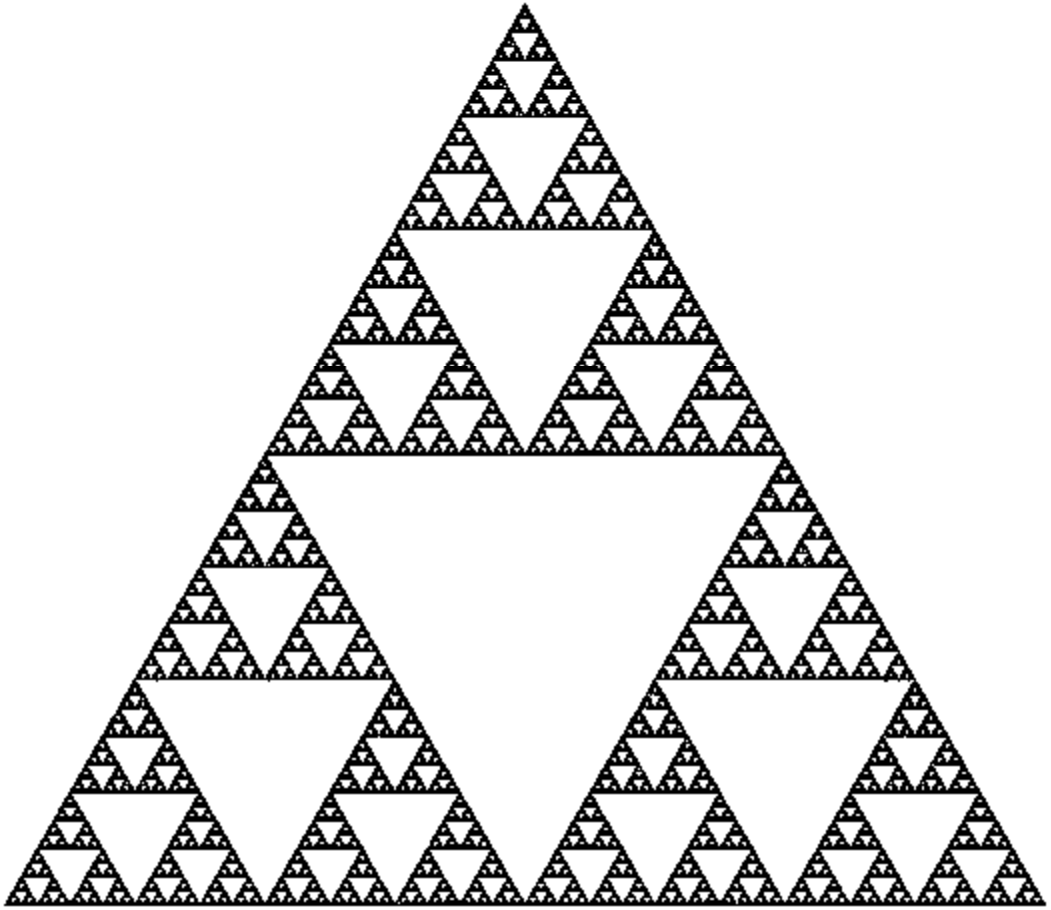
418341: สภาพแวดล้อมการทำงานคอมพิวเตอร์กราฟิกส์
การบรรยายครั้งที่ 7

pramook@gmail.com

การแปลงกับการวาดภาพ

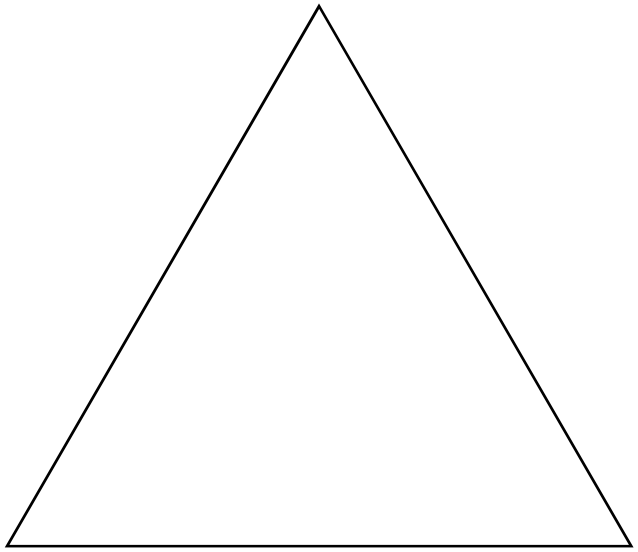
- การแปลงสามารถนำมาใช้สร้างภาพที่มีความซับซ้อนได้มากมาย
- เราจะมาดูตัวอย่างการสร้าง **แฟรกตัล (fractal)**
  - รูปที่พอเอาแว่นขยายส่องดูแล้วเห็นเป็นลักษณะเหมือนกับตอนไม่ได้ใช้แว่นขยายดู

Sierpinski Triangle



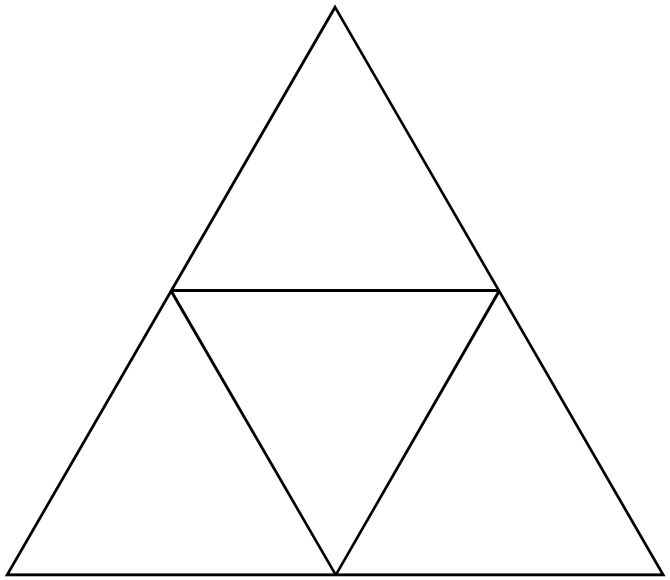
Sierpinski Triangle

- เพื่อความง่ายในการสร้าง เราจะแบ่ง Sierpinski triangle ออกเป็นชั้นๆ
- ชั้นที่ 0 เป็นสามเหลี่ยมด้านเท่าธรรมดา



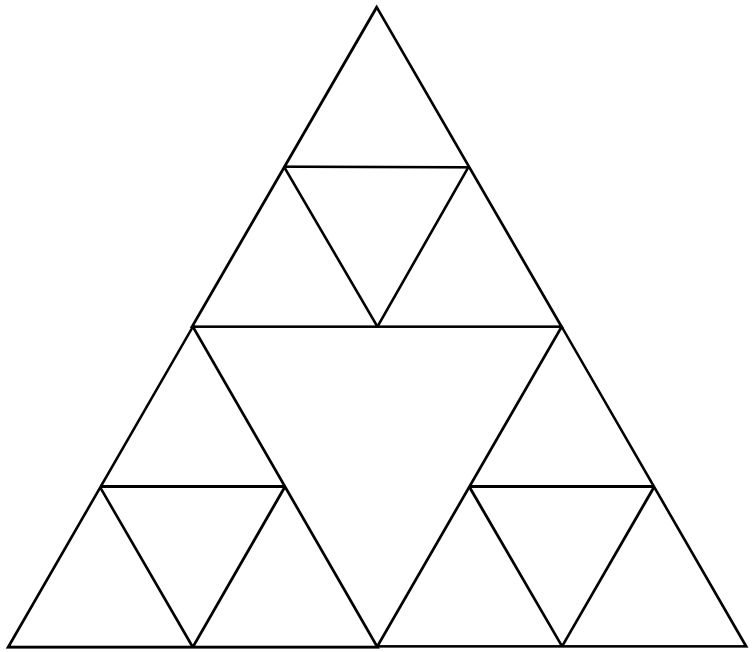
Sierpinski Triangle

- ชั้นที่ 1 เกิดจากการเอา Sierpiński triangle ชั้นที่ 0 ที่ย่อส่วนลงสองเท่ามาเรียงกันตามรูปข้างล่างนี้



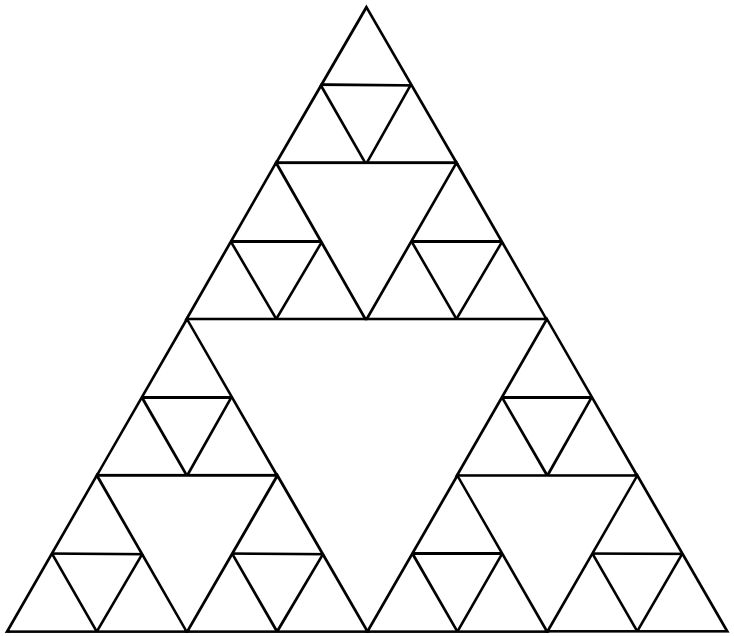
Sierpinski Triangle

- Sierpiński Triangle ชั้นที่ 2 เกิดจากการเอา Sierpiński triangle ชั้นที่ 1 ที่ย่อส่วนลงสองเท่ามาเรียงกันตามรูปข้างล่างนี้



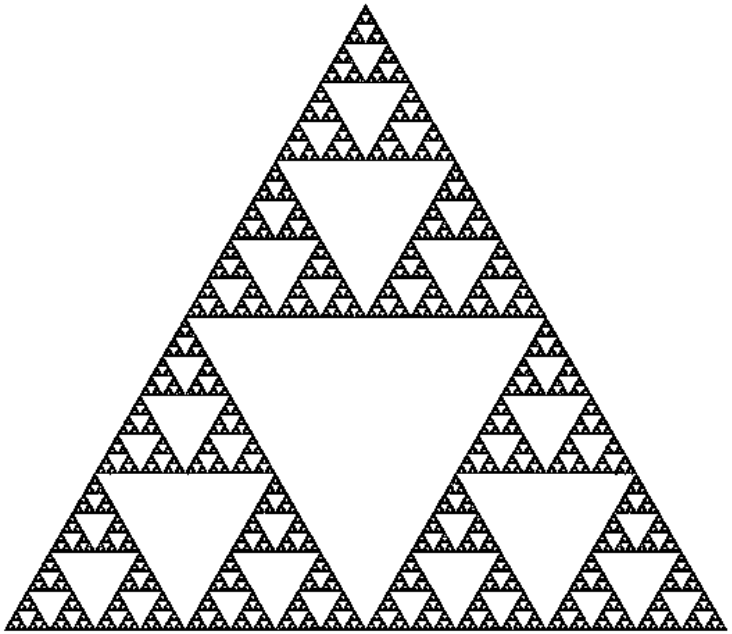
Sierpinski Triangle

- Sierpiński Triangle ชั้นที่ 3 เกิดจากการเอา Sierpiński triangle ชั้นที่ 2 ที่ย่อส่วนลงสองเท่ามาเรียงกันตามรูปข้างล่างนี้



