Feuille de travaux pratiques n° 3

Projet : Langage de génération d'images SVG

7 mars 2013

1 Objectif

Le but est de générer des images au format SVG [1] à partir d'un langage de de programmation décrivant le dessin. Vous devez créer ce langage et en implémenter son compilateur en Ocaml avec ocamllex et ocamllyacc.

2 Formats

2.1 Langage

Vous êtes libres de créer le langage comme bon vous semble. Voici un exemple de langage duquel vous pouvez vous inspirer :

```
dessin "exemple svg" [500x500] {

    procedure(entier e) { // déclaration d'une procédure
        dessine rectangle {x=(i,i), y=(i+2,i*3)};
    }

    reel x := 5.3;

    point a; //déclaration d'un point
    point b := (x,5); // déclaration et affectation

    a := (0,1); // affectation

    // déclaration de quatre cercles
    cercle c1;
    cercle c2 := {centre=a, rayon=3};
    cercle c3 := {centre=(3,6), rayon=5, fond="bleu", bord="rouge"};
    cercle c4 := c2 avec rayon = 6;

    dessine c1;
    dessine c4;
    dessine a;

    pour entier i de 1 à 10 { // une boucle
        dessine c1 avec rayon = i;
        dessine(i, i+1); // dessine un point
        si i%2 = 0,
        alors f(i);
    }
}
```

Vous n'êtes pas obligés de suivre ce langage, mais il doit tout de même avoir certaines caractéristiques :

1. Produire un dessin de la taille spécifiée ;
2. Être capable de produire différents types d'objets simples (points, cercles, rectangles, lignes) en noir et blanc ;
3. Reconnaître des commentaires fin de ligne (comme en C++ ou en Python) ;
4. Les types d'objets peuvent avoir des propriétés optionnelles telles que leur couleur de remplissage ou de bordure, etc. ;
5. Être capable de produire différents types d'objets complexes (courbes, polygones fermés ou ouverts) ;
6. Implémenter les affectations ;
7. Implémenter des opérations ;
8. Implémenter la modification des champs d'un objet (exemple ligne 18) ;
9. Implémenter des boucles ;
10. Implémenter des conditionnelles ;
11. Implémenter des procédures ;
12. Implémenter des fonctions.

Ces caractéristiques sont à implémenter si possible l'une après l'autre (ayez une étape qui fonctionne avant de passer à la suivante) dans l'ordre de votre choix. Il est conseillé de conserver une copie du code à chaque étape. Vous pourrez alors inclure les différentes étapes dans l'archive que vous rendrez et les décrire dans le rapport en expliquant vos choix.

2.2 SVG

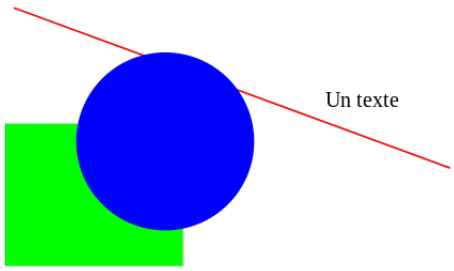

FIGURE 1 – Exemple de figure SVG

SVG est un format XML de description d'image. Il s'agit d'un format dit vectoriel. La page Wikipedia sur SVG [1] donne comme exemple le dessin de la Figure 1. Le code SVG de ce dessin est le suivant :

```
<?xml version="1.0" encoding="utf-8"?>
  <svg
    xmlns="http://www.w3.org/2000/svg"
    version="1.1"
    width="300"
    height="200">
  <title>Exemple simple de figure SVG</title>
  <desc>
    Cette figure est constituée d'un rectangle,
    d'un segment de droite et d'un cercle.
  </desc>
  <rect
    width="100" height="80"
```

```
    x="0" y="70"
    fill="green" />
  <line
    x1="5" y1="5"
    x2="250" y2="95"
    stroke="red" />
  <circle
    cx="90" cy="80"
    r="50"
    fill="blue" />
  <text x="180" y="60">
    Un texte
  </text>
</svg>
```

Il existe de nombreux sites internet sur lesquels vous pouvez trouver beaucoup d'informations utiles sur le SVG, par exemple [2] pour une description des formes élémentaires.

3 Instructions

Ce projet est à réaliser seul ou en binôme. Vous devez rendre une archive au format .tar.gz ou .zip par courriel (benoit@guedas.com) au plus tard une semaine après la dernière séance de TP. Cette archive doit contenir les sources de votre programme, un Makefile pour le compiler ainsi qu'un rapport de TP au format PDF. N'incluez pas d'exécutable. Votre archive doit être nommée nom1_nom2.tar.gz où nom1 et nom2 correspondent aux deux noms de famille du binôme. La décompression de l'archive doit créer un dossier du même nom comtenant tous les éléments.

Votre compilateur doit prévenir des erreurs de compilation (Motif non reconnu, oublie d'une déclaration, ...) en indiquant la ligne de l'erreur dans le code source. Vous n'êtes pas obligés d'implémenter la totalité du langage, mais il vous est demandé d'en faire le plus possible, le plus correctement possible et de suivre les étapes décrites. Par