Đại Học Quốc Gia Thành Phố Hồ Chí Minh
Đại Học Khoa Học Tự Nhiên
Khoa Công Nghệ Thông Tin

BÁO CÁO ĐỒ ÁN CUỐI KỲ
SVG

Lớp 15CNTN

Họ Và Tên: Trần Chí Khang

Mã Số Sinh Viên : 1512237

16 THÁNG MƯỜI HAI 2016

1) Các Vấn đề cần giải quyết.

Đọc File SVG và khởi tạo các đối tượng từ thông tin trong file SVG

Đọc các thuộc tính của từng loại hình trong file SVG

Tinh mở rộng của chương trình , thêm hình mới, thuộc tính mới..

Vẽ các Hình ra màn hình

Xuất ra file SVG

2) Phương pháp

1. **Đọc file SVG và khởi tạo các đôi tượng**

Sử dụng thư viện Rapid XML để đọc file SVG ta có thể dựa vào thuộc tính xml_node để lấy tên hình bằng lệnh node->name(), sau khi lấy được tên hình ta có thể dùng cách đơn giản là dùng các lệnh if else để khởi tạo các đối tượng hình như sau

if(strcmp(node->name,"circle")==0) m_List.pushback(new Circle())

với cách làm như trên ta sẽ phải viết khá nhiều câu lệnh if else gây trùng lặp mã nguồn ở các đoạn if else, và khi thêm một loại hình mới ta lại phải sủa mã nguồn. Để giải quyết vấn đề trên ta có thể dùng cách khởi tạo đối tượng nhờ tham số chuỗi chứa tên lớp như sau:

ta sẽ phối hợp sử dụng kỉ thuật xác định tên lớp nhờ dùng phương thức className() sử dụng đối tượng bản mẫu nhờ pương thức Clone(), ở lớp Figure ta tạo một danh sách để lưu các bản mẫu

```
class Figure {
private:
    static vector<Figure*> SampleObject;
protected:
    static void AddSample(Figure* obj);
public:
    static Figure* CreateObject(char* name);
    virtual char* ClassName() = 0;
```

```
void Figure::AddSample(Figure* obj) {
    if (obj == NULL) return;
    int pos = SampleObject.size();
    while (--pos >= 0) {
        Figure *temp = SampleObject[pos];
        if (NULL == temp) continue;
        // so sanh ten obj voi ten cac doi tuong trong mau, neu
        if (strcmp(temp->ClassName(), obj->ClassName
            break;
    }
    if (pos < 0) {
        // neu trong danh sach mau chua co thi them vao
        SampleObject.push_back(obj->Clone());
```

Hình 2: phương thức AddSample

trường dữ liệu SampleObject dùng để lưu các đối tượng hình, trường này được khai báo static để dùng chung cho tất cả các đối tượng con, phương thức AddSample() để thêm đối tượng vào mẫu, phương thức CreateObject là phương thức chính sẽ tạo lập đối tượng của lớp có tên được lưu trong biến chuỗi name, các phương thức ClassName() và Clone sẽ được cài đặt trong lớp con

Trong phương thức này Với mỗi lớp con thì chỉ có đúng một đối tượng mẫu được đưa vào SampleObject phương thức ClassName() được xây dựng từ các lớp con, khi khởi tạo các lớp con ta thêm lời gọi phương thức AddSample(this) để thêm đối tượng vào mẫu

```
MyCircle::MyCircle() : DrawAttribute(){
    Figure::AddSample(this);
}
```

Hình 1: ví dụ thêm đối tượng vào mẫu

Sau khi đã cài đặt các phương thức trên thì việc cài đặt CreateObject(char* name) khá dễ dàng ta chỉ việc duyệt SampleObject tìm kiếm đối tượng bản mẫu nào thuộc về lớp có tên trùng với tên lớp cần tạo đối tượng(name) sau đó tạo bản sao của đối tượng tìm thấy nhờ vào phương thức Clone()

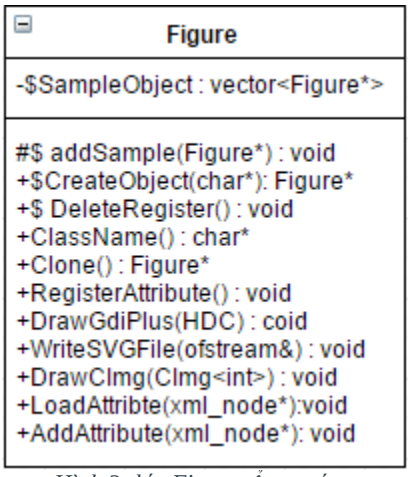
```
Figure* Figure::CreateObject(char* name) {
    if (name == NULL) return NULL;
    int pos = SampleObject.size();
    while (--pos >= 0) {
        Figure *temp = SampleObject[pos];
        if (temp == NULL) continue;
        if (strcmp(name, temp->ClassName()) == 0) break;
    }
    if (pos >= 0) return SampleObject[pos]->Clone();
    return NULL;
}

Figure::~Figure() {

}
```

Hình 2: CreateObject

trường hợp mẫu không chứa đối tượng có tên lớp là name thì trả về NULL.
Công việc tiếp theo ta chỉ việc xây dựng các lớp con và cho nó kế thừa Figure



Hình 3: lớp Figure tổng quát

2. **Đọc các thuộc tính của đối tượng Figure.**
Để đọc các thuộc tính của đối tượng ta cũng áp dụng phương pháp tạo đối tượng nhờ tham số chuỗi chưa tên lớp, ta xây dựng lớp cơ sở Attribute, trong lớp Attribute sẽ chứa một trường dữ liệu AttributeSampleObject , trường này sẽ lưu các mẫu đối tượng thuộc tính, các phương thức addSample,

```
class Attribute {
private:
    static vector<Attribute*> AttributeSampleObject;
protected:
    static void addSample(Attribute* obj);
public:
    static Attribute* CreateObject(char* name);
    virtual char* GetName() = 0;
    virtual Attribute* Clone() = 0;
    // phương thức khác
```