Sierpinski Triangle

- Sierpiński Triangle ชั้นที่ k เกิดจากการเอา Sierpiński triangle ชั้นที่ k-1 ที่ย่อส่วนลงสองเท่ามาเรียงกันตามรูปเดิม
- ข้างล่างนี้คือ Sierpiński triangle ประมาณชั้นที่ 8

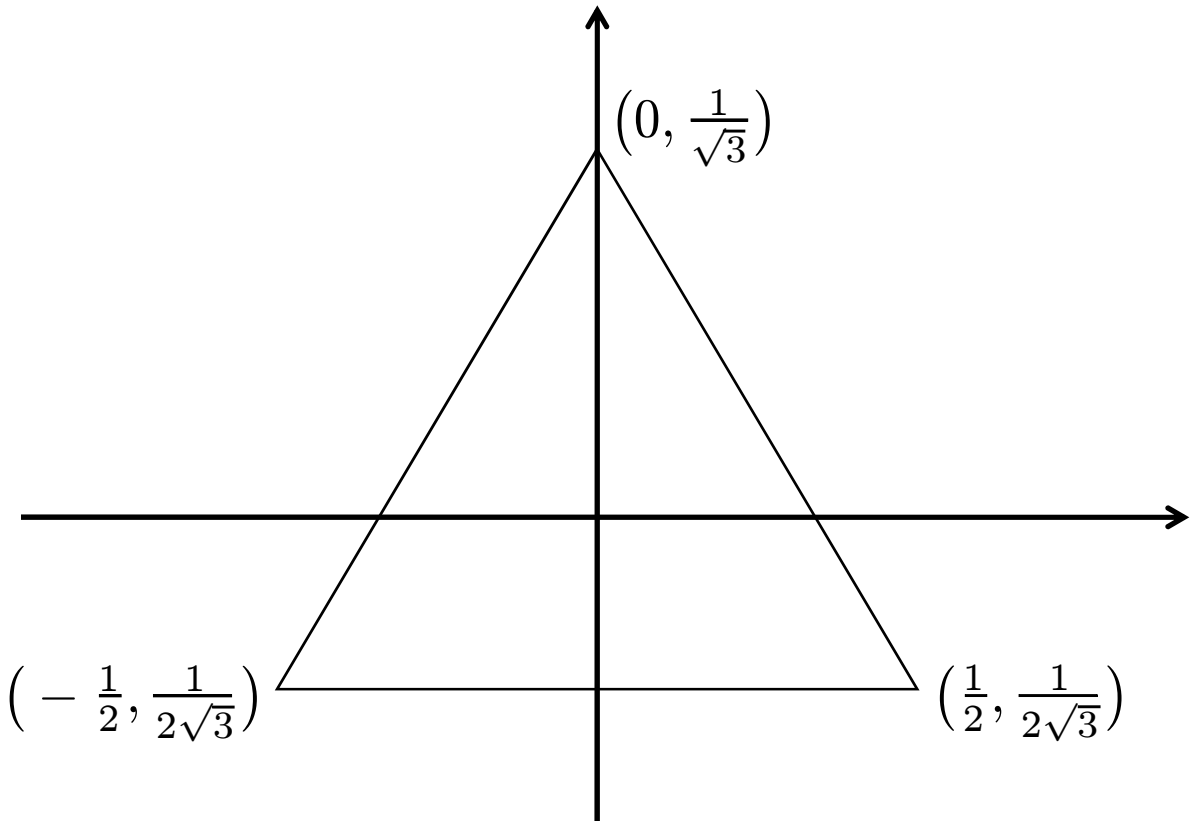


เราจะวาด Sierpinski Triangle ได้อย่างไร?

- เขียนฟังก์ชัน void draw_sierpinski(int k) ที่ทำการวาด Sierpinski Triangle ชั้นที่ k
- มีกฎอยู่สองข้อในการวาด Sierpinski Triangle
  - ถ้า k = 0 ให้วาดสามเหลี่ยมด้านเท่า
  - ถ้า k > 0 ให้วาด Sierpinski Triangle ชั้นที่ k-1 สามอันเรียงกันตามรูปที่เราเคยเห็นมา

วาดสามเหลี่ยมด้านเท่า

- ต้องการวาดสามเหลี่ยมด้านเท่าความยาวด้านละ 1 หน่วย
- จุดศูนย์กลาง (จุด centroid) อยู่ที่จุด (0,0)



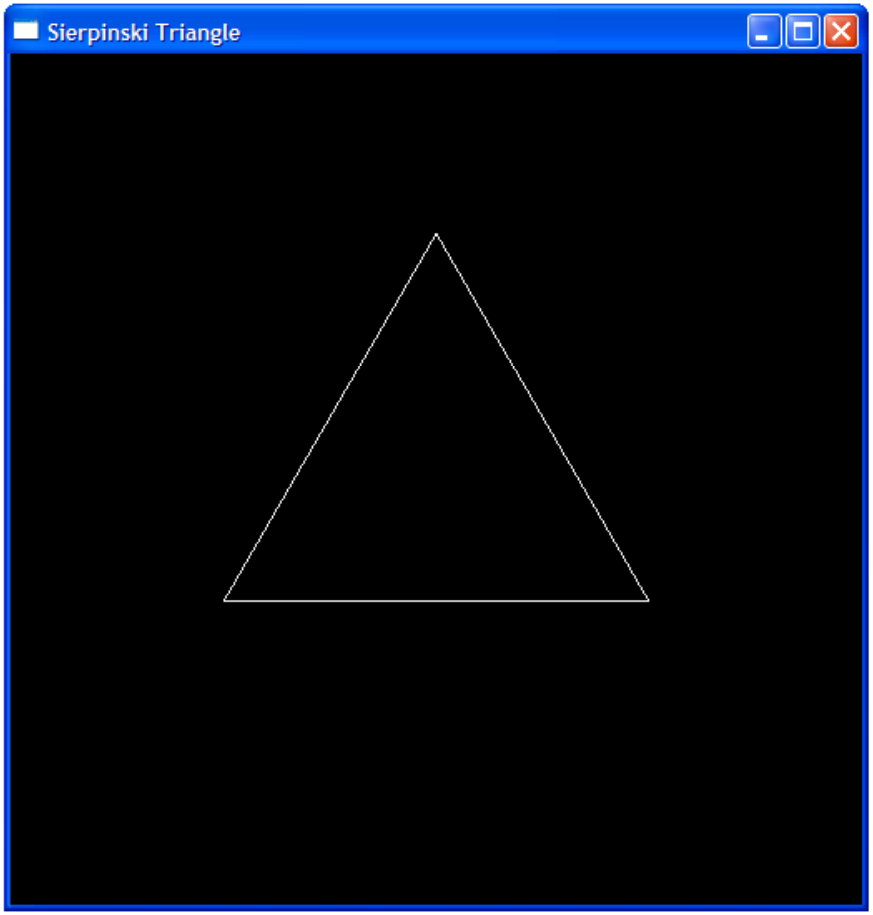
วาดสามเหลี่ยมด้านเท่า

```
void draw_triangle()
{
  glBegin(GL_LINE_LOOP);

  glVertex2d( 0.0,  1.0/sqrt(3.0));
  glVertex2d( 0.5, -0.5/sqrt(3.0));
  glVertex2d(-0.5, -0.5/sqrt(3.0));

  glEnd();
}
```

ผลลัพธ์

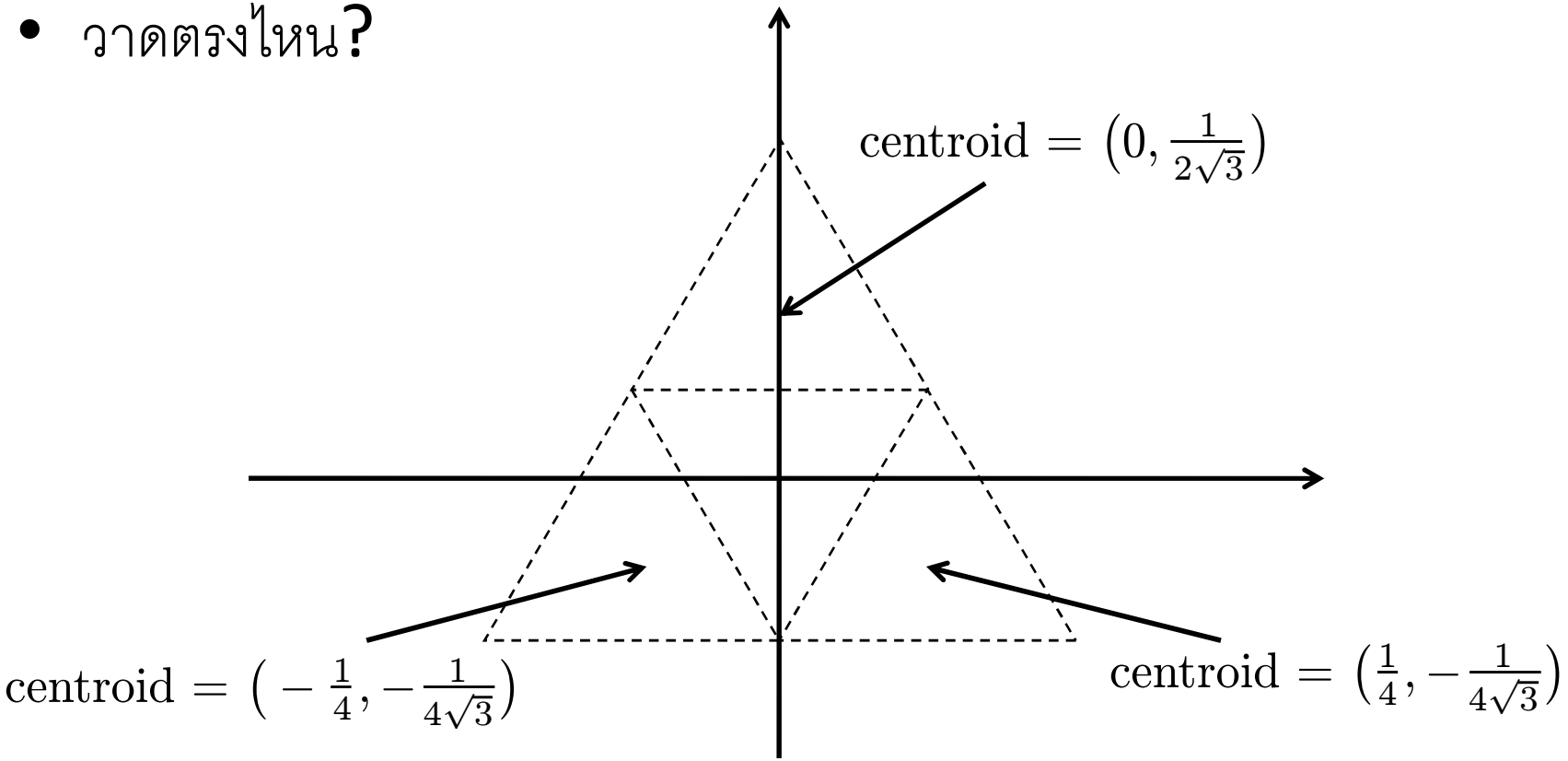


ฟังก์ชัน draw_sierpinski

```
void draw_sierpinski(int k)
{
  if (k == 0)
     draw_triangle();
  else
  {
     // วาด Sierpinski triangle
     // ชั้นที่ k-1 สามอัน
  }
}
```

วาด Sierpinski Triangle ชั้นที่ k-1 สามอัน

- วาดอย่างไร?
  - เรียก draw_sierpinski(k-1)
- วาดตรงไหน?



