SVG 作成フレームワーク SVuGy の拡張

前村 武，藤森 誠，伊藤 一成，Martin J. DÜRST
青山学院大学理工学部

1 はじめに

画像表示媒体の高精細化，携帯機器の処理能力の向上に伴い，画像を点の集合体ではなく点や線の方程式として表現するベクタグラフィック技術の需要が高まっている．代表的な規格として SVG [1]，Adobe Flash，EPS 等が利用されている．グラフィックの作成にはドローツールと呼ばれるソフトウェアを用いるのが一般的であり，自由な描画ができる一方，細かい座標指定や再利用は容易ではない．

特に，SVG は XML フォーマットであり，オープン規格であるという特徴から，近年注目を浴びている．そこで我々は，手に馴染んだテキストエディタ上で作成でき，さらに容易な記述形式を実現したフレームワーク SVuGy [2] を考案した．SVuGy は 2007 年春より継続して実装を進めている．今回，その基本構造を拡張し，外部ファイルとの親和性の向上や図形オブジェクトの抽象化を進め，利便性を向上させたので報告する．

2 SVG の記法の問題点

2.1 座標の再利用性

SVG では属性値に変数を用いることができない．例えば先に記述した図形の右端から，その図形幅の 1/10 のマージンを取って新たな図形を記述したい際には改めて具体的な座標を手計算し，記述する必要がある．

2.2 煩雑な未定義図形の描画法

SVG では基本図形として，矩形，円，楕円，直線，折れ線，多角形が定義されている．例えば円は中心の x, y 座標と半径を指定して描画する．しかし標準で定義されていない図形，例えば星型正五角形を描く場合は，多角形とみなし，10 個の頂点の x, y 座標を一つ一つ指定する必要がある．当然ながら，これら全ての座標は前もって計算しておく必要があり，記述者に対して大きな負担となる．

3 SVuGy の概要

SVuGy はオブジェクト指向スクリプト言語である Ruby を用いて作られた内部 DSL（ドメイン特化言語）[3] である．名称は，Ruby の母音部と SVG を組み合わせたもので，スヴィージーと発音する．

3.1 使用言語 Ruby

前述の問題点を解決し，かつ極力シンプルな記法を可能にするためフレームワークの作成に Ruby を選択した．ここでは 3 点の長所を説明する．第一にメソッドがコードブロックを受け取ることで文脈依存の処理が可能なこと，第二に動的言語であるため実行中のメソッド追加や未定義変数に対する処理を細かく指定できメタプログラミングを行い易いこと，そして第三に言語自身の構文がシンプルかつ柔軟なことである．これにより，記述の際にクラス生成などの本質的でない処理を省略し，宣言的な記法を可能とした．

3.2 記法

図 1 の図形を SVuGy と SVG で記述した例を以下に示す．SVuGy コードを記述した Ruby ファイルを実行すると SVG コードが出力される．SVuGy では引数の順序を固定することで属性名の指定を省略した．また Ruby の構文ではメソッドの引数を囲む括弧や文末のセミコロンが省略可能なため，より一層，記述量の削減と可読性の向上が実現できる．

```
g {
  translate 10, 10
  rect :r, 150, 50
  ellipse(:e, r.mc, r.w/2 - 1, r.h/2 - 1) {
    fill  'yellow'
    label 'Hello!'
  }
}
```

上記コード内の label メソッドは対象図形の大きさと中心座標から最適なパラメータを計算し，文字列要素を出力するという作業をまとめたものである．

```
<g transform='translate(10 10)'>
  <rect id='r' width='150' height='50'/>
  <ellipse id='e' cx='75' cy='25'
           rx='74' ry='24' fill='yellow'/>
  <text x='75' y='38.44' font-size='38.4'
        text-anchor='middle'>Hello!</text>
</g>
```

図 1: SVG 図形の例

Extension to the SVG Authoring Framework SVuGy
Takeshi MAEMURA, Makoto FUJIMORI, Kazunari ITO and Martin J. DÜRST
Department of Integrated Information Technology, College of Science and Engineering, Aoyama Gakuin University
5-10-1 Fuchinobe, Sagamihara, Kanagawa 229-8558, Japan
{maemura, fujimori}@sw.it.aoyama.ac.jp,
{kaz, duerst}@it.aoyama.ac.jp

4 構造の拡張

今回 SVuGy に追加，変更した機能を説明する．

4.1 変換行列を加味した座標の呼び出し

以前より SVuGy では図形の特徴点の座標数値を保持していたが，図形に transform 属性が指定された場合，属する座標系が変形し再利用が困難となる問題があった．図 2 では別々の座標系に位置する点 A と点 B を直線で結んでいる．両点とも各々の座標系では（100, 100）に位置するため，変換行列を加味しない場合，点 A と点 B が同一点として扱われてしまう．しかし実際は，点 B は点 A の座標系から見ると別座標となるため，直線が引かれなければならない．この問題を解決するために，座標の呼び出し元と座標の座標系が異なる際は，呼び出し元の座標系に自動変換するようにした．

