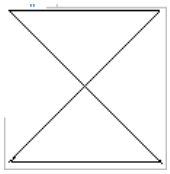
# Лабораторна робота № 1. Графічні примітиви.

Виконав: студент групи ІП-32 Олександр Ковальчук.

## Завдання

1. Побудувати графічний примітив згідно варіанту.
2. Розробити модель зображення M(P1, P2, … PN), де P1 – заданий графічний примітив, P2 … PN – параметри моделі, які визначають конфігурацію графічних примітивів
3. На основі розробленої моделі, змінюючи її параметри побудувати орнамент.

Графічний примітив:



# Виконання

## Лістинг

```
void base_shape_2(float cx, float cy, float w, float h) {
  float x1 = cx - w/2;
  float y1 = cy - h/2;

  float x2 = cx + w/2;
  float y2 = cy + h/2;

  beginShape();
   vertex(x1, y1);
   vertex(x2, y1);
   vertex(x1, y2);
   vertex(x2, y2);
   vertex(x1, y1);
  endShape();
}

void compose(
  int count, float w1, float h1,
  float wn, float hn,
  float rot, float koef
) {
  float dw = (wn - w1) / count;
  float dh = (hn - h1) / count;

  for(int i = 0; i < count; i++) {
    float w = w1 + dw * i;
    float h = h1 + dh * i;

    float posx = (w + max(w1, wn)) / 2;
    pushMatrix();
      translate(posx * koef * i, 0);
      rotate(rot*i);
      base_shape_2(0, 0, w, h);
    popMatrix();
  }
```

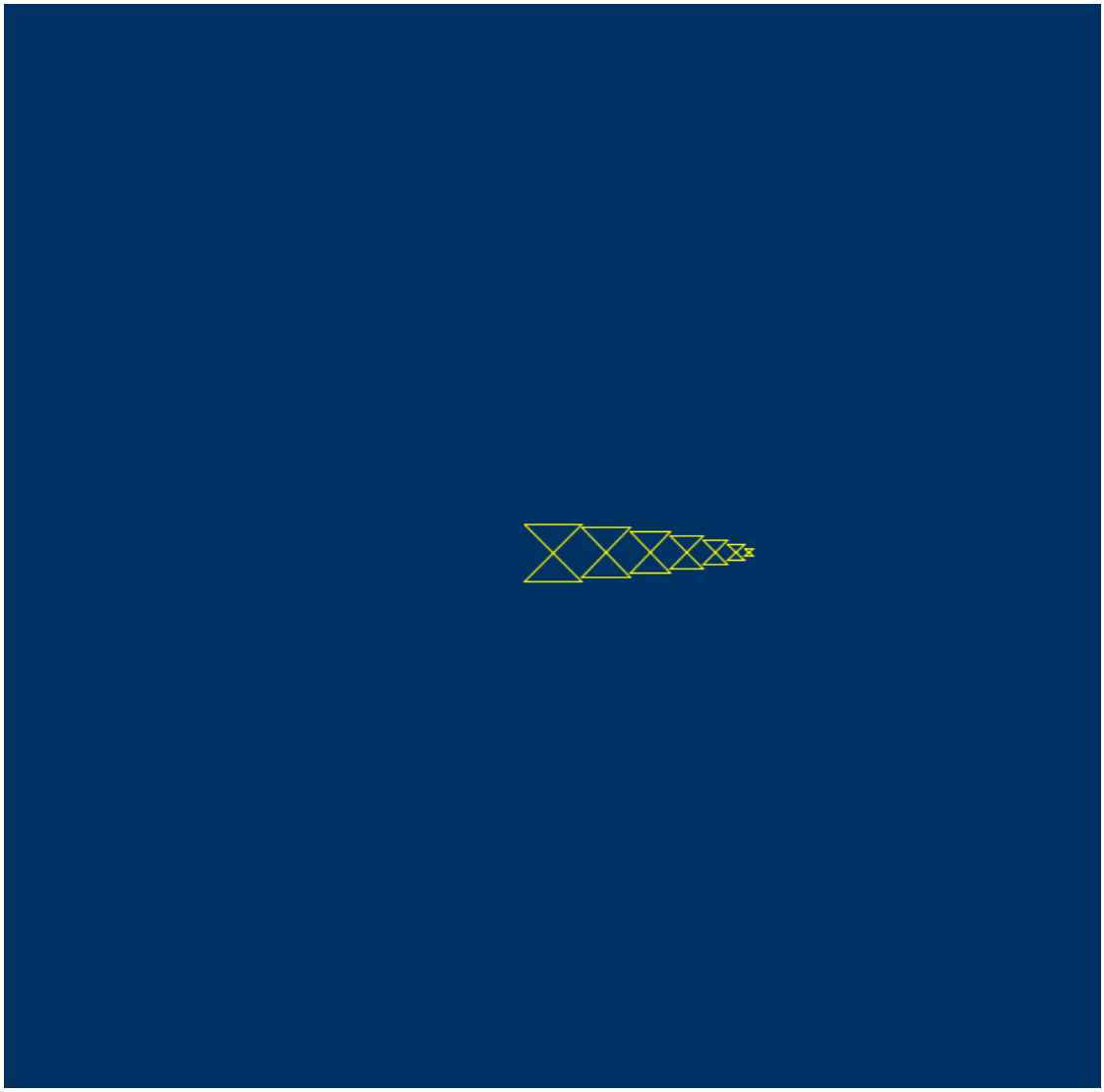
```
}

void pattern(
  int cnt, int cnt_l, int w1, int h1,
  int wn, int hn, float angle,
  float dist_koef,
  int scale_koef,
  boolean rotate
) {
 background(0, 50, 100);
 translate(width*0.5, height*0.5);
 if (scale_koef > 1) {
   scale((frameCount % scale_koef) / (1.0 * scale_koef));
 }
 if (rotate) {
   rotate(frameCount / 200.0);
 }

 for (int i = 0; i < cnt; ++i) {
   float rot = 2*PI / cnt;
   pushMatrix();
     rotate(i * rot);
     compose(cnt_l, w1, h1, wn, hn, angle, dist_koef);
   popMatrix();
 }
}

void setup() {
  size(768, 768);
  stroke(255, 255, 0);
  fill(0, 50, 100);
}

void draw() {
  pattern(12, 10, 40, 40, 0, 0, PI/4, 0.9, 200, true);
}
```