วาด Sierpinski Triangle ชั้นที่ k-1 สามอัน

- วาด Sierpinski Triangle อันบน
  - Translate centroid ไปเป็นจุด $(0, \frac{1}{2\sqrt{3}})$
  - Scale ขนาดลดลง 2 เท่า (= ขยาย 0.5 เท่า)
- วาด Sierpinski Triangle อันล่างขวา
  - Translate centroid ไปเป็นจุด $(\frac{1}{4}, -\frac{1}{4\sqrt{3}})$
  - Scale ขนาดลดลง 2 เท่า (= ขยาย 0.5 เท่า)
- วาด Sierpinski Triangle อันล่างซ้าย
  - Translate centroid ไปเป็นจุด $(-\frac{1}{4}, -\frac{1}{4\sqrt{3}})$
  - Scale ขนาดลดลง 2 เท่า (= ขยาย 0.5 เท่า)

วาด Sierpinski Triangle ชั้นที่ k-1 สามอัน

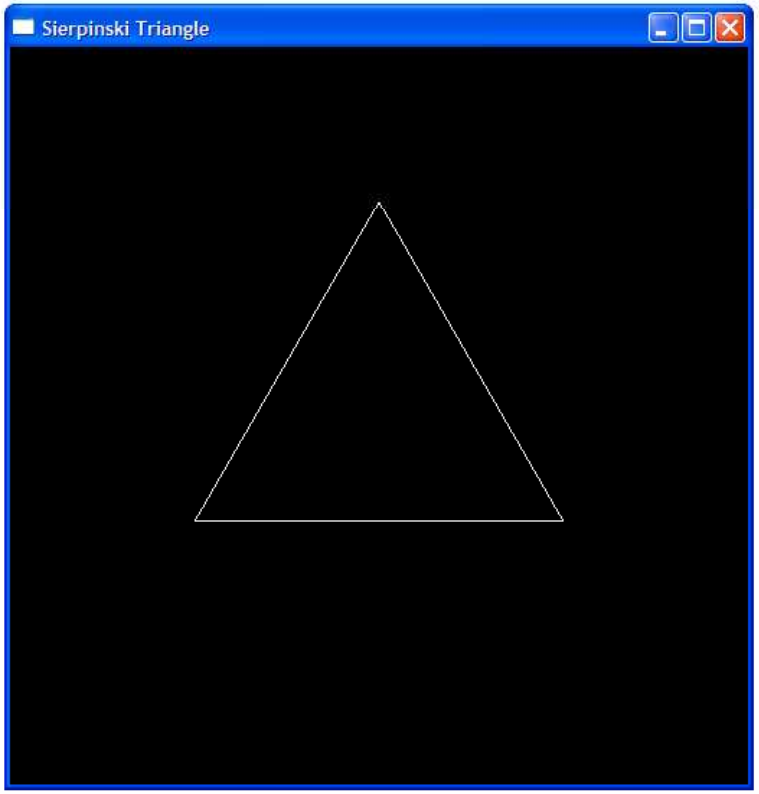
```
void draw_sierpinski(int k)
{
    if (k == 0)
          draw_triangle();
    else
    {
          glLoadIdentity();
          glTranslated(0.0, 0.5 / sqrt(3.0), 0.0);
          glScaled(0.5, 0.5, 0.5);
          draw_sierpinski(k-1);

          glLoadIdentity();
          glTranslated(0.25, -0.25 / sqrt(3.0), 0.0);
          glScaled(0.5, 0.5, 0.5);
          draw_sierpinski(k-1);

          glLoadIdentity();
          glTranslated(-0.25, -0.25 / sqrt(3.0), 0.0);
          glScaled(0.5, 0.5, 0.5);
          draw_sierpinski(k-1);
    }
}
```

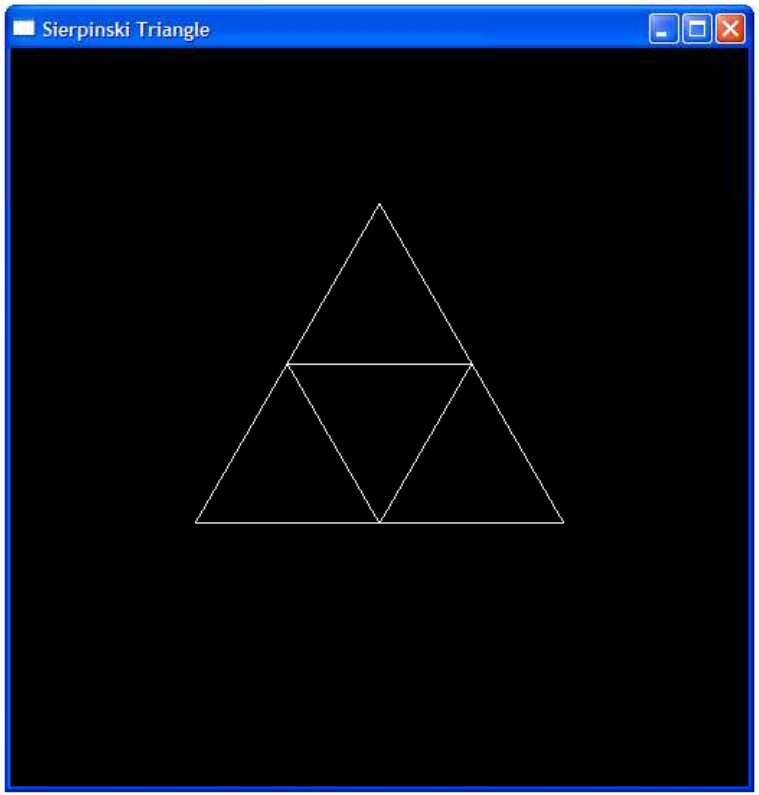
วาด Sierpinski Triangle ชั้นที่ k-1 สามอัน

- draw_siepinski(0)



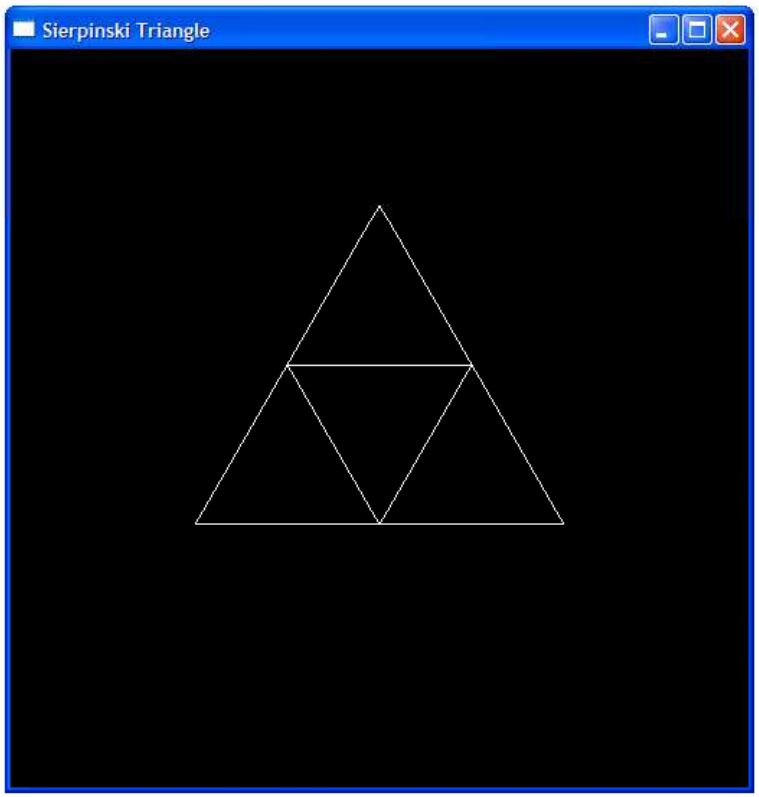
วาด Sierpinski Triangle ชั้นที่ k-1 สามอัน

- draw_siepinski(1)



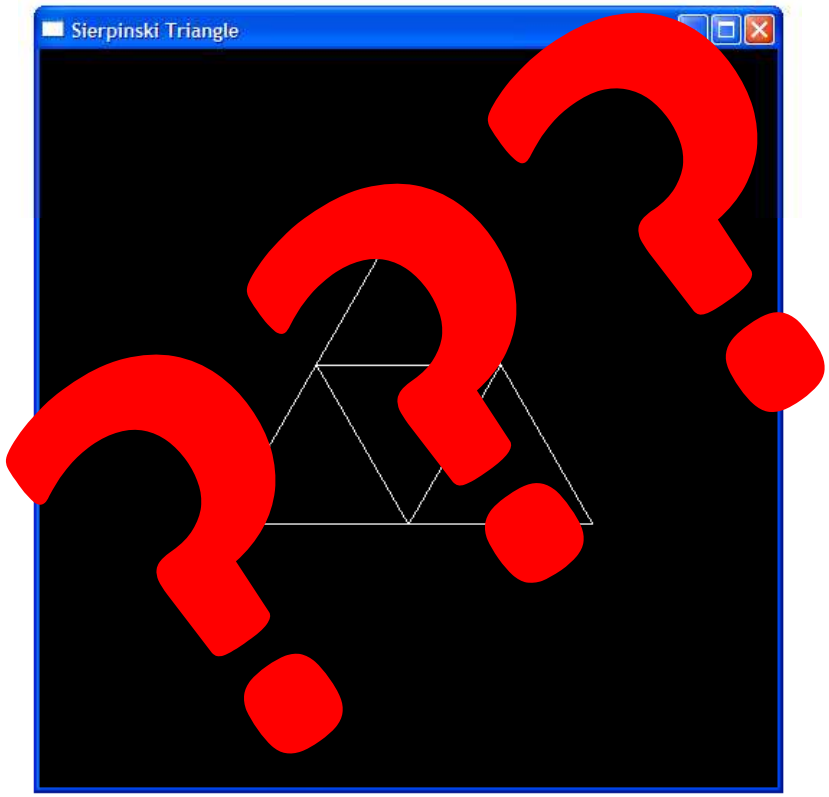
วาด Sierpinski Triangle ชั้นที่ k-1 สามอัน