Hình 4 : lớp Attribute

className, getName, Clone tương tự như lớp Figure. Với mỗi thuộc tính ta sẽ xây dựng cho nó một class và kế thừa Attribute.
Tiếp theo để quản lý các đối tượng Attribute ta sẽ xây dựng lớp DrawAttribute lớp này sẽ chứa một danh

sách các Attribute dùng để lưu giữ các thuộc tính phục vụ cho việc sử dụng.

```
class DrawAttribute {
private:
    // danh sach cac thuoc tinh
    vector<Attribute*> m_Attrs;
public:
    DrawAttribute();
    DrawAttribute(DrawAttribute &obj);
    // thuc hien transform (rotate, scale, Translate);
    void transform(Graphics &graphics);
    void WriteAttribute(ofstream &ofs , int start=0);
    void Load(xml_node<> *node);
    int getNumAttribute();
    void addAttribute(Attribute* obj);
    void SetAttributeForFigure(Figure* fig);
    Pen* getPen();
    Brush* getBrush();
    virtual Attribute* findAttributeByName(char *name, int nth=0);
    virtual ~DrawAttribute();
};
```

Hình 5: lớp DrawAttribute

Trong lớp Attribute phương thức Load() có nhiệm vụ tạo các đối tượng thuộc tính từ node và thêm thuộc tính vào danh sách,ở phương thức Load này dựa vào tính đa hình ta sẽ gọi đến phương thức SetAttribute() (được cài đặt ở các lớp con của Attribute) để cài đặt giá trị cho thuộc tính. Phương thức findAttributeByName() cho phép lấy đối tượng thuộc tính trong danh sách có tên lớp trùng với name, nhờ phương thức này ta có thể truy xuất đối tượng thuộc tính để phục vụ cho việc vẽ và ghi file, theo mặc định phương thức này sẽ trả về thuộc tính có tên lớp là name ở cuối danh sách, phương thức transform() sẽ tìm tất cả đối tượng Transform trong danh sách và thực hiện Transform, các phương thức getPen() và getBrush() dùng để trả về Pen và Brush khi vẽ bằng GDI+
sau khi xây dựng DrawAttribute ta cho các lớp con của Figure kế thừa DrawAttribute để lớp này quản lý các thuộc tính.

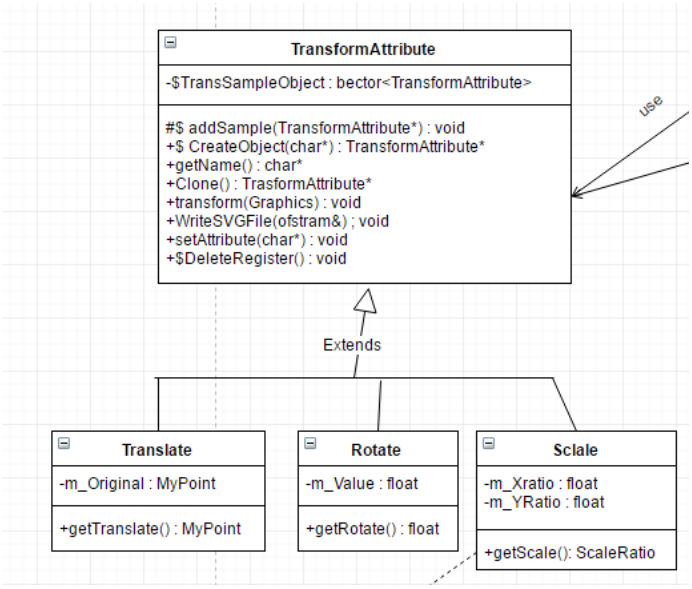
3. **Giải quyết thuộc tính Transform**

Với thuộc tính Transform ta cũng sử dụng phương pháp tạo đối tượng nhờ tham số chuỗi chứa tên lớp tương tự như lớp Figure trước tiên ta tạo lớp TransformAttribute chứa trường dữ liệu mẫu TrasformSampleObject, các phương thức Addsample(), GetName() , Clone() tương tự như lớp Figure

```
class TransformAttribute {
    // danh sach mau Cac TransformAttribute (rotate , translate, scale)
    static vector<TransformAttribute*> TransSampleObject;
protected:
    // them TransformAttribute vao danh sach mau
    static void addSample(TransformAttribute* obj);
public:
    // tao doi tuong TransformAttribute bang ten
    static TransformAttribute* CreateObject(char *name);
    // tra ve ten lop con cua TransformAttribute
    virtual char* GetName() = 0;
    // nhan ban doi tuong TransformAttribute
    virtual TransformAttribute* Clone() = 0;
    // thuc hien lenh transform
    virtual void transform(Gdiplus::Graphics &graphics) = 0;
    //gi gia tri transform ra SVG file
    virtual void WriteSVGFile(ofstream &ofs) = 0;
    // cai dat gia tri cho cac doi tuong TrasformAttribute
    virtual void SetAtribute(char* attr) = 0;
    // xoa danh sach mau
```

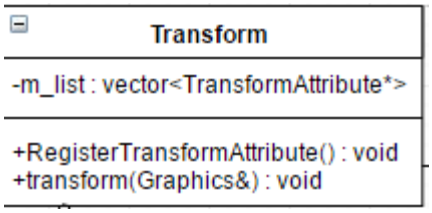
Hình 6: lớp TrasformAttribute

Phương thức transform() thực hiện transform sẽ được cài đặt trong các lớp con, phương thức WriteSVGFile() dùng để ghi thông tin ra file và phương thức setAttribute() sẽ được cài đặt ở các lớp con. Tiếp theo với mỗi thuộc tính transform ta sẽ xây dựng một lớp và kế thừa lại lớp TransformAttribute.