## Пояснення

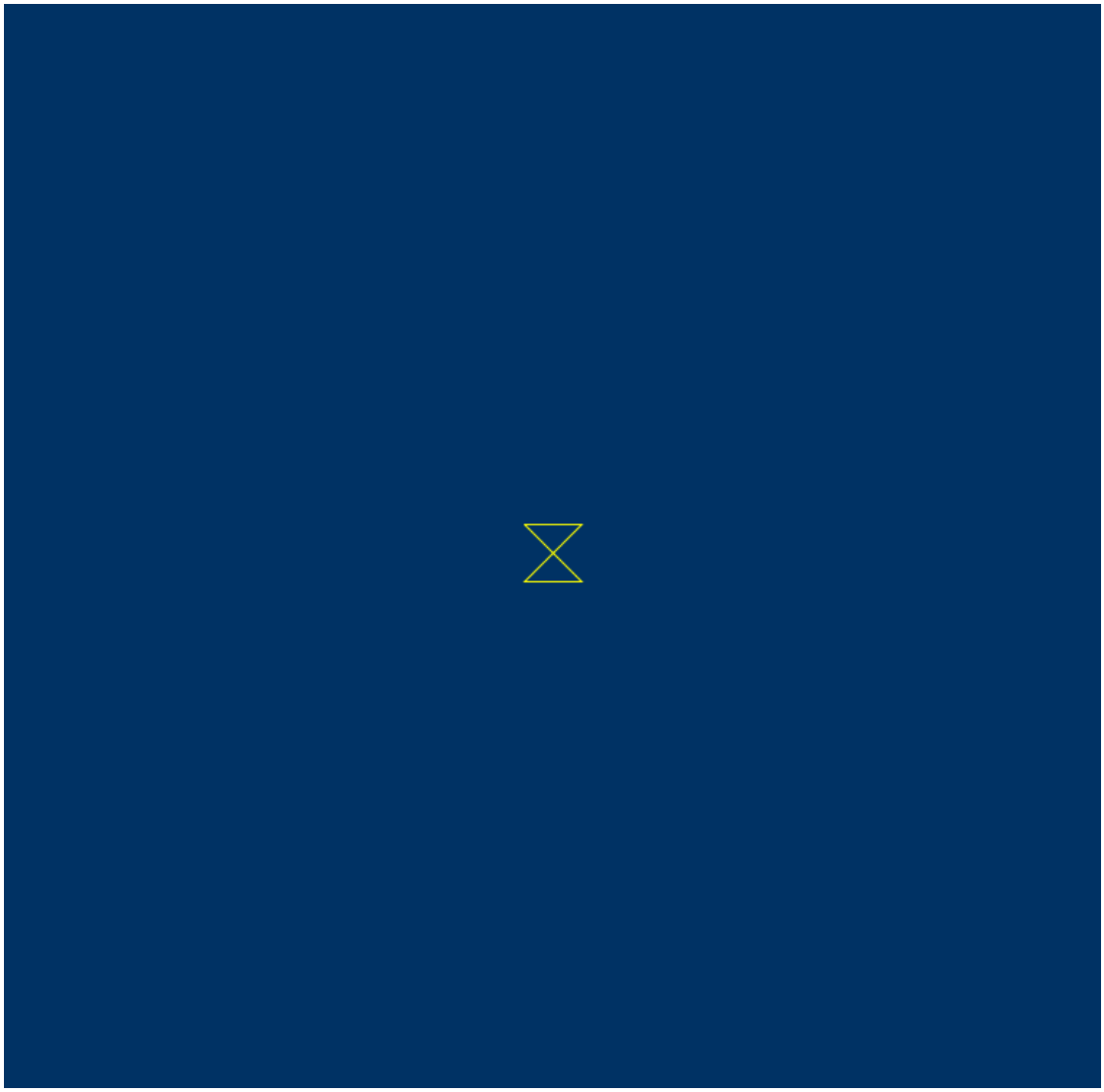
Функція pattern приймає такі аргументи:

- int cnt -- кількість променів, утворених з базової фігури
- int cnt_l -- клькість базових фігур у одному промені
- int w1, -- ширина першої фігури у промені
- int h1 -- висота першої фігури у промені
- int wn -- ширина останньої фігури у промені
- int hn -- висота останньої фігури у промені
- float angle -- кут повороту базової фігури у промені
- float dist_koef -- коефіцієнт відстані між фігурами
- int scale_koef -- коефіцієнт "пульсації" фігур
- boolean rotate -- чи потрібно обертати патерн