- draw_siepinski(2)

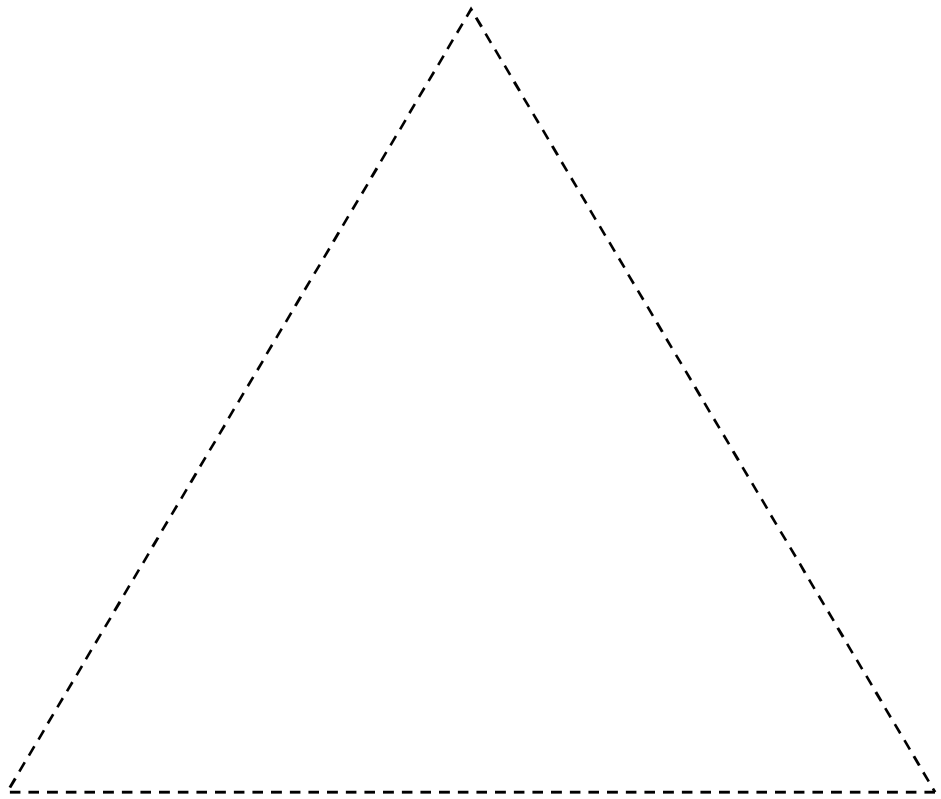


วาด Sierpinski Triangle ชั้นที่ k-1 สามอัน

- draw_siepinski(2)

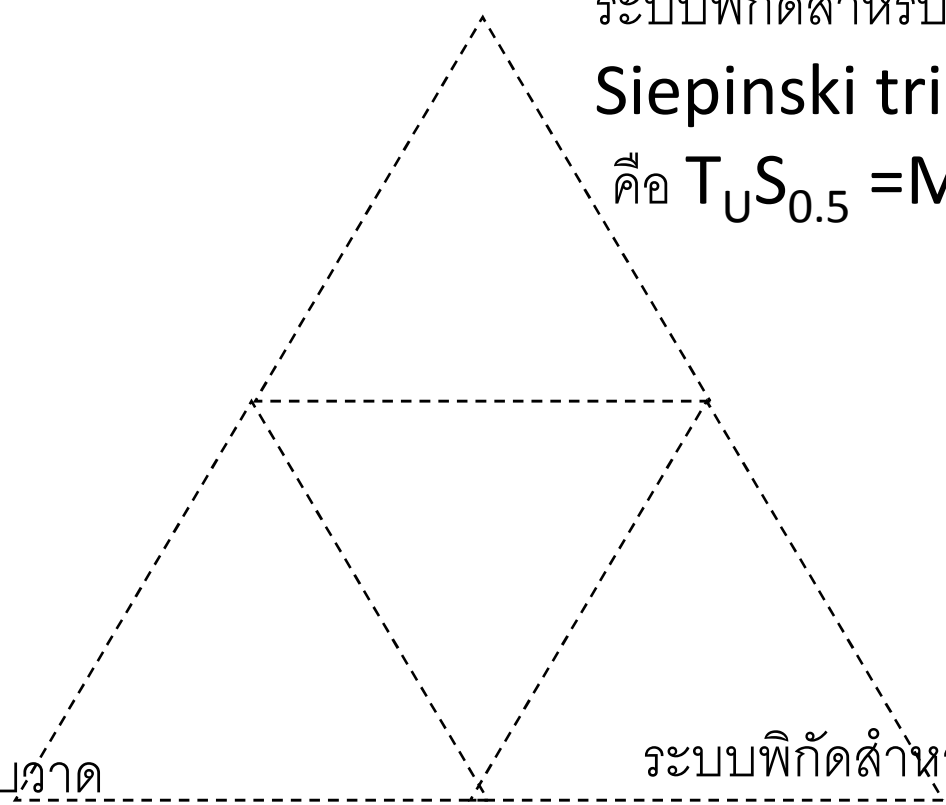


เกิดอะไรขึ้น?



ระบบพิกัดสำหรับวาด Siepinski triangle ระดับ k = I

เกิดอะไรขึ้น?

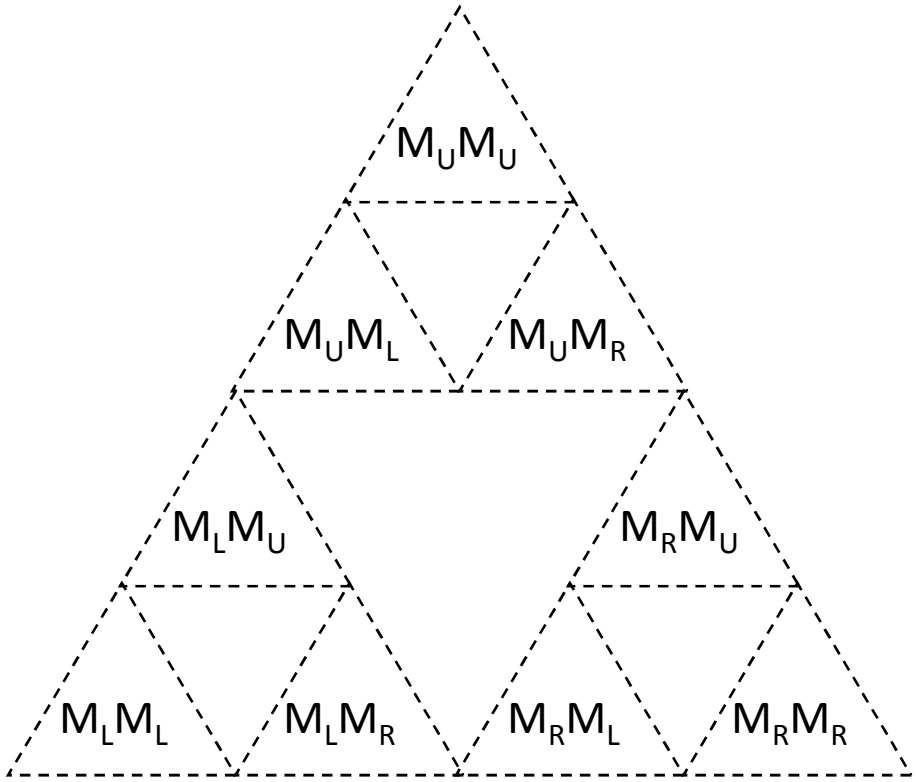


ระบบพิกัดสำหรับวาด Siepinski triangle บน คือ $T_U S_{0.5} = M_U$

ระบบพิกัดสำหรับวาด Siepinski triangle ซ้าย คือ $T_L S_{0.5} = M_L$

ระบบพิกัดสำหรับวาด Siepinski triangle ขวา คือ $T_R S_{0.5} = M_R$

เกิดอะไรขึ้น?



พิจารณา code ใหม่อีกครั้ง

```
                                    MODELVIEW = M
glLoadIdentity();                   MODELVIEW = I
glTranslated(...);                  MODELVIEW = T
glScaled(...);                      MODELVIEW = TS
draw_sierpinski(k-1);

                                    MODELVIEW = ???
glLoadIdentity();                   MODELVIEW = I
glTranslated(...);                  MODELVIEW = T
glScaled(...);                      MODELVIEW = TS
draw_sierpinski(k-1);
```

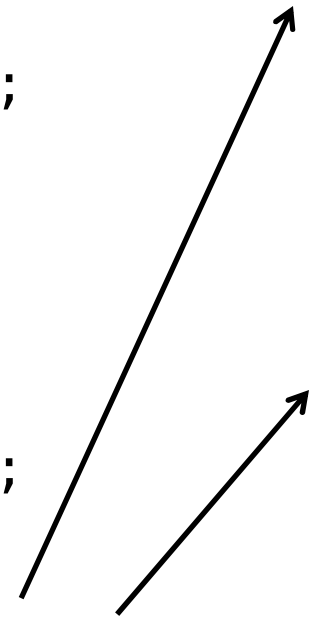
พิจารณา code ใหม่อีกครั้ง

```
glLoadIdentity();
glTranslated(...);
glScaled(...);
draw_sierpinski(k-1);

glLoadIdentity();
glTranslated(...);
glScaled(...);
draw_sierpinski(k-1);
```

```
MODELVIEW = M
MODELVIEW = I
MODELVIEW = T
MODELVIEW = TS

MODELVIEW = ???
MODELVIEW = I
MODELVIEW = T
MODELVIEW = TS
```



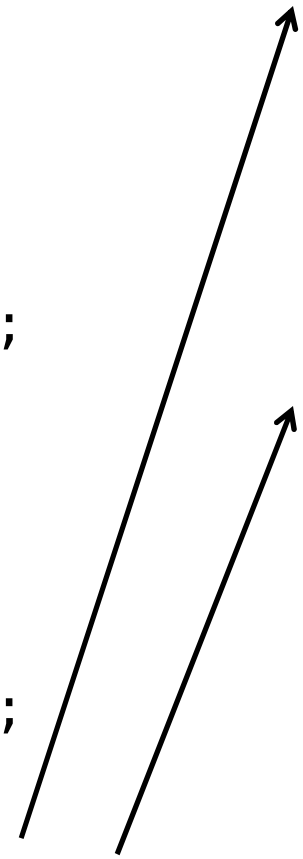
ความจริงแล้ว MODELVIEW ควรมีค่าเท่ากับ MTS!!!