Hình 7: các thuộc tính Transform

Ta tạo lớp Transform chứa danh sách các đối tượng TransformAttribute



Hình 8: lớp Transform

khi thực hiện lệnh vẽ ta chỉ việc gọi DrawAttribute::transform() thì phương thức này sẽ tìm các đối tượng Transform và duyệt danh sách TransformAttribute và cho các đối tượng TransformAttribute gọi phương thức transform nhờ vào tính đa hình,

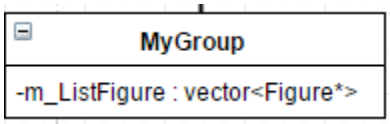
4. **Giải quyết đọc thông tin Đối tượng Group**

Lớp Group sẽ kế thừa lớp Figure và lớp DrawAttribute, nó sẽ giữ danh sách các đối tượng Figure, trong lớp Group ta sẽ cài đặt phương thức LoadAttribute() để đọc thông tin từ SVG như sau

```
void Group::LoadAttribute(xml_node<> *node) {
    if (NULL == node) return;
    // load cac Attribute chung
    DrawAttribute::Load(node);
    xml_node<> *subnode = node->first_node();
    do {
        // khoi tao cac doi tuong Figure ma group chua thong qua ten
        Figure* object = Figure::CreateObject(subnode->name());
        if (object != NULL) {
            // dang ki Attribute
            object->RegisterAttribute();
            // nap cac Attribute cua group cho cac con cua no
            DrawAttribute::SetAttributeForFigure(object);
            // cho cac con cua group cai dat Attribute cua rieng no
            object->LoadAttribute(subnode);
            // nap vao danh sach
            m_ListFigure.push_back(object);
        }
        // chuyen den node ke tiep
        subnode = subnode->next_sibling();
    } while (subnode);
}
```

Hình 10 : phương thức LoadAttribute

Phương thức này sẽ đọc thông tin từ node và tạo lập các đối tượng Figure mà group bọc lấy, ở lớp Group các thuộc tính của Group sẽ là thuộc tính chung cho các loại hình mà nó bọc lấy nên ta sẽ cho Group đọc thuộc tính chung trước bằng lệnh DrawAttribute::Load(), sau đó ta sẽ duyệt các node và tạo lập đối tượng Figure bằng phương thức Figure::CreateObject() với mỗi đối tượng Figure được tạo ta sẽ gọi DrawAttribute::SetAttributeForFigure() để nạp các thuộc tính chung vào đối tượng Figure, sau đó ta sẽ cho các đối tượng Figure đọc các thuộc tính của riêng nó như thế các thuộc tính của các Figure sẽ được sắp theo thứ tự từ thuộc tính chung đến thuộc tính riêng, cuối cùng thêm đối tượng Figure vào danh sách ListFigure.



Hình 9: lớp Group

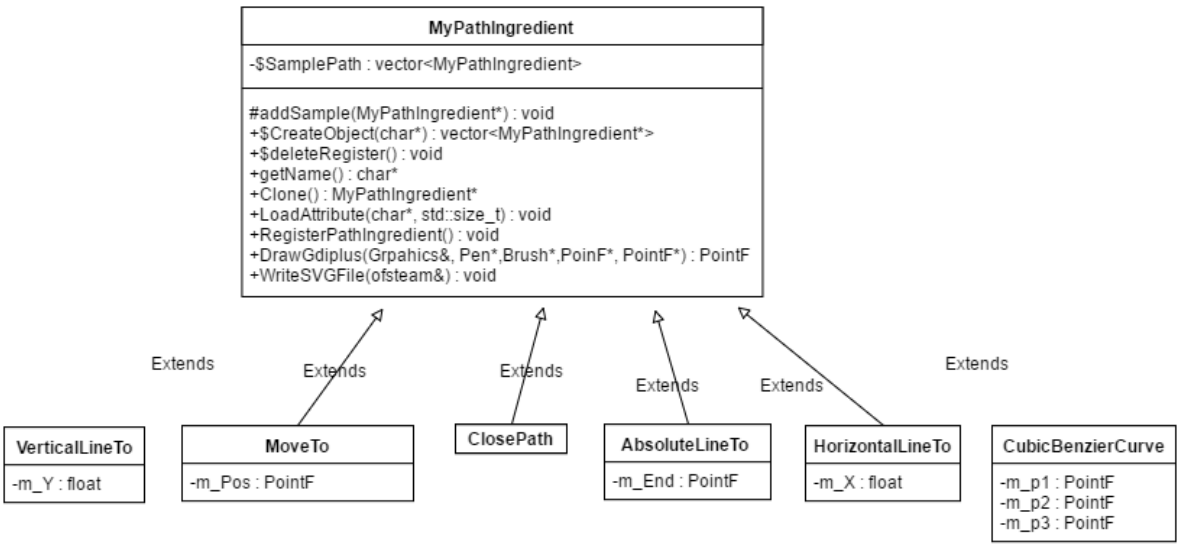
5. **Giải quyết Đọc thông tin đối tương Path**

Ở đây ta cũng áp dụng kỉ thuật tạo đối tượng nhờ tham số chuỗi chứa tên lớp, ta xây dựng lớp MyPathIngredient tương tự như Figure chứa trường dữ liễu mẫu SamplePath (các loại lệnh path)

```
class MyPathIngredient {
private:
    // danh sach mau cac loai lenh trong path
    static vector<MyPathIngredient*> SamplePath;
protected:
    // them mau
    static void addSample(MyPathIngredient* obj);
public:
    // tao doi tuong bang ten
    static vector<MyPathIngredient*> CreateObject(char *name);
    // tra ve ten cac lenh cua path
    virtual char* getName() = 0;
    // nhan ban doi tuong
    virtual MyPathIngredient* Clone() = 0;
    // loat thuoc tinh cho Attribute
    virtual void LoadAttribute(char* s, std::size_t start) = 0;
    // ve gdi+ voi diem  hien tai lastPoint
    virtual PointF DrawGdiplus(Gdiplus::Graphics &graphics, Pen* pen , Brush* brush , PointF* LastPoint = NULL, PointF*  StartPoint = NULL) = 0;
    // viet ra file SVG
    virtual void WriteSVGFile(ofstream &ofs) = 0;
```