## Аналіз

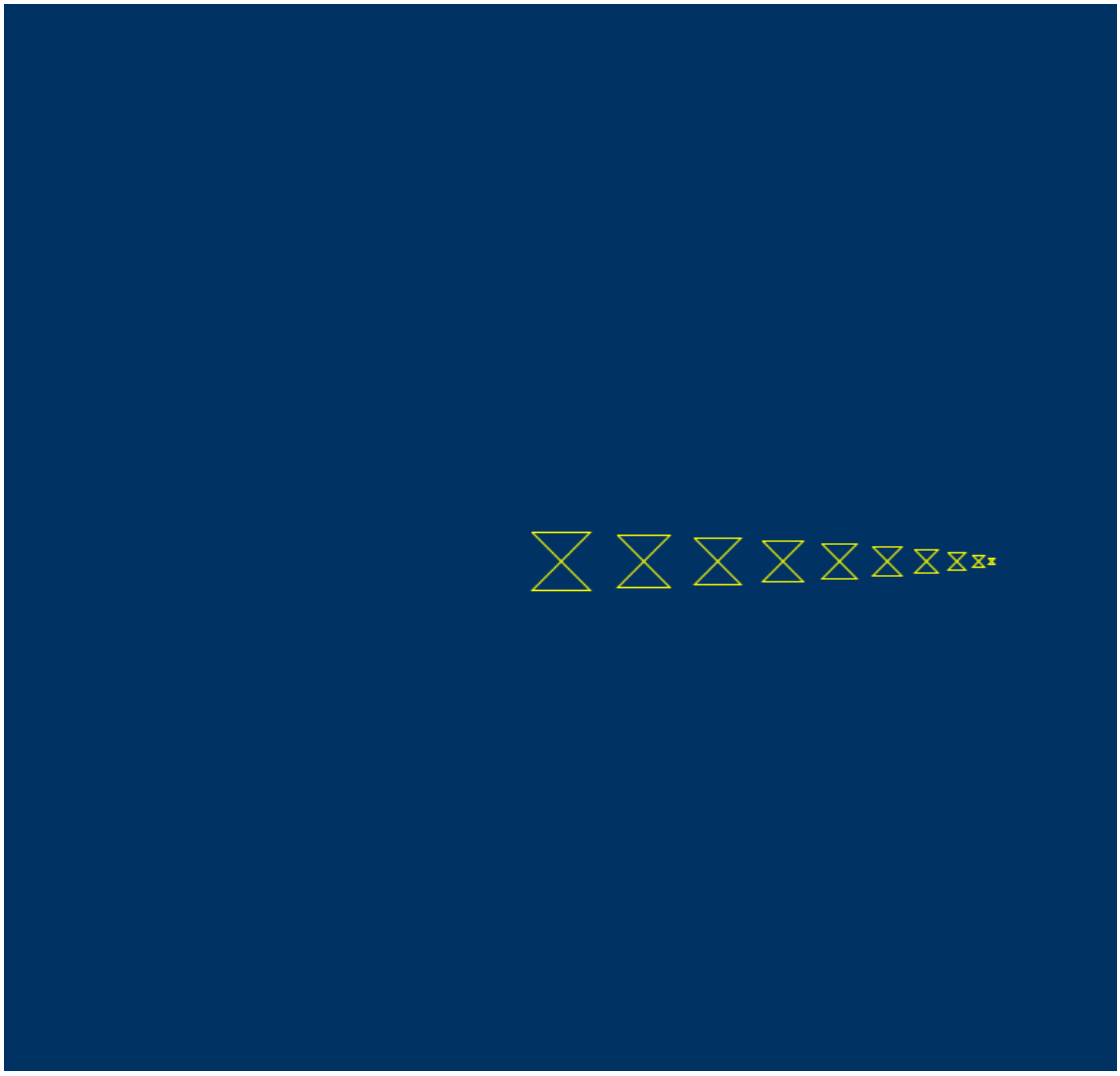
При `cnt = 1`, `cnt_l` = 1, довільних додатніх значеннях `w1`, `wn`, нульових значеннях `wn`, `hn`, `angle`, `dist_koef`, `scale_koef` та значенню `false` для параметра `rotate` отримуємо базову фігуру.

```
pattern(1, 1, 40, 40, 0, 0, 0, 0, 0, false);
```