พิจารณา code ใหม่อีกครั้ง

```
glLoadIdentity();
glTranslated(...);
glScaled(...);
draw_sierpinski(k-1);

glLoadIdentity();
glTranslated(...);
glScaled(...);
draw_sierpinski(k-1);
```

```
MODELVIEW = M
MODELVIEW = I
MODELVIEW = T
MODELVIEW = TS
MODELVIEW = ???
MODELVIEW = ???
MODELVIEW = I
MODELVIEW = T
MODELVIEW = TS
MODELVIEW = ???
```



ตรงจุดสองจุดนี้ MODELVIEW ควรมีค่าเท่ากับ M

แล้วจะต้องทำอะไร?

- ก่อนสั่ง glTranslated(...) ต้องมีการจำค่าเมตริกซ์ MODELVIEW อันเดิมเอาไว้
- หลังเรียก draw_sierpinski(...) ต้องมีการเอาค่า MODELVIEW อันเดิมคืนกลับมา

glPushMatrix() และ glPopMatrix()

- glPushMatrix()
  - ทำการ push ค่าของเมตริกซ์ใน mode ปัจจุบันลง stack
- glPopMatrix()
  - pop stack ที่เก็บค่าเมตริกซ์เอาไว้แล้วนำค่าที่ได้ไปให้

- เราสามารถใช้ฟังก์ชันสองฟังก์ชันนี้ในการ “จำ” transform ได้

เขียนใหม่

```
                                MODELVIEW = M
glPushMatrix();                 MODELVIEW = M (จำ)
glTranslated(...);              MODELVIEW = MT
glScaled(...);                  MODELVIEW = MTS
draw_sierpinski(k-1);           MODELVIEW = ???
glPopMatrix();                  MODELVIEW = M

glPushMatrix();                 MODELVIEW = M (จำ)
glTranslated(...);              MODELVIEW = MT
glScaled(...);                  MODELVIEW = MTS
draw_sierpinski(k-1);           MODELVIEW = ???
glPopMatrix();                  MODELVIEW = M
```

ทั้งฟังก์ชัน

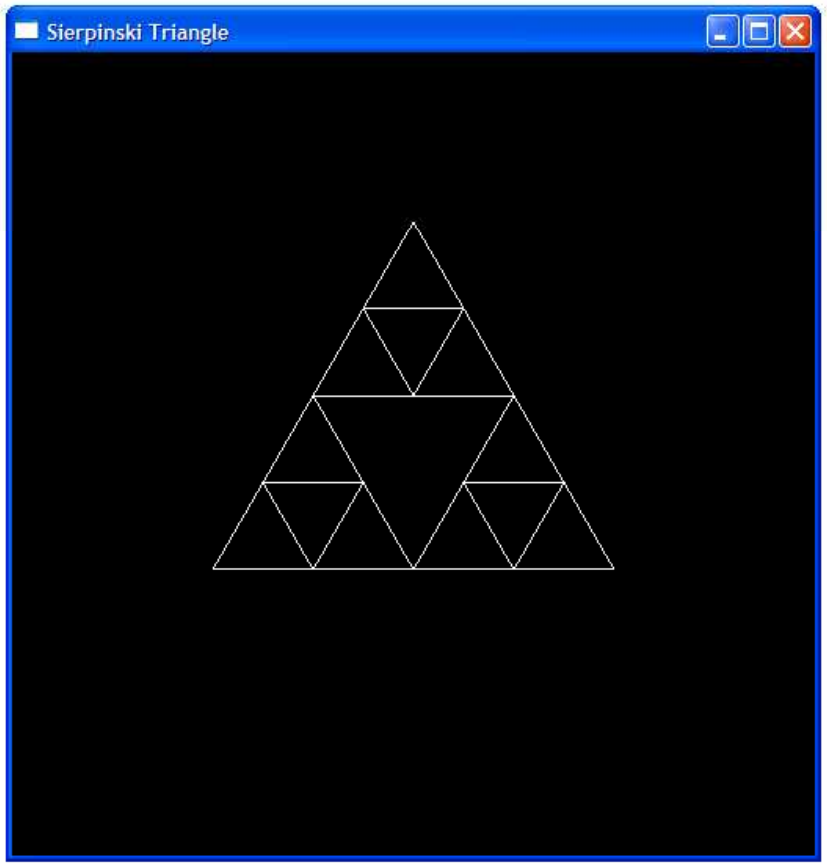
```
void draw_sierpinski(int k)
{
    if (k == 0)
          draw_triangle();
    else
    {
          glPushMatrix();
          glTranslated(0.0, 0.5 / sqrt(3.0), 0.0);
          glScaled(0.5, 0.5, 0.5);
          draw_sierpinski(k-1);
          glPopMatrix();

          glPushMatrix();
          glTranslated(0.25, -0.25 / sqrt(3.0), 0.0);
          glScaled(0.5, 0.5, 0.5);
          draw_sierpinski(k-1);
          glPopMatrix();

          glPushMatrix();
          glTranslated(-0.25, -0.25 / sqrt(3.0), 0.0);
          glScaled(0.5, 0.5, 0.5);
          draw_sierpinski(k-1);
          glPopMatrix();
    }
}
```

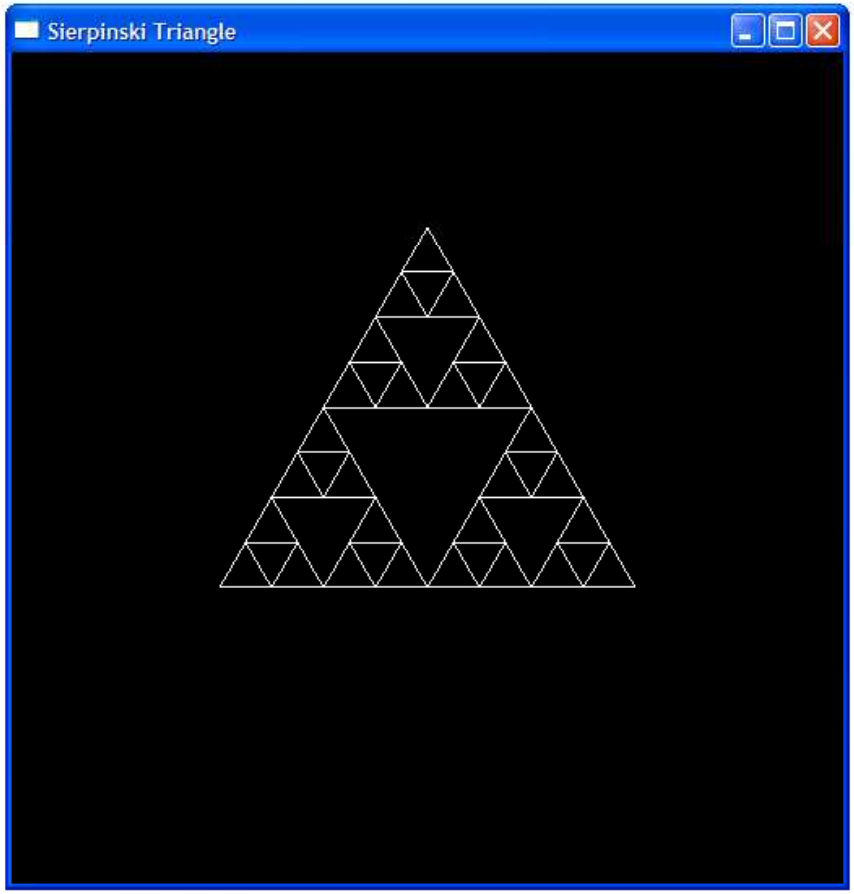
ผลลัพธ์

- draw_sierpinski(2)



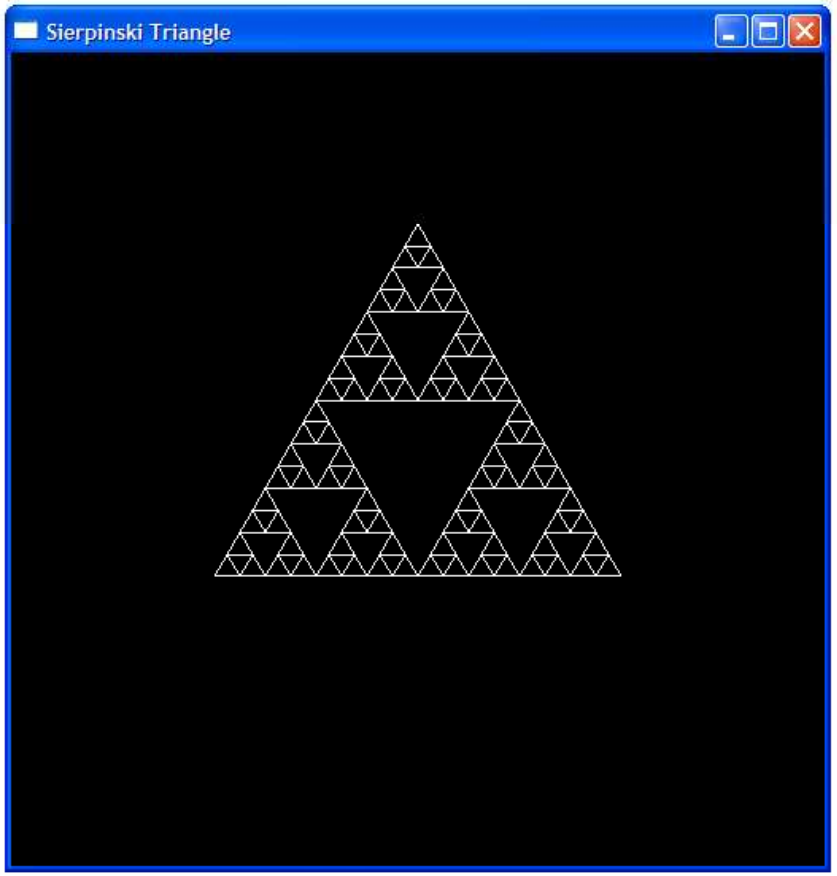
ผลลัพธ์

- draw_sierpinski(3)



