{dnkha, nvtiep}@fit.hcmus.edu.vn

Đồ họa với SVG, GDI+ và CImg – Bài 4

YÊU CẦU 1: Tập thuộc tính của đối tượng

Phương thức thiết lập thuộc tính của lớp `Rect` có thể được viết như sau, nhận tham số là dữ liệu đọc được dạng chuỗi từ tập tin SVG:

```
void Rect::setAttribute(char* attr_name, char* attr_value) {
    if (string(attr_name) == "x") {
        this->x = atoi(attr_value);
        return;
    }

    if (string(attr_name) == "y") {
        this->y = atoi(attr_value);
        return;
    }

    if (string(attr_name) == "width") {
        this->width = atoi(attr_value);
        return;
    }

    if (string(attr_name) == "height") {
        this->height = atoi(attr_value);
        return;
    }

    if (string(attr_name) == "stroke-width") {
        this->strokeWidth = atoi(attr_value);
        return;
    }
    // và nhiều thuộc tính nữa…
}
```

Cách viết như thế này gây ra:

- Sự trùng lắp các mã nguồn chuyển đổi chuỗi sang dữ liệu
- Sự trùng lắp các đoạn lệnh If
- Khi thêm thuộc tính mới sẽ phải sửa lại phương thức `setAttribute`

Hãy tìm cách thiết kế lớp `Attribute/Property` *tổng quát cho các thuộc tính, và cơ chế gắn kết thuộc tính với đối tượng sao cho có thể viết lại phương thức* `setAttribute` ***theo một cách tương tự như gợi ý sau****:*

```
class Rect {
private:
    set<Attribute*> attributes;
    // …
};

void Rect::setAttribute(char* attr_name, char* attr_value) {
    int size = this->attributes.size();
    for (int i = 0; i < size; i++) {
        if (string(attr_name) == this->attributes[i]->getName()) {
            this->attributes[i]->setValue(attr_value);
            return;
        }
    }
}
```

YÊU CẦU 2: Khởi tạo các đối tượng

Mã nguồn tạo các đối tượng đồ họa khi đọc được dữ liệu từ tập tin SVG (dùng RapidXML) có thể được viết như sau:

```
do {
    if (string(node->name()) == "rect") {
        Figure *r = new Rect();
        for (xml_attribute<> *attr = node->first_attribute();
                attr; attr = attr->next_attribute()) {
            r->setAttribute(attr->name(), attr->value());
        }
        this->figures.push_back(r);
    }

    if (string(node->name()) == "circle") {
        Figure* cir = new Circle();
        for (xml_attribute<> *attr = node->first_attribute();
                attr; attr = attr->next_attribute()) {
```

```
            cir->setAttribute(attr->name(), attr->value());
        }
        this->figures.push_back(cir);
    }

    if (string(node->name()) == "ellipse") {
        Figure* ellipse = new Ellipse();
        for (xml_attribute<> *attr = node->first_attribute();
                attr; attr = attr->next_attribute()) {
            ellipse->setAttribute(attr->name(), attr->value());
        }
        this->figures.push_back(ellipse);
    }

    if (string(node->name()) == "line") {
        Figure* line = new Line();
        for (xml_attribute<> *attr = node->first_attribute();
                attr; attr = attr->next_attribute()) {
            line->setAttribute(attr->name(), attr->value());
        }
        this->figures.push_back(line);
    }

    // … tương tự cho các loại đối tượng khác
    node = node->next_sibling();
}
while (node);
```

Cách viết như thế này gây ra:

- Sự trùng lắp các mã nguồn tạo đối tượng và đọc thuộc tính từ tập tin.
- Sự trùng lắp các đoạn lệnh If và mã hóa cứng tên của lớp trong mệnh đề If.
- Khi thêm loại đối tượng mới phải sửa lại mã nguồn.

Hãy tham khảo Giáo trình Phương pháp Lập trình Hướng đối tượng (2010) của Khoa CNTT, tìm hiểu cách tạo đối tượng bằng cách sao chép (Chương 4, mục V) và giải quyết vấn đề trùng lắp mã nguồn trên.

{dnkha, nvtiep}@fit.hcmus.edu.vn

THÁCH THỨC MỞ RỘNG

Có 3 thách thức mở rộng là:

1. thêm một loại thuộc tính mới vào các lớp
2. thêm một loại đối tượng hình học mới
3. thêm cấu trúc phức hợp vào hệ thống đối tượng hình học.

YÊU CẦU 3: Hãy bổ sung và hiện thực hóa thuộc tính transform cho tất cả các đối tượng hình học. Thuộc tính viết trong SVG có dạng transform="<content>". Thuộc tính dùng để thể hiện phép biến hình được sử dụng cho đối tượng, có 3 phép biến hình là phép tịnh tiến (translate), phép xoay (rotate) và phép thay đổi tỉ lệ (scale).

Phép tịnh tiến viết trong SVG có dạng translate(x,y), dịch chuyển trục tọa độ đi x theo trục hoành, y theo trục tung. Phép xoay trong SVG có dạng rotate(d), xoay trục tọa độ một góc d (tính bằng độ). Phép thay đổi tỉ lệ có dạng scale(d) hoặc scale(x,y), thay đổi tỉ lệ kích thước của trục tọa độ theo d lần hoặc theo x lần trục hoành và y lần trục tung. Phép tịnh tiến, phép xoay, phép thay đổi tỉ lệ có thể dùng chung hoặc riêng. Khi dùng chung, thứ tự thực hiện phép biến hình chính là thứ tự khai báo thuộc tính transform.