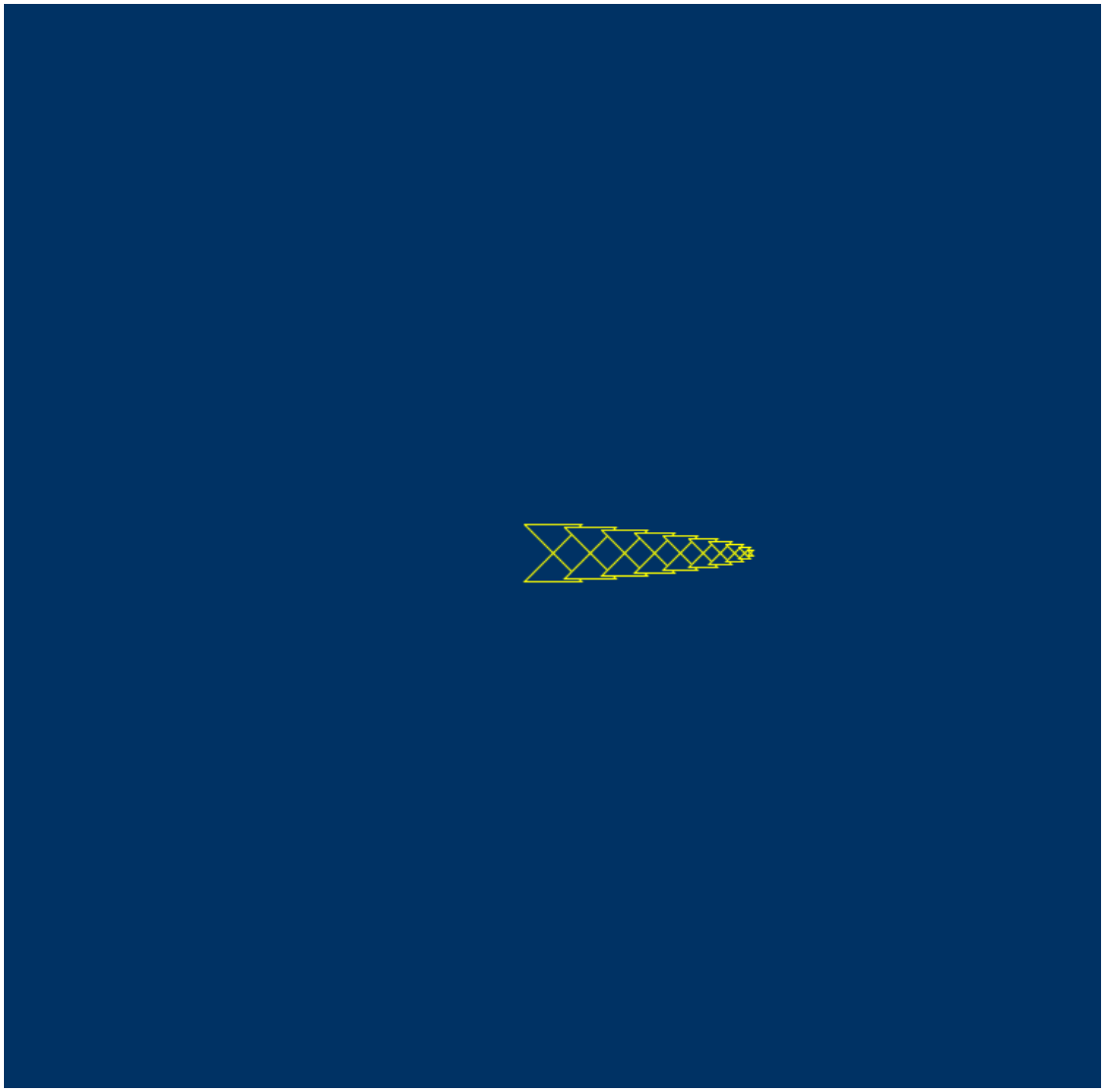


Коефіцієнт `dist_koef` впливає на відстань між елементами у промені. При `dist_koef > 1` -- зі збільшенням коефіцієнта збільшується відстань між елементами. При `0 < dist_koef < 1` спостерігається накладання елементів у промені.

```
    pattern(1, 10, 40, 40, 0, 0, 0, 1.5, 0, false);
```