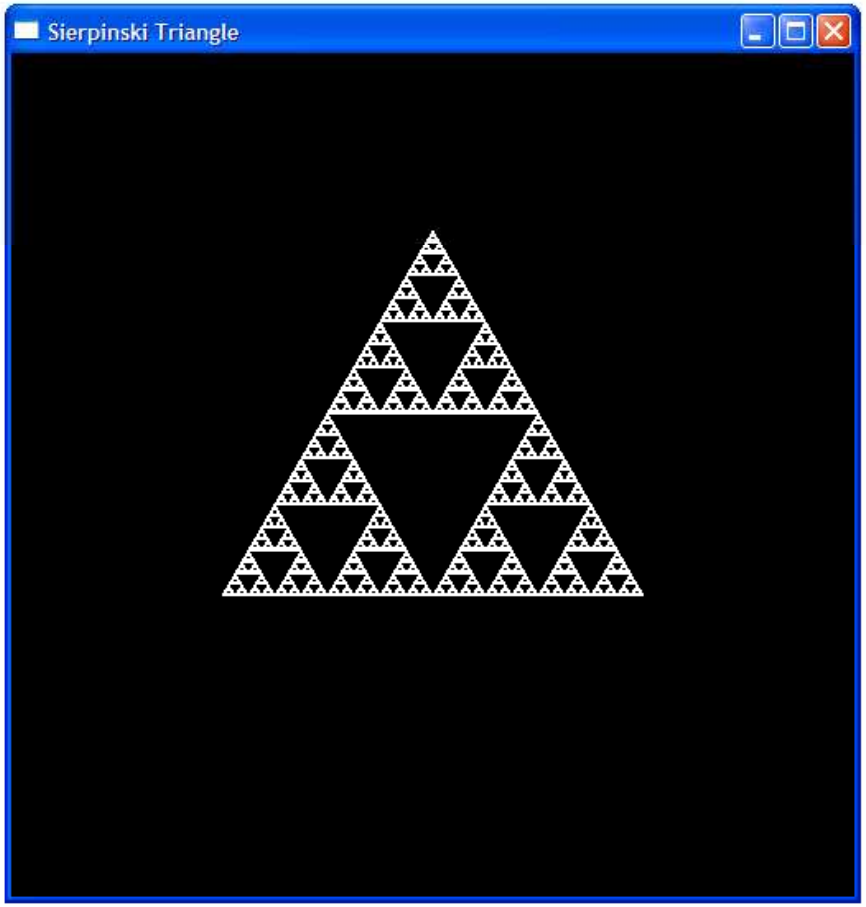
ผลลัพธ์

- draw_sierpinski(4)



ผลลัพธ์

- draw_sierpinski(8)

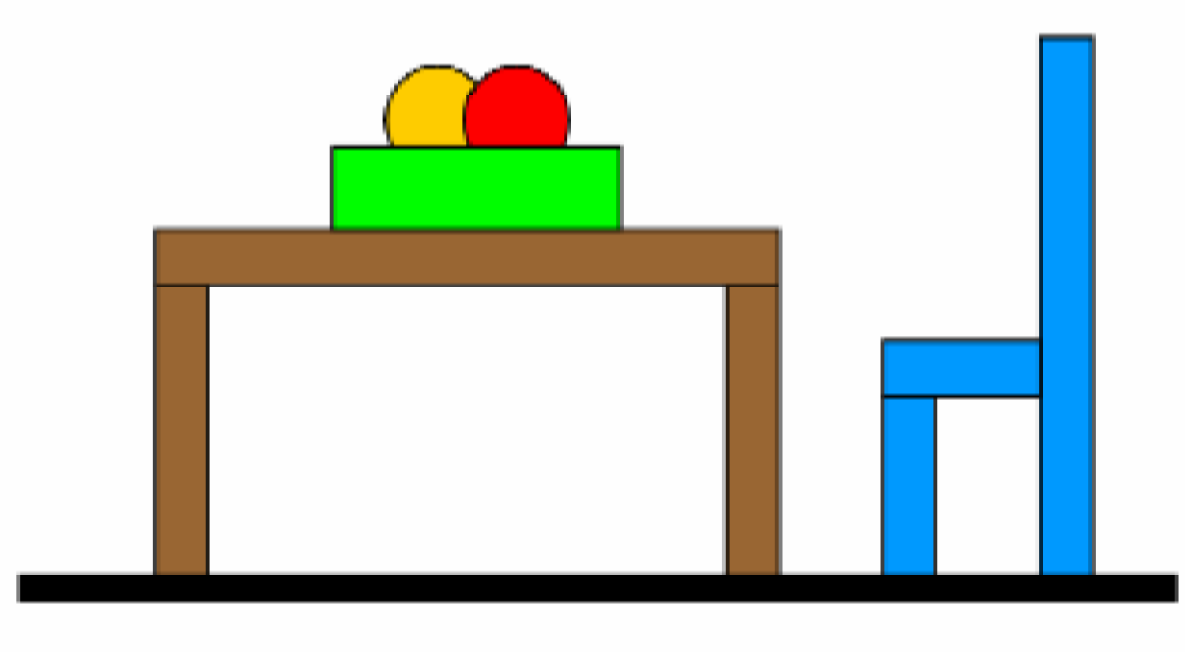


HIERARCHICAL SCENE ORGANIZATION

การจัดฉาก

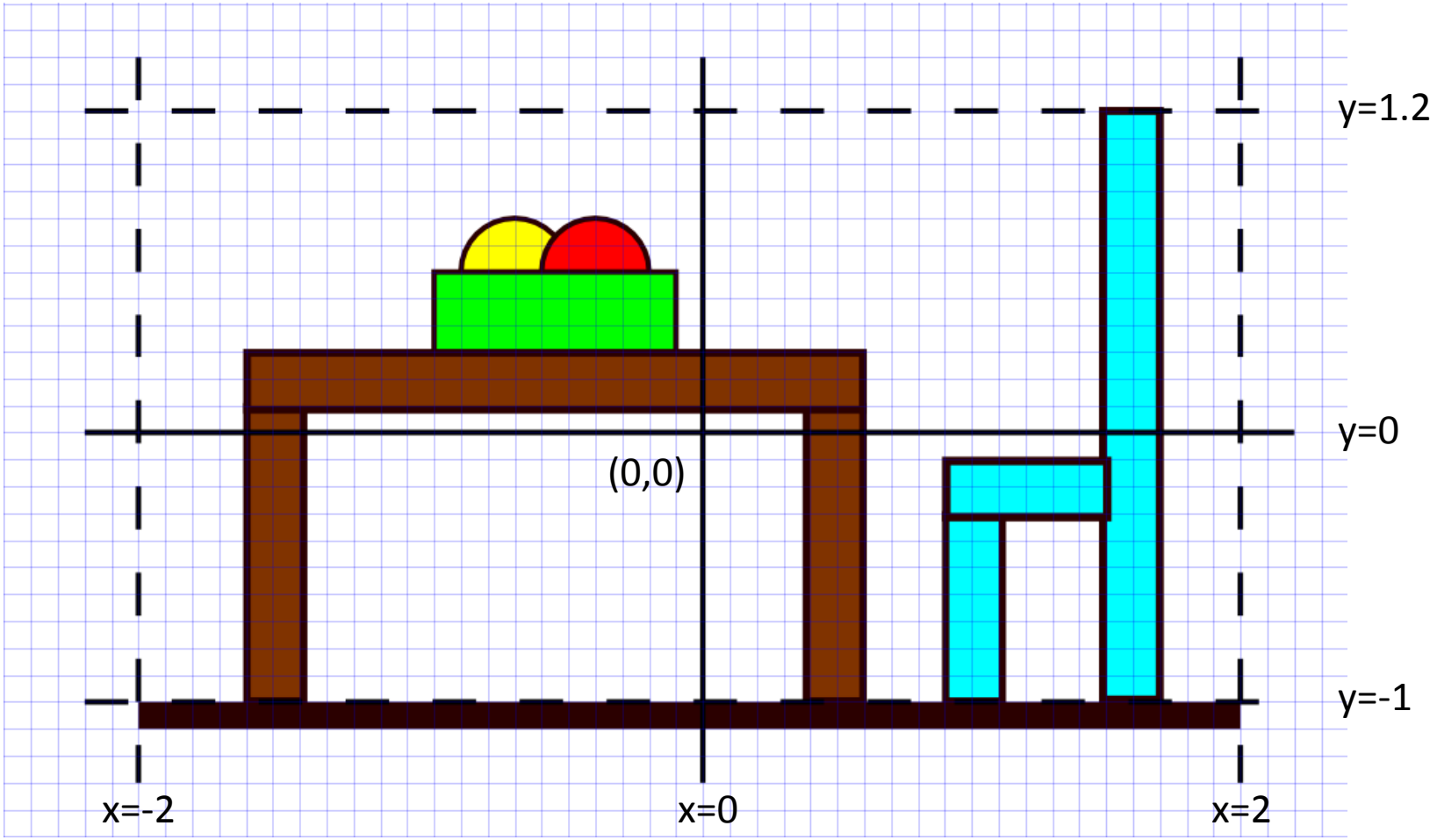
- ฉากประกอบด้วยวัตถุหลายๆ อย่าง
- ศิลปินสร้างวัตถุแต่ละชิ้นขึ้นมาใน object space ของมันเอง
- วัตถุแต่ละวัตถุจะต้องถูกแปลงจากที่อยู่ใน object space ให้มาอยู่ใน world space

ตัวอย่างฉาก



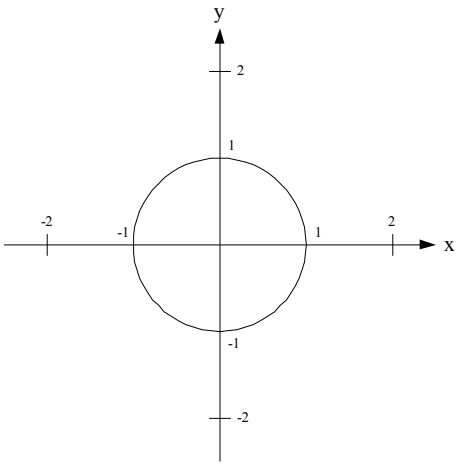
ภาพจาก Durand and Cutler, 6.837 Fall 2003 Lecture Note: Transformations 2: In Modeling