Hình 11: lớp MyPathIngredient

phương thức chính CreateObject sẽ trả về một danh sách các đối tượng MyPathIngredient thông qua chuỗi giá trị của Thuộc tính MyPathData., phương thức LoadAttibute() được cài ở lớp con dùng để cài đặt thuộc tính cho các đối tượng lệnh path, phương thức DrawGdiPlus() được cài ở lớp con dùng để vẽ các lệnh Path với điểm hiện tại (lastPoint) do lớp MyPath cung cấp. Phương thức WriteSVGFile() được cài ở lớp con dùng để viết thông tin các lệnh path ra file.



Hình 12: lớp các lệnh Path

sau khi xây dựng lớp MyPathIngredient, với mỗi lệnh path ta xây dựng lớp con kế thừa lại MyPath-Ingredient va nạp nó vào danh sách mẫu.
Sau các bước trên ta xây dựng lợp MyPathData kế thừa lớp Attribute lớp này sẽ chứa danh sách các đối tượng MyPathIngredient (m_list)

```
class MyPathData : public Attribute {
private:
    // danh sach cac doi tuong con MyPathIngredient
    vector<MyPathIngredient*> m_list;
public:
    MyPathData();
    MyPathData(MyPathData &obj);
    char* GetName();
    Attribute* Clone();
    vector<MyPathIngredient*> getPathData();
    void SetAtribute(char* attr);
    void WriteSVGFile(ofstream &ofs);
    ~MyPathData();
};
```

Hình 13: lớp MyPathData

tiếp đến ta xây dựng lớp MyPath kế thửa Figure và DrawAttribute

```
class MyPath : public DrawAttribute , public Figure {
private:
    // diem bat dau cua net ve truoc do
    PointF m_StartPoint;
    // diem hien tai
    PointF m_LastPoint;
public:
    MyPath();
    MyPath(MyPath &obj);
    // trien khai cac phuong thuc cua lop Figure
    char* ClassName();
    Figure* Clone();
    void RegisterAttribute();
    void DrawGDIPlus(HDC hdc);
    void WriteSVGFile(ofstream &ofs, int startAttribute=0);
    void DrawCImg(CImg<int> &Frame);
    void LoadAttribute(xml_node<> *node);
    void addAttribute(Attribute* obj);
    ~MyPath();
};
```

Hình 14: lớp MyPath

lớp này sẽ chứa điểm khởi đầu của nét vẽ trước (m_StartPoint) và điểm hiện tại (m_LastPoint), phương thức DrawGDIPlus() cho phép vẽ trên thư viện GDI+ trong phương thức này ta dùng DrawAttribute::findAttributeByName("d") để lấy đối tượng MyPathData sau đó dùng đối tượng MyPathData gọi phương thức getPathData() để lấy danh sách MyPathIngredient và dựa vào đa hình ta sẽ gọi phương thức DrawGdiPlus() ở các lớp con của MyPathIngredient vẽ lên màn hình.

6. **Thực hiện vẽ**

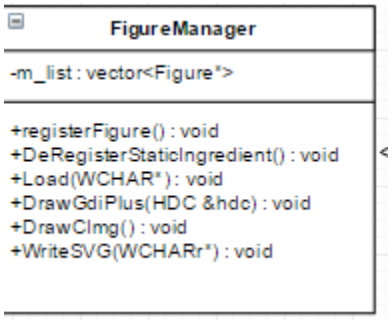
sau khi đã cài đạt xong thông tin cho các thuộc tính ta có thể cho các Figure thực hiện lệnh vẽ bằng phương

thức DrawGDIPlus hoặc DrawCImg, trong các hàm vẽ để lấy attribute, ở đây ta đã xác định attribute là loại nào nên ta chỉ việc gọi phương thức DrawAttribute::FindAttributeByName(char*name) để lấy ra đối tượng Attribute trong danh sách và ép kiểu từ Attribute* sang kiểu con trỏ Attribute tương ứng và gọi các phương thức get để lấy dữ liệu, phương thức DrawAttribute::FindAttributeByName(char* name) cho phép tìm kiếm Attibute thứ n trong danh sách, mặc định phương thức này sẽ kiếm Attribute có tên name ở cuối trong danh sách. Trong DrawAttribute đã cung cấp sẵn phương thức getPen() và GetBrush().

```
void MyCircle::DrawGDIPlus(HDC hdc){
    Pen *pen = DrawAttribute::getPen();
    Brush *brush = DrawAttribute::getBrush();
    XCenter *xcenter = (XCenter*)DrawAttribute::findAttributeByName("cx");
    YCenter *ycenter = (YCenter*)DrawAttribute::findAttributeByName("cy");
    Radius *radius = (Radius*)DrawAttribute::findAttributeByName("r");
    int cx = (xcenter == NULL) ? 0 : xcenter->getX();
    int cy = (ycenter == NULL) ? 0 : ycenter->getY();
    int r = (radius == NULL) ? 0 : radius->getR();
    Graphics graphics(hdc);
    DrawAttribute::transform(graphics);
    graphics.FillEllipse(brush, cx - r, cy - r, 2 * r, 2 * r);
    graphics.DrawEllipse(pen, cx - r, cy - r, 2 * r, 2 * r);
    delete pen;
    delete brush;
}
```