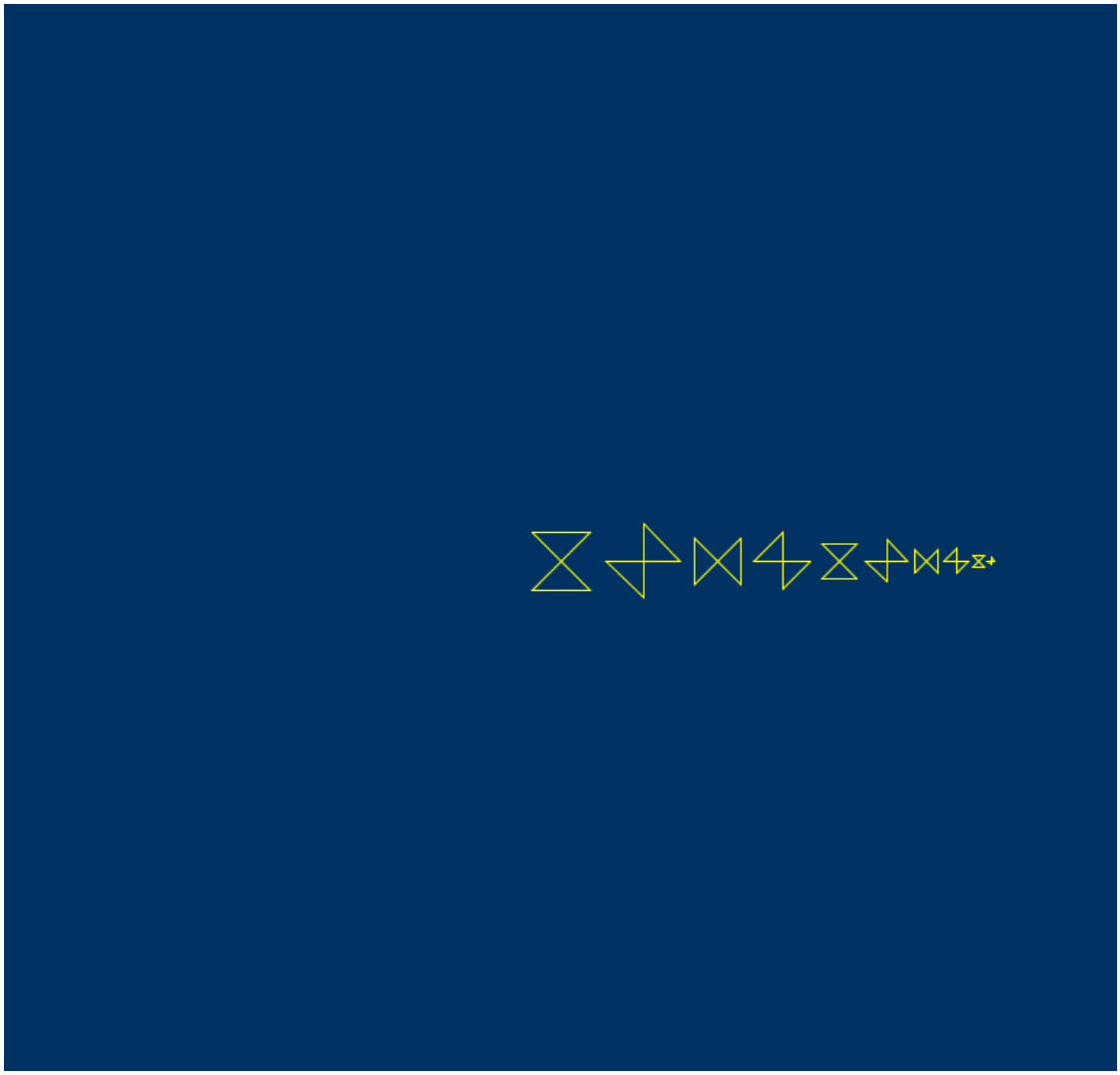


```
pattern(1, 10, 40, 40, 0, 0, 0, 0.5, 0, false);
```