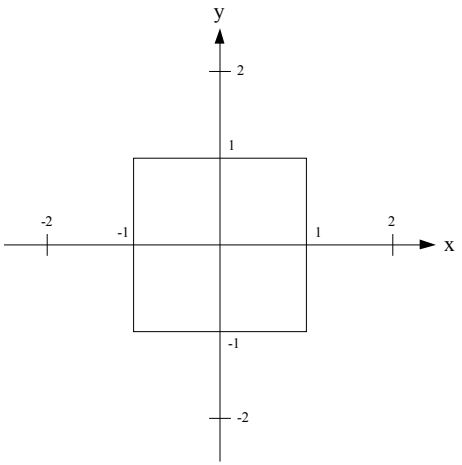
ฉากตัวอย่าง



วัตถุ

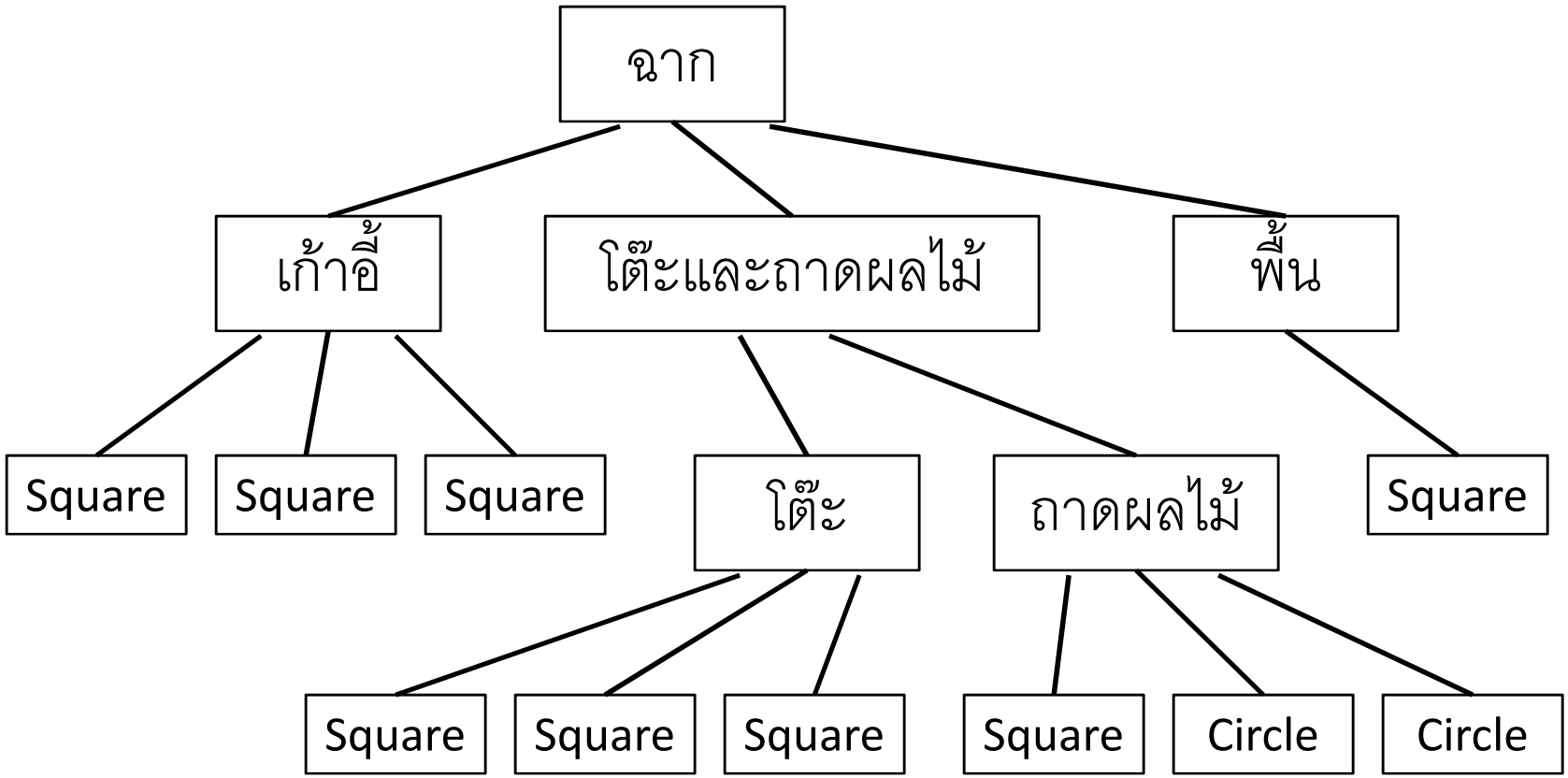
- สมมติว่าศิลปินสร้างวัตถุให้เรามาสองอย่าง
  - Circle: วงกลมรัศมีหนึ่งหน่วยที่มีจุดศูนย์กลางอยู่ที่จุด (0,0)
  - Square: สี่เหลี่ยมจัตุรัสที่มีจุดมุมล่างซ้ายอยู่ที่จุด (-1,-1) และมุมบนขวาอยู่ที่ (1,1)
- เราจะสร้างฉากที่เห็นในสไลด์ที่แล้วอย่างไร?





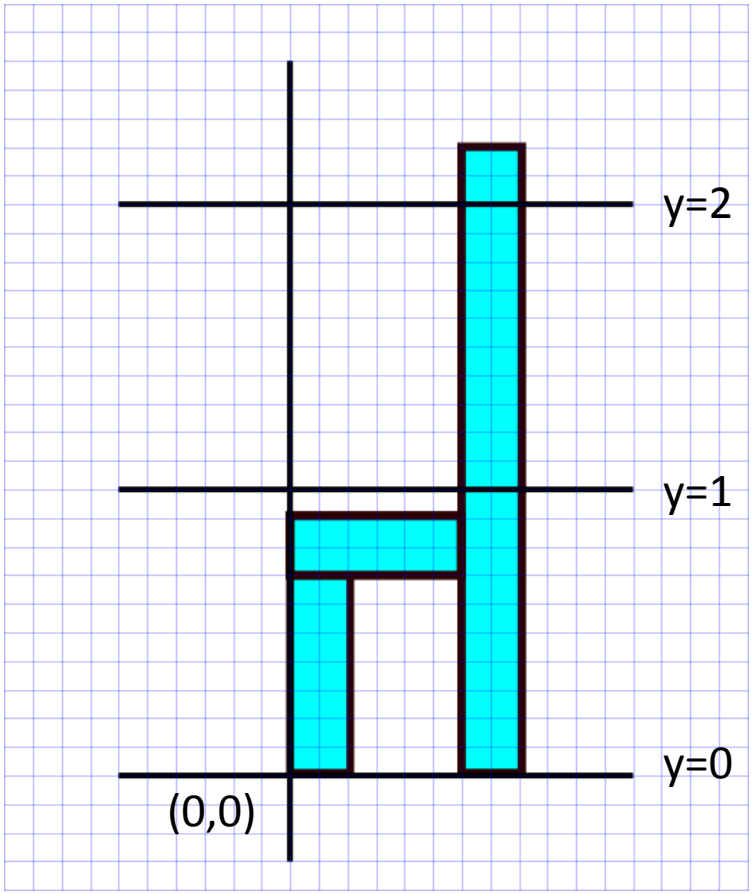
การจัดฉากแบบเป็นลำดับขั้น

- เวลาจัดฉากเรามักจะแบ่งมันเป็นลำดับขั้น



วาดเก้าอี้

- เราเลือก object space ของเก้าอี้ให้มุมล่างซ้ายของมันอยู่ที่จุด (0,0)

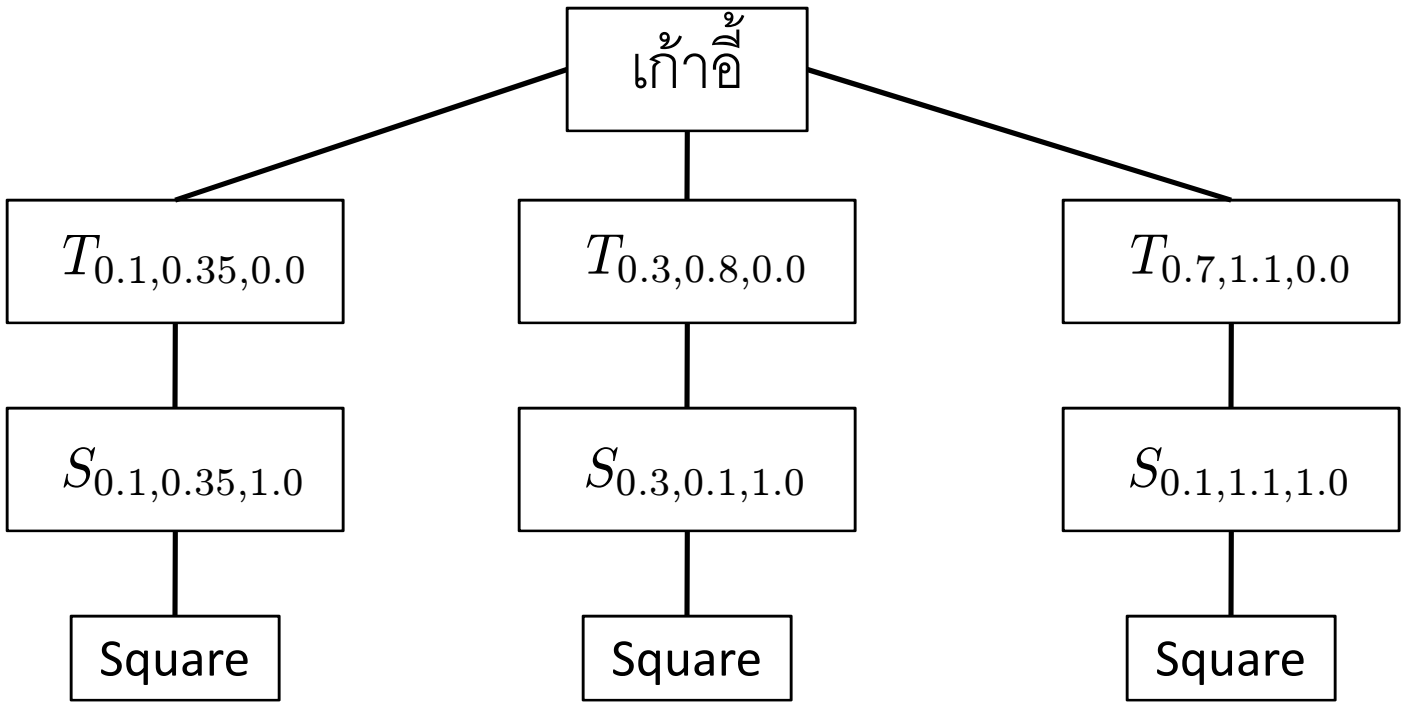


วาดเก้าอี้

- ขาหน้า
  - ย้ายจุดศูนย์กลางไปอยู่ที่ (0.1, 0.35)
  - ขยายตามแกน x = 0.1 เท่า ตามแกน y = 0.35 เท่า
  - วาด Square
- พื้น
  - ย้ายจุดศูนย์กลางไปอยู่ที่ (0.3, 0.8)
  - ขยายตามแกน x = 0.3 เท่า ตามแกน y = 0.1 เท่า
  - วาด Square
- ขาหลังและพนัก
  - ย้ายจุดศูนย์กลางไปอยู่ที่ (0.7, 1.1)
  - ขยายตามแกน x = 0.1 เท่า ตามแกน y = 1.1 เท่า
  - วาด Square

Scene Graph

- เราสามารถแทนการแปลงและการวาดภาพในสไลด์ที่แล้วได้ด้วยแผนภาพที่เรียกว่า scene graph



โค้ด

- เราสามารถเปลี่ยน scene graph เป็นโค้ดได้อย่างง่ายดาย

```
void draw_chair()
{
    glPushMatrix();
    glTranslated(0.1, 0.35, 0.0);
    glScaled(0.1, 0.35, 1.0);
    draw_square(...);
    glPopMatrix();

    glPushMatrix();
    glTranslated(0.3, 0.8, 0.0);
    glScaled(0.3, 0.1, 1.0);
    draw_square(...);
    glPopMatrix();

    glPushMatrix();
    glTranslated(0.7, 1.1, 0.0);
    glScaled(0.1, 1.1, 1.0);
    draw_square(...);
    glPopMatrix();
}
```

