างๆ ที่อยู่ด้านหน้าของ partition
back คือ รากของต้นไม้ย่อย BSP ที่เก็บกำแพงต่างๆ ที่อยู่ด้านหลังของ partition

- เขียน method เสริมต่างๆ ดังต่อไปนี้

```
float[] getLineEquation(int[] line)
int     evalPoint(int x,int y,float[] partition)
int     evalLine(int[] line,float[] partition)
int[][] splitLine(int[] line,float[] partition)
```

  - getLineEquation รับเส้นตรงมาหนึ่ง (line) จากนั้นคำนวณค่า dy, dx, และ (y1*dx - x1*dy) เก็บใส่ array ของ float สามช่อง คืนกลับมาเป็นตัวแทนเส้นแบ่งส่วนระนาบ
  - evalPoint พิจารณาว่าจุด (x, y) นั้นอยู่ด้านหน้า อยู่ด้านหลัง หรืออยู่บนเส้นแบ่งส่วนระนาบ partition
  - evalLine พิจารณาว่าเส้นตรง line อยู่ด้านหน้า อยู่ด้านหลัง อยู่บน หรือตัดกับเส้นแบ่งส่วนระนาบ partition
  - splitLine ทำการแยกเส้นตรง line ซึ่งถูกเส้นแบ่งส่วนระนาบ partitionตัดผ่าน ออกเป็นสองเส้น เก็บใส่ array 2 มิติ โดยที่เส้นแรกอยู่ด้านหน้า partition เก็บใน int [0][ ] และเส้นที่สองอยู่ด้านหลัง partition เก็บใน int [1][ ]

- จากนั้นจึงเขียน build ซึ่งคือ method หลักที่ต้องการ ซึ่งมีโครงดังนี้

```
public BspNode build(Vector lines){
  BspNode root = new BspNode();
  build(root,lines);
  return root;
}

private void build(BspNode tree, Vector lines {

  ...

}
```

download

- unzip แฟ้ม 427hw2.zip ลงใน directory ที่เก็บเฉพาะการบ้านนี้ จะพบแฟ้มต่างๆ ดังนี้
  - Hw2.java
  - hw2.jar
  - map.txt

    - readme.pdf
    - hw2.htm
- แก้ไขแฟ้ม Hw2.java ด้วย editor อะไรก็ได้ (เช่น notepad, edit plus, ...)
- compile ด้วย javac -classpath hw2.jar Hw2.java
- สั่งทำงานด้วย java -classpath hw2.jar Bsp
  (Bsp เป็น class หลักสำหรับการเริ่มต้นการทำงานของการบ้านนี้)

วิธีทดสอบ

- เปิด hw2.htm ด้วย Internet Explorer เพื่อดูการทำงานของ applet (เนื่องจาก applet นี้มีการใช้อะไรบางอย่างของ Java 2 ระบบที่ใช้งานจึงต้องได้รับการติดตั้ง Java Plug-in ซึ่งจะติดตั้งให้อยู่แล้ว ถ้าได้ลง Java SDK 1.3 ในระบบ)

กำหนดส่ง

- วันที่ 19 กรกฎาคม 2544