Ví dụ: dưới đây dùng riêng phép tịnh tiến, cho hình chữ nhật có tọa độ tuyệt đối là (150,150), tịnh tiến đi (-100,200) và sẽ có tọa độ tuyệt đối mới là (50,350).

```
<rect x="150" y="150" width="200" height="200" stroke="rgb(0, 0, 255)"
    stroke-width="10" fill="rgb(255, 0, 0)" stroke-opacity="0.7"
    fill-opacity="0.5" transform="translate(-100, 200)" />
```

Hình ellipse dưới đây có cả xoay và tịnh tiến

```
<ellipse cx="150" cy="100" rx="200" ry="100" stroke="rgb(255, 255, 0)"
    stroke-width="3" fill="rgb(0, 255, 0)" stroke-opacity="0.7"
    fill-opacity="0.5" transform="translate(-200, -30) rotate(-30)" />
```

Đoạn thẳng chỉ có xoay

```
<line x1="100" y1="300" x2="300" y2="100" stroke="rgb(0, 0, 255)"

    stroke-width="5"  stroke-opacity="0.7" transform="rotate(45)" />
```

YÊU CẦU 4: Bổ sung đối tượng đường bất kì - đối tượng Path của SVG.

Đối tượng Path là đối tượng nhóm chứa các đối tượng đồ họa cơ bản dạng đường thẳng hoặc đường cong. Đối tượng Path sẽ chứa một danh sách các điểm điều khiển và các lệnh tương ứng.

* Các loại đường của Path có thể tham khảo từ:

http://www.w3.org/TR/SVG/paths.html#PathData

* Để đơn giản, ta giới hạn các thành phần của Path gồm:

- Các đường thẳng: lệnh moveto (M), lineto (L, H, V) và closepath (Z)
- Đường cong Bezier bậc 3 (Cubic Bézier Curve): lệnh curveto (C)
- Lệnh moveto (M) đi đến 1 điểm với tọa độ x, y.
- Lệnh lineto (L, H, V) nhận input là 1 điểm, vẽ đường thẳng từ điểm hiện tại đến điểm input. Điểm hiện tại có thể là điểm đã moveto, hoặc là điểm đích cuối sau khi thực hiện các lệnh vẽ khác. H, V vẽ các đường nằm ngang hay thẳng đứng.
- Lệnh closepath (Z) vẽ đường thẳng từ điểm cuối đến điểm đầu tiên của Path.
- Lệnh curveto (C) nhận 3 điểm input, và vẽ đường cong bậc 3 qua 4 điểm: điểm hiện tại + 3 điểm input.

Ví dụ:

```
<svg xmlns="http://www.w3.org/2000/svg">
  <path fill="none" stroke="rgb(255,0,0)" stroke-width="5"
        d="M100,200 C100,100 250,100 250,200
            C250,300 400,300 400,200" />
</svg>
```

- M100,200 đi đến điểm (100,200)

- C100,100 250,100 250,200: vẽ đường cong qua 4 điểm (100,200), (100,100), (250,100) và (250,200)
- C250,300 400,300 400,200: vẽ đường cong qua 4 điểm (250,200), (250,300), (400,300) và (400,200)

YÊU CẦU 5: Bổ sung cấu trúc phức hợp - đối tượng Group của SVG.

Đối tượng g là đối tượng nhóm dùng để chứa các đối tượng đồ họa khác. Các thuộc tính của group được áp dụng cho tất cả các đối tượng con, và có thể kết hợp với thuộc tính khai báo trong từng đối tượng con để tạo ra hiệu ứng cuối cùng.

Ví dụ:

```
<svg xmlns="http://www.w3.org/2000/svg">
    <g transform="translate(100, 100) rotate(45)">
      <line x1="10" y1="10" x2="85" y2="10" stroke="rgb(255,0,0)"/>
      <rect x="10" y="20" height="50" width="75"
            stroke="rgb(0,255,0)" fill="rgb(0,255,0)"/>
      <text x="10" y="120" stroke="rgb(255,0,0)" fill="rgb(0,0,255)"
font-size ="50">
        Text grouped with shapes</text>
    </g>
</svg>
```

* Đối tượng g có thể lồng nhau.

Ví dụ:

```
<svg xmlns="http://www.w3.org/2000/svg">
  <g stroke="rgb(255,0,0)" stroke-width="2" transform="translate(100,
0) rotate(45)">
    <line x1="10" y1="10" x2="85" y2="10"/>
    <rect x="10" y="20" height="50" width="75" fill="rgb(0,255,0)"/>
```

```
    <text x="10" y="110" stroke="rgb(255,255,0)" fill="rgb(0,0,255)"
font-size ="55"> Text grouped with shapes</text>
    <g transform="translate(200, 100) rotate(45)">
          <text x="50" y="110" stroke="none" fill="rgb(0,0,255)" font-
size="40"> Nested Group Text</text>
          <line x1="50" y1="130" x2="360" y2="130"/>
    </g>
  </g>
</svg>
```

* Để đơn giản, chúng ta giới hạn các thuộc tính mà g có thể có:

- stroke
- stroke-width
- stroke-opacity
- fill
- fill-opacity
- transform