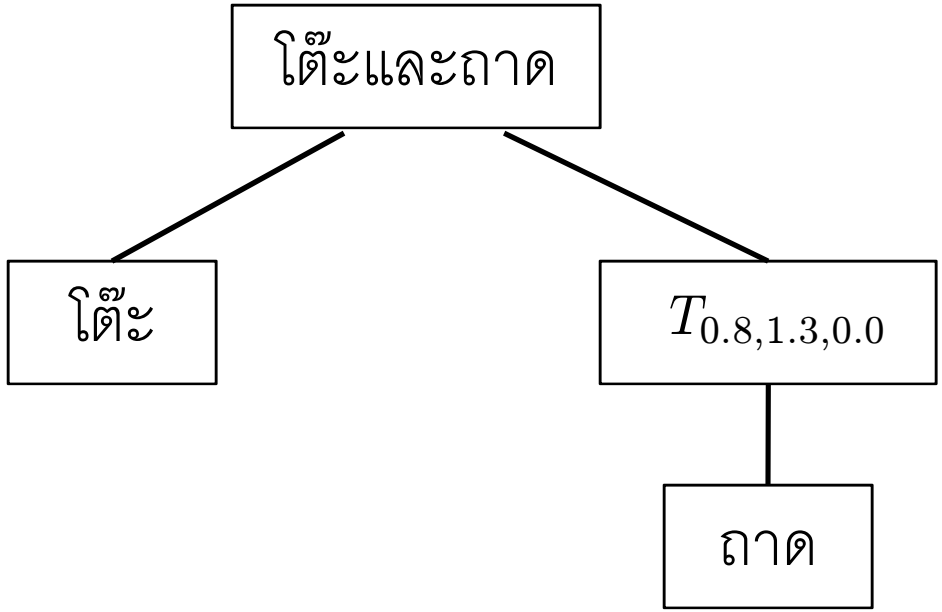
วาดส่วนประกอบอื่นๆ ของฉาก

- สมมติว่าเราสร้างฟังก์ชัน
  - draw_table() เพื่อวาดโต๊ะ โดยที่มุมล่างซ้ายของมันอยู่ที่จุด (0,0)
  - draw_tray() เพื่อวาดถาดผลไม้ โดยที่มุมล่างซ้ายของมันอยู่จุด (0,0)
  - draw_floor() เพื่อวาดพื้น
- ฟังก์ชันพวกนี้สามารถสร้างได้เหมือน draw_chair()
- โค้ดจริงๆ ไปดูได้ในโค้ดตัวอย่าง

วาดส่วนประกอบอื่นๆ ของฉาก

- เราสามารถวาดโต๊ะและถาดผลไม้ได้ดังต่อไปนี้
  - วาดโต๊ะโดยการเรียก draw_table()
  - วาดถาดผลไม้
    - ย้ายจุดมุมซ้ายไปอยู่ที่จุด (0.8, 1.3)
    - แล้วเรียก draw_tray()

Scene Graph ของโต๊ะและถาด

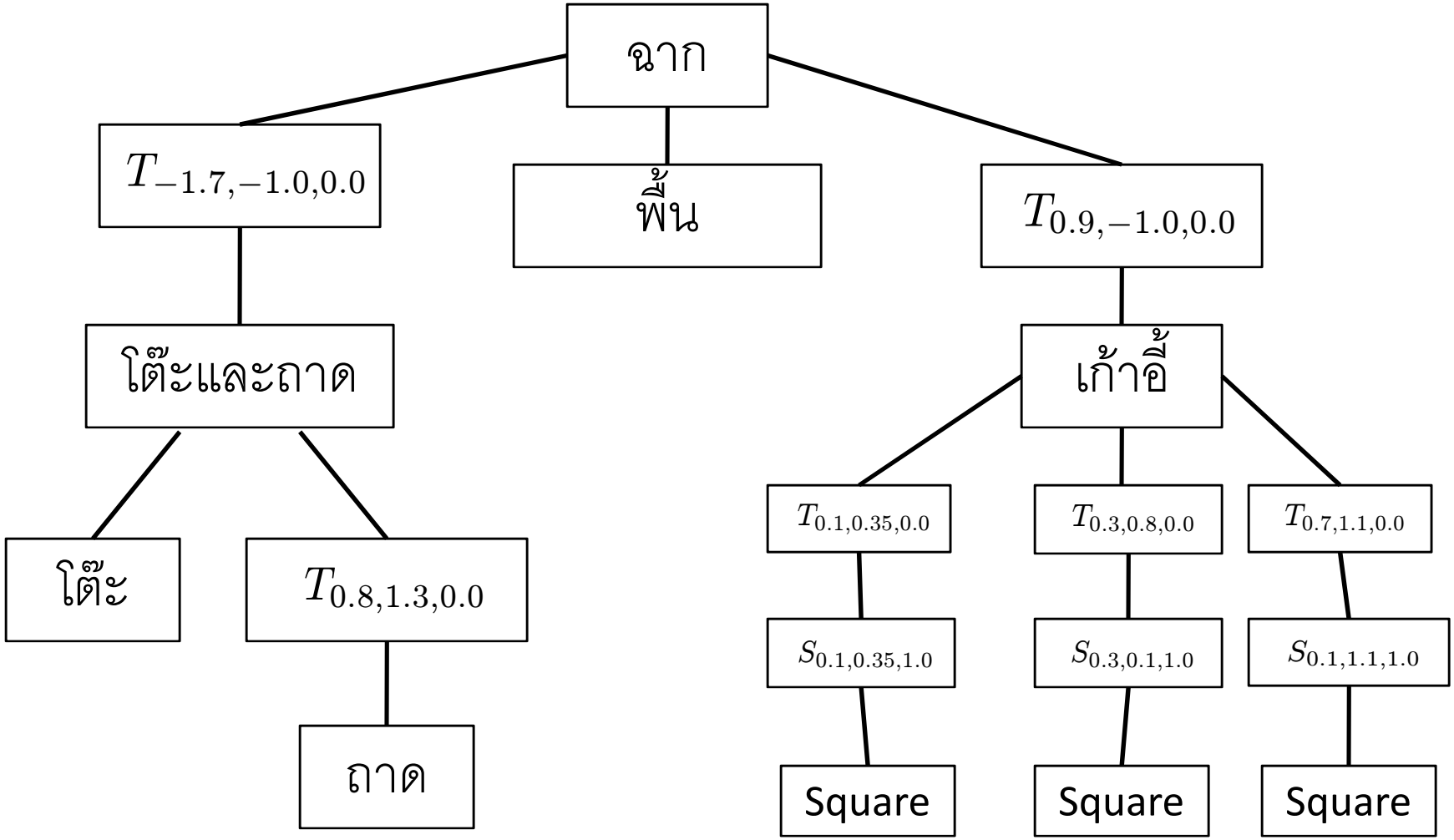


โค้ด

```
void draw_table_and_tray()
{
  draw_table();

  glPushMatrix();
  glTranslated(0.8, 1.3, 0.0);
  draw_tray();
  glPopMatrix();
}
```

Scene Graph ของฉาก



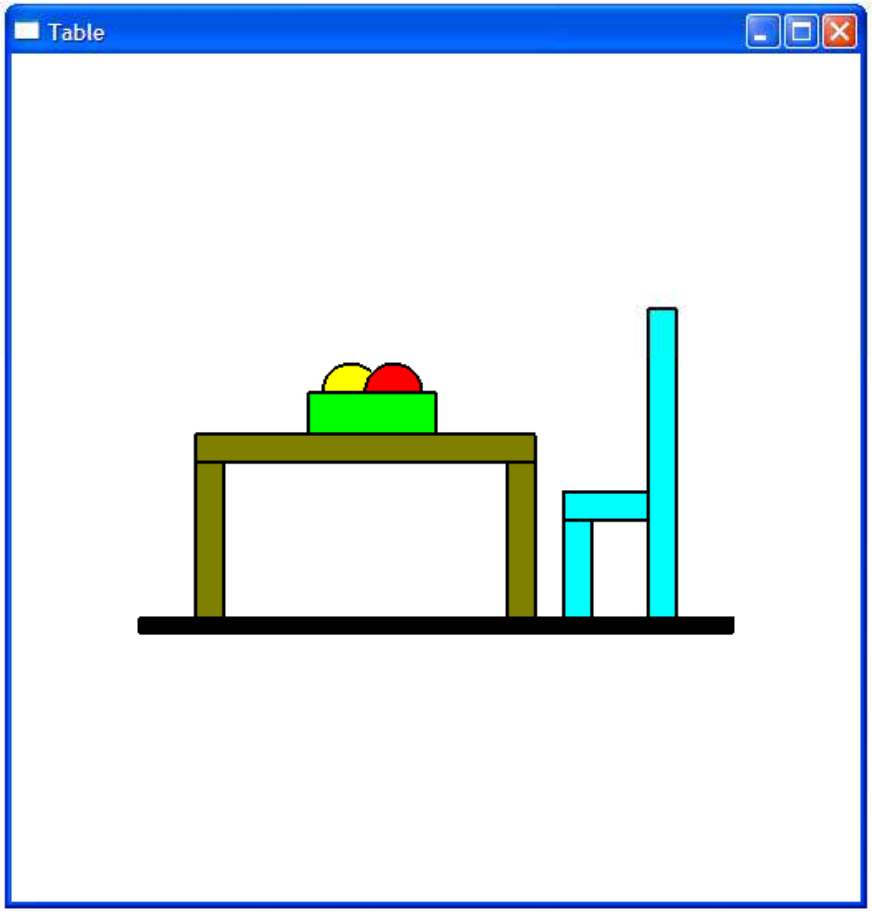
โค้ด

```
void draw_scene()
{
  glPushMatrix();
  glTranslated(-1.7, -1.0, 0.0);
  draw_table_and_tray();
  glPopMatrix();

  draw_floor();

  glPushMatrix();
  glTranslated(0.9, -1.0, 0.0);
  draw_chair();
  glPopMatrix();
}
```

ผลลัพธ์



บทเรียน

- เราสามารถสร้างฉากที่มีความซับซ้อนได้จาก
  - วัตถุง่ายๆ
  - การแปลง
- เพื่อความสะดวกและความเข้าใจง่าย เราสามารถจับกลุ่มวัตถุเป็นกลุ่มๆ แล้วสร้างฉากจากกลุ่มของวัตถุได้
- เราสามารถแทนการจัดฉากได้ด้วยแผนภาพที่เรียกว่า scene graph ซึ่งประกอบด้วย
  - กล่องสำหรับแทนวัตถุ
  - กล่องสำหรับแทนกลุ่มของวัตถุ
  - กล่องสำหรับแทนการแปลง

บทเรียน

- เมื่อเขียน scene graph แล้วเราสามารถเขียนโค้ดเพื่อวาดฉากที่ scene graph บรรยายได้อย่างง่ายดาย
  - กล่องวัตถุหรือกลุ่มของวัตถุ -> ฟังก์ชัน
  - กล่องการแปลง -> การเรียกฟังก์ชันทำการแปลง เช่น glTranslate, glScale หรือ glRotate
  - แขนงของกล่อง -> การเรียก glPushMatrix() แล้วลงไปจัดการแขนงนั้น แล้วจึงเรียก glPopMatrix()