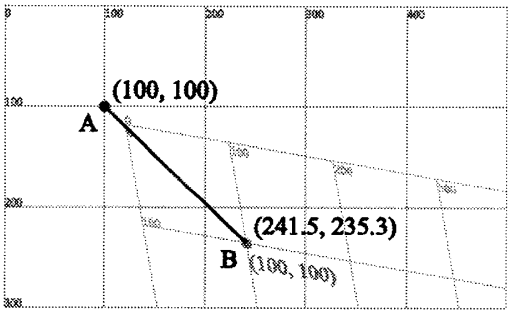

図 2: 座標系

4.2 図形の輪郭線が通過する座標の処理

図形の輪郭線長および面積，図形が含まれる矩形範囲情報（図 3 の枠線），各図形間を繋ぐ最適線（図 4）などを算出し利用するため，通過座標情報を保持するようにした．図 3 中の点は SVG のパスを用いて描画した線が保持している座標情報を示したものである．

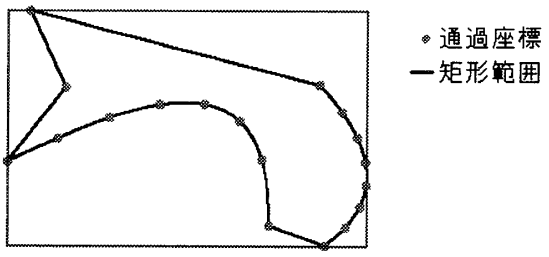

図 3: 通過座標と矩形範囲

4.3 応用図形クラスの作成

未定義の図形は，SVG で定義されている範囲の要素を用いて描画する必要がある．しかしその際に指定するパラメータは必ずしも直感的なものではない．そこで未定義の図形や，複数の要素をまとめて作成し，他の図形オブジェクトと同様に扱うことのできる応用図形クラスを作成した．図 2 の罫線や，図 4 の星形五角

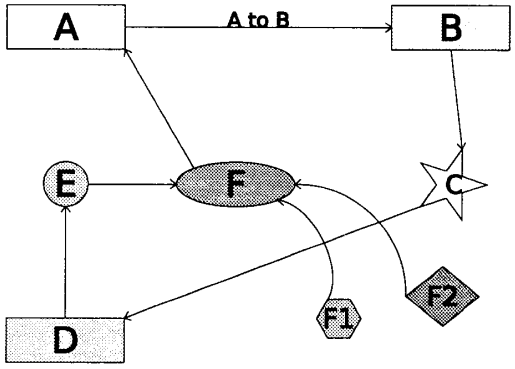

図 4: 最適線と応用図形クラスを用いた例

形，正六角形などが一例である．また，ユーザによる応用図形の追加にも配慮し，専用クラスを継承することを除けば，通常の SVuGy の構文の記述のみでクラスを作成可能にした．

4.4 既存 SVG ファイルの取込および解析

当初 SVuGy は スクラッチから作成することを目的に開発されたため，既に作成された SVG ファイルを活用する方法が存在しなかった．そこで，Ruby の XML パーサである REXML を用いて既存の SVG ファイルを解析することにより，それぞれの図形要素を SVuGy のオブジェクトとして取り込むことを可能にした．これにより，例えば SVG 地図上への道順案内や距離の追記など，既存の図形情報を利用してより効率良くグラフィックを作成できる．

5 まとめと今後の課題

今回の機能拡張により，既存データの再利用性を高め，ライブラリとしての利用も可能になった．また，座標単位だけではなく，図形単位でオブジェクト同士の関係性を表し，利用する筋道を示すことができた．しかし SVG には DOM インタフェースやスタイルシートとの連携も可能となっている．これらを SVuGy 側でラッピング処理し，よりインタラクティブなグラフィックの作成に活かすことができるだろう．今後はこれらを踏まえ，アニメーションやフィルタ等のデザイン面の強化や，より直感的な記法をさらに検討していく予定である．

参考文献

[1] Scalable Vector Graphics (SVG) 1.1 Specification, W3C Recommendation. `http://www.w3.org/TR/SVG11/`, 2003.

[2] Martin J. Dürst, Makoto Fujimori, Takeshi Maemura, Tohru Koga, and Kazunari Ito. SVuGy - Exploring the Space between Procedural and Declarative Graphics. In *Proceedings of the 5th International Conference on Scalable Vector Graphics (SVG Open 2007)*, 2007.

[3] Martin Fowler. Domain Specific Language. `http://www.martinfowler.com/bliki/DomainSpecificLanguage.html`, 2004.