Hình 15: vi du hàm DrawGDIPlus của lớp Cicle

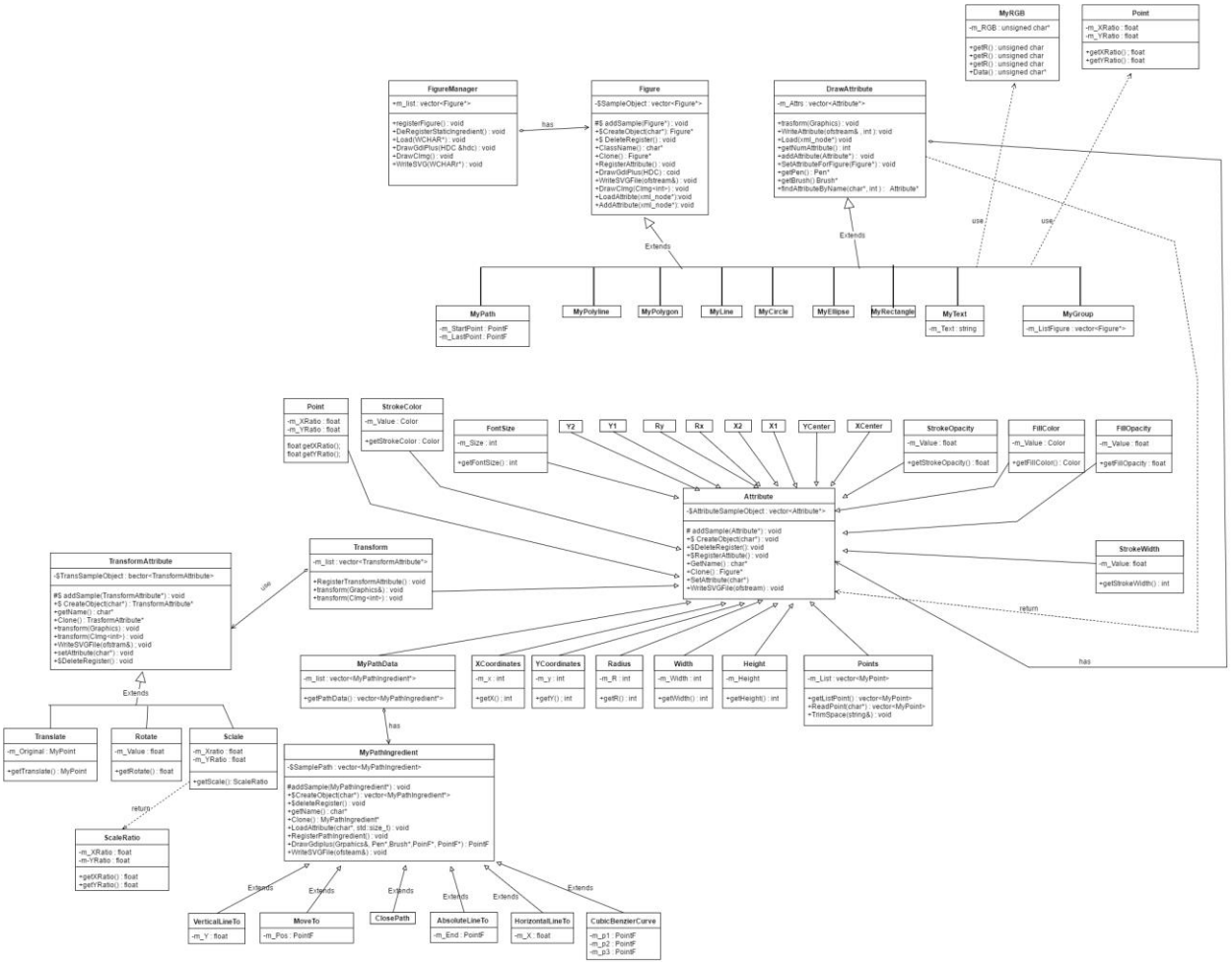
7. **Thực hiện viết File SVG**
   cũng giống như vẽ khi viết file SVG ta cũng cho từng đối tượng Figure viết thông tin của nó ra file, các Attribute sẽ do phương thức WriteAttribute của lớp DrawAttribute thự hiện, theo mặc định các Attribute sẽ được viết từ đầu cho đến cuối danh sách DrawAttribute, ta có thể tùy chỉnh viết Atribute bằng cách thêm vị trí bắt đầu viết trong Phương thức DrawAttribute::WriteAttribute(ofstream, int).
8. **Xây dựng Lớp FigureManager**
   Lớp FigureManger sẽ chứa danh sách các đối tượng Figure, Lớp FigureManager sẽ đảm nhiệm việc quản lý các đối tượng Figure. Phương thức RegisterFigure() dùng để nạp các đối tượng Figure vào danh sách mẫu