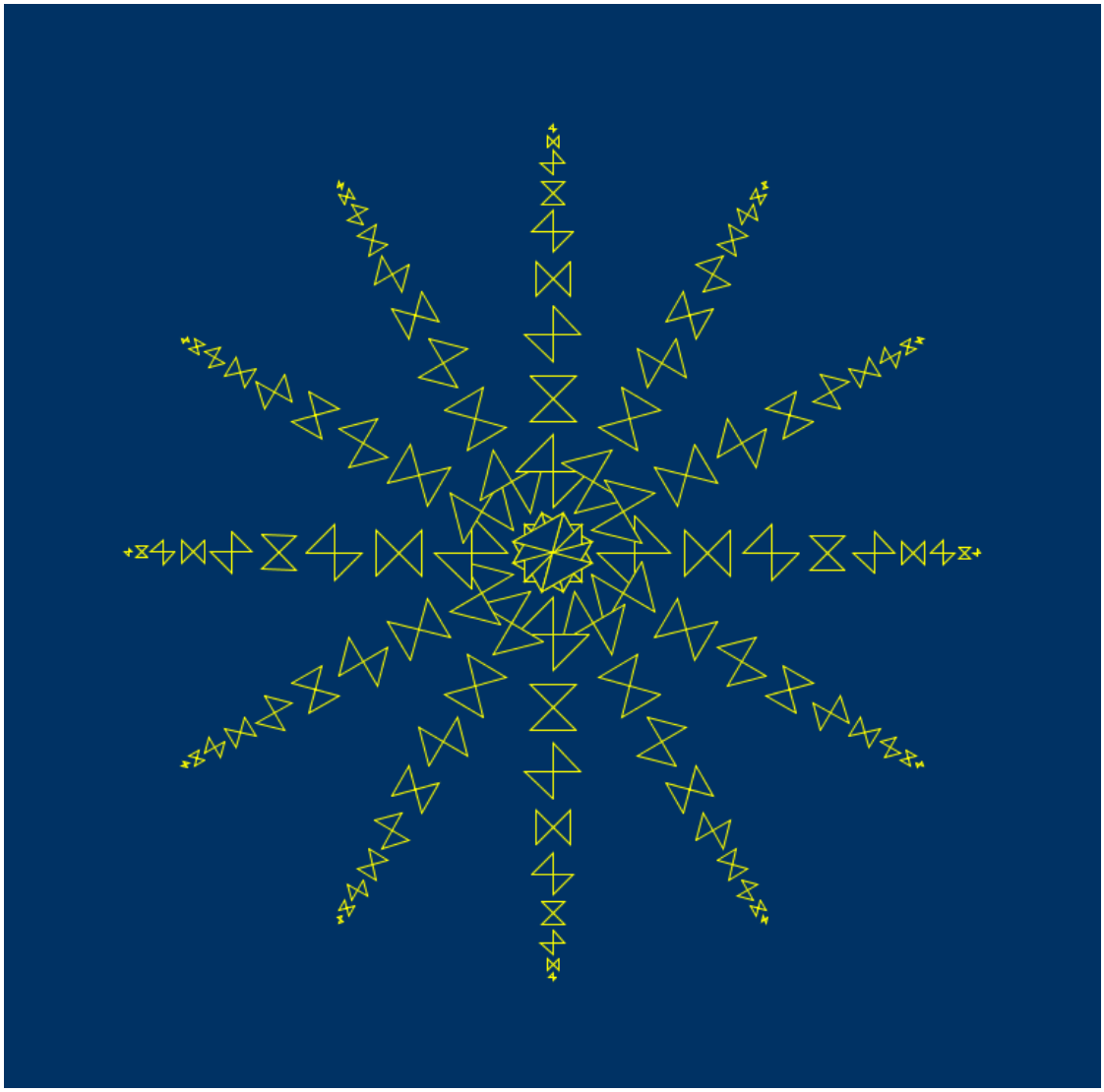
Параметр `angle` відповідає за кут повороту наступного елемента у промені відносно попереднього:

```
    pattern(1, 10, 40, 40, 0, 0, PI/4, 1.5, 0, false);
```



Перший параметр `cnt` відповідає за кількість променів:

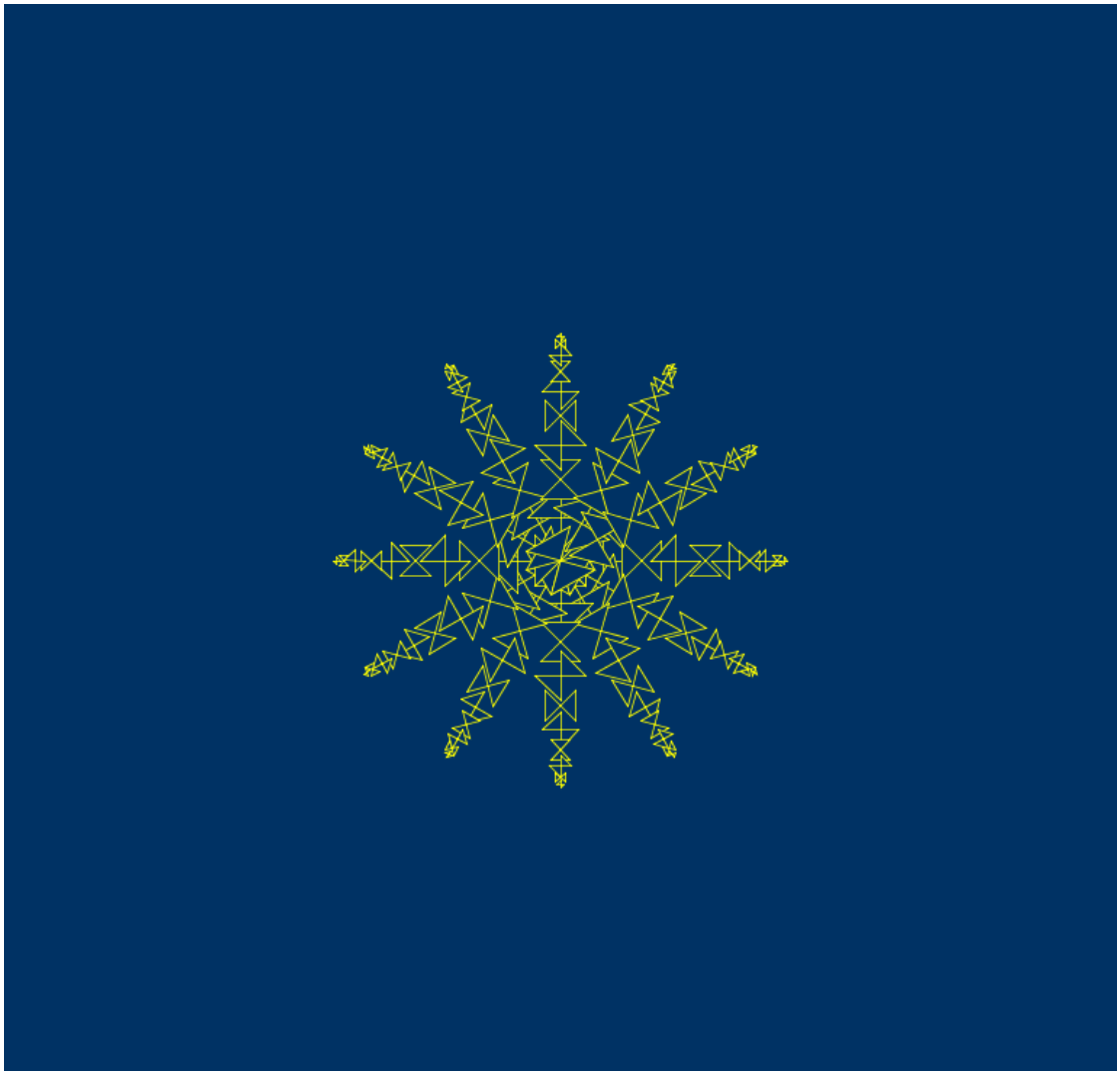
```
    pattern(12, 10, 40, 40, 0, 0, PI/4, 1.5, 0, false);
```



## Фінальне зображення

Фінальне зображення було отримано при таких параметрах:

```
pattern(12, 10, 40, 40, 0, 0, PI/4, 0.9, 200, true);
```



## Висновки

В ході роботи було розроблено та проаналізовано модель згідно варіанту. Аналіз моделі доступний у розділі `Аналіз`. Модель складається з променів базових фігур, які виходять з однієї точки та направлені в різні боки. Базові фігури можуть накладатися (але не перетинатися). Сама модель обертається навколо свого центру та пульсує.