Hình 16: Lớp FigureManager

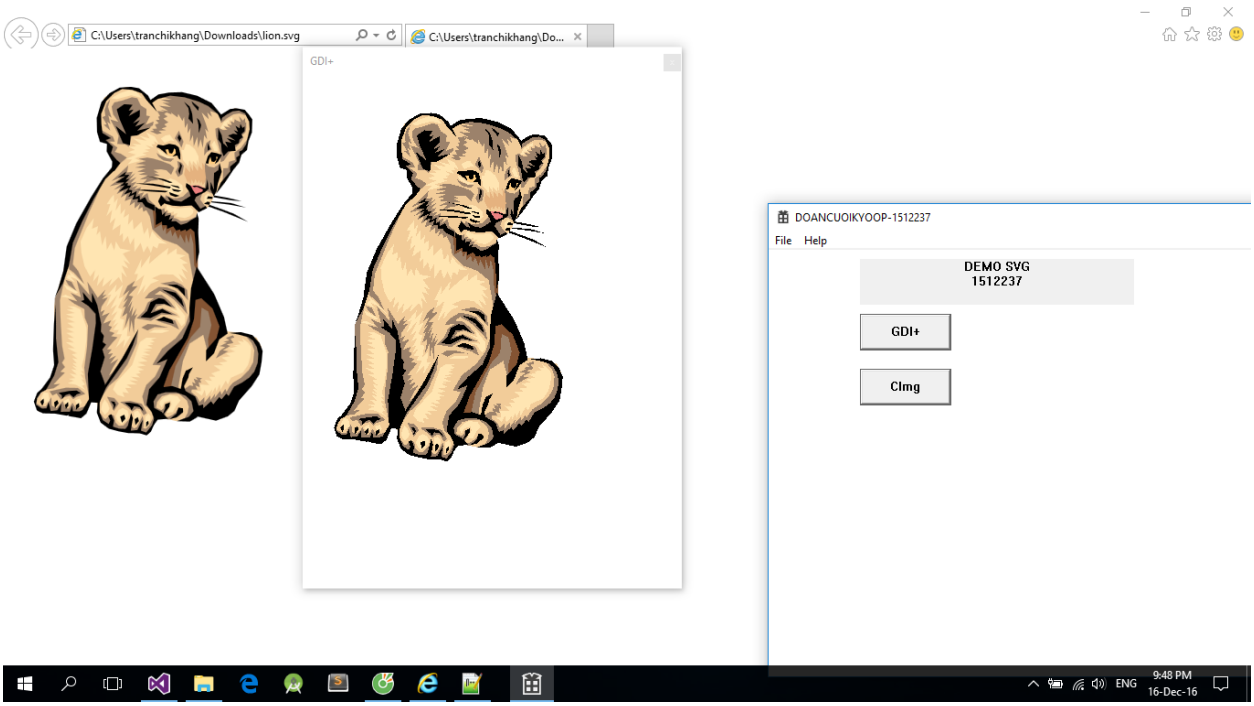
(SampleObject) trong lớp Figure, phương thức DeRegisterStaticIngredient() dùng để giải phóng vùng nhớ các danh sách mẫu khi kết thúc chương trình, phương thức Load dùng để mở file SVG tạo các đối tượng Figure và cho các đối tượng Figure cài đạt giá trị thuộc tính, phương thức DrawGdiplus nhờ vào đa hình sẽ gọi các đối tượng con của Figure vẽ bằng thư viện GDI+, tương tự phương thức DrawCImg() vẽ bằng thư viện CImg, phương thức WriteSVG() hờ vào đa hình sẽ gọi các con của Figure trong danh sách m_list viết ra file SVG.

3) UML

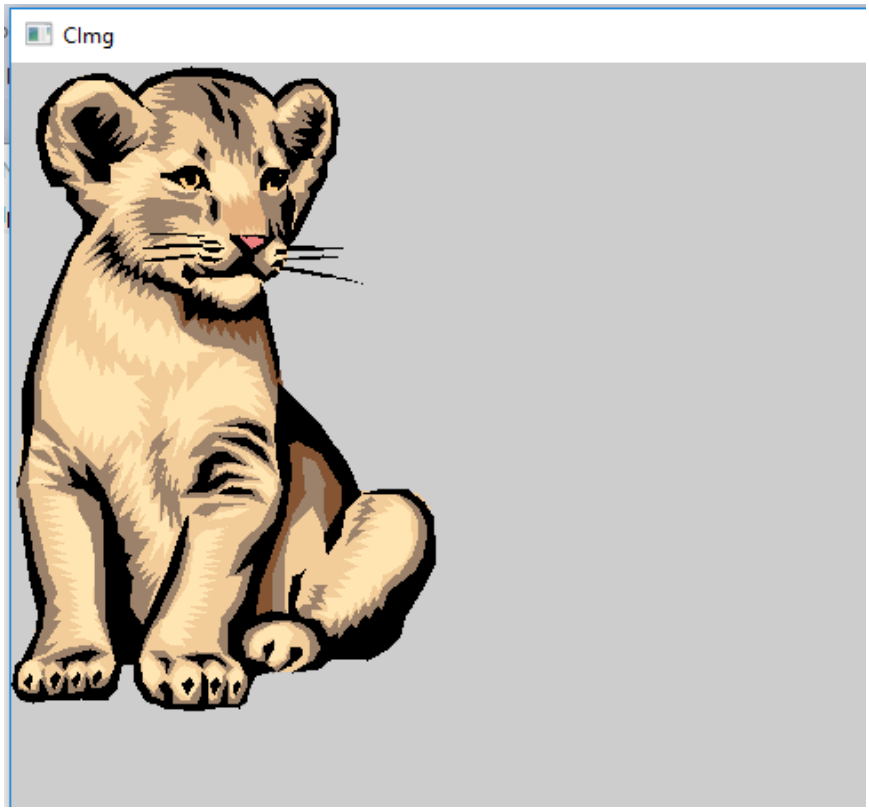


Hình 17: sơ đồ UML

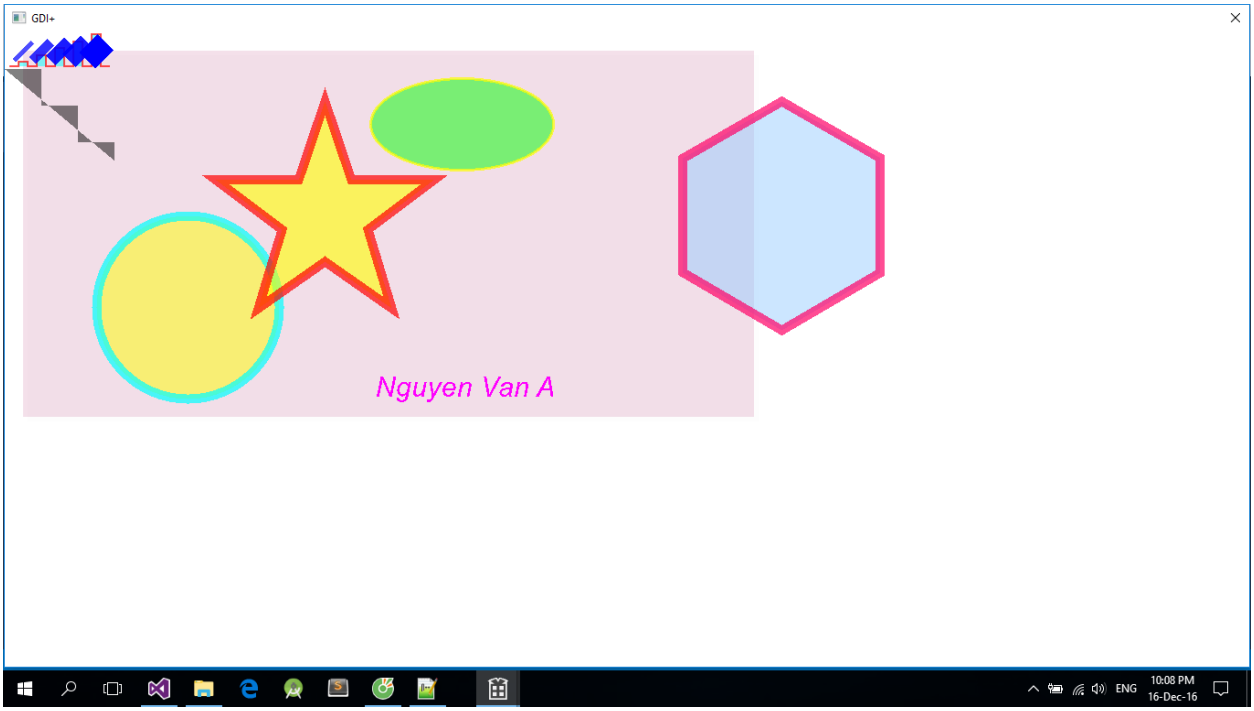
4) Kết quả



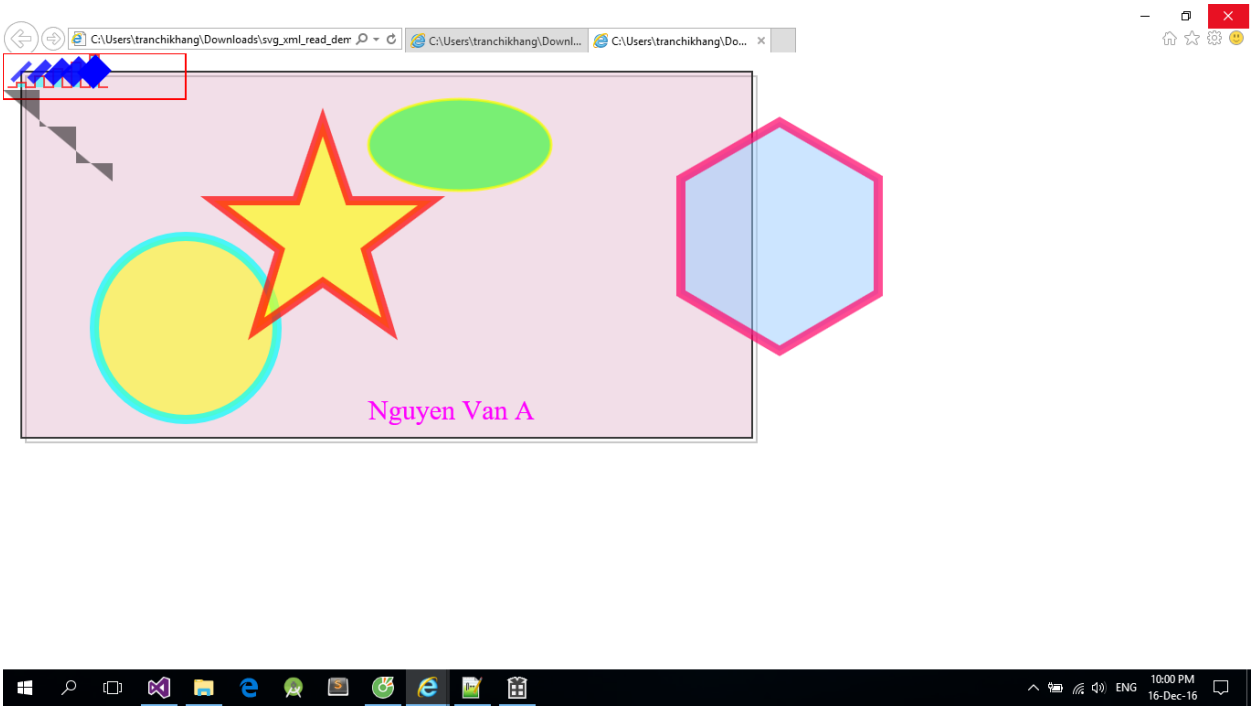
Hình 18: lion.svg vẽ bằng GDI+ so sánh với Trình duyệt



Hình 19: lion.svg ve bang CImg



Hình 20: file sample.svg vẽ bằng GDI+



Hình 21: sample.svg mở bằng trình duyệt

5) Đánh Giá

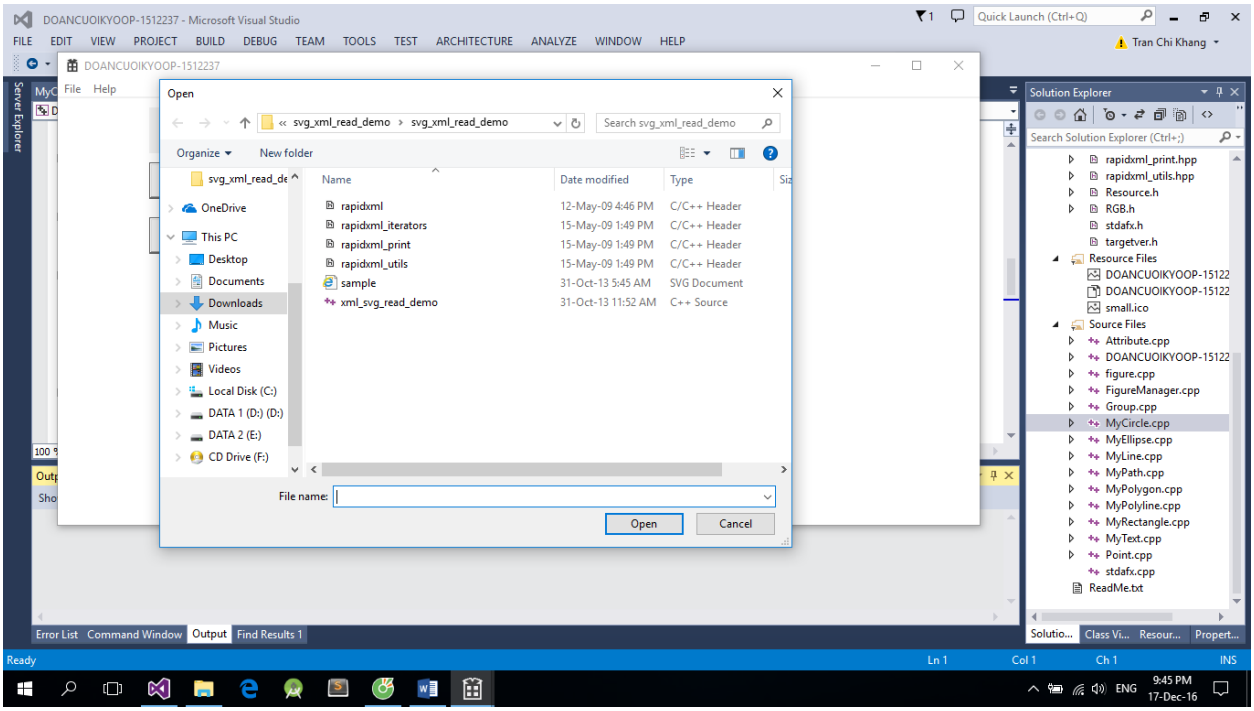
Việc dùng phương pháp tạo đối tượng bằng tên giúp cho việc khởi tạo đối tượng trở nên mềm hóa hơn và giúp chương trình trở nên tổng quát hơn so với việc tạo đối tượng bằng cách gọi trực tiếp toán tử new và chương

trình có khả năng tái sử dụng cao, khi thêm một loại hình mới ta chỉ việc kế thừa lớp cơ sở và triển khai các phương thức thuần ảo của nó sau đó thêm vào danh sách mẫu, tương tự với các Attribute khi thêm một thuộc tính mới ta chỉ việc kế thừa lớp cơ sở và triển khai các phương thức thuần ảo của nó và thêm vào danh sách mẫu.

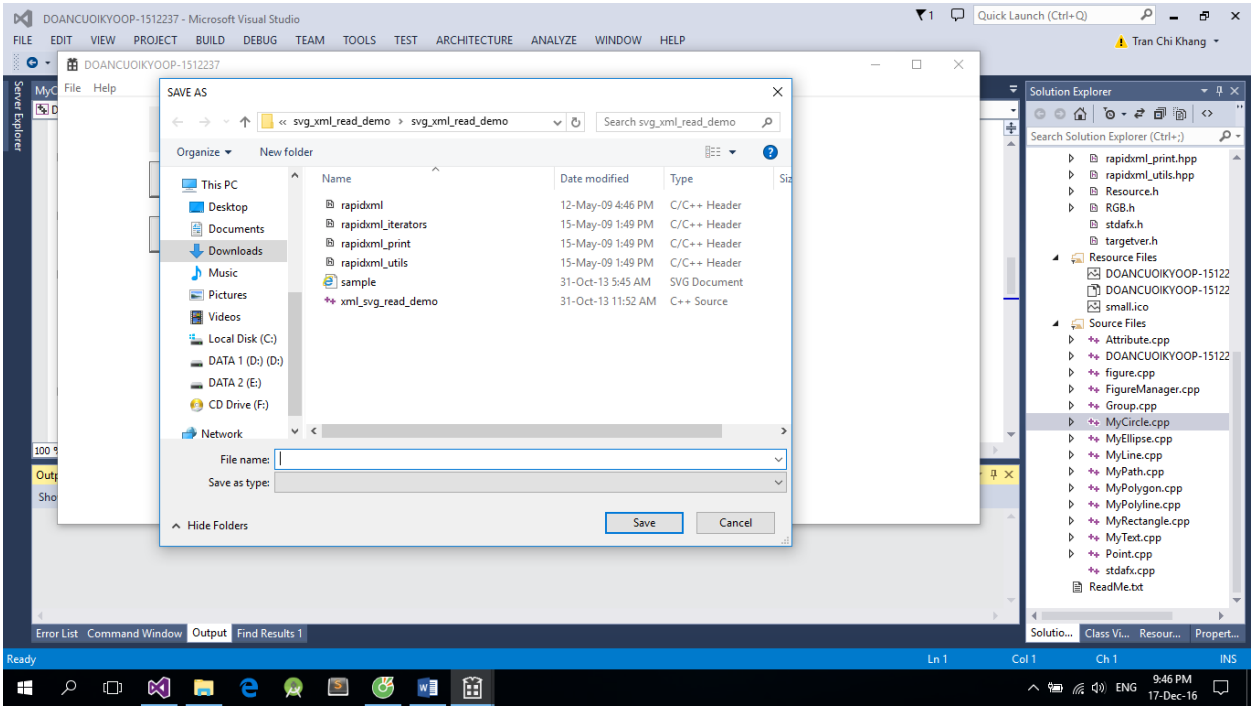
Trong đồ án này chỉ vẽ tốt được trên thư viện GDI+, thư viện CImg tuy có nhưng chưa được hoàn chỉnh, còn thiếu ở phần vẽ Path, Group, và Polyline chưa vẽ các nét.

trong đồ án có xây dựng một giao diện đơn giản để thực hiện mở file, lưu file và vẽ.

Hình 22: giao diện khi mở chương trình



Hình 23: giao diện mở file



Hình 24: giao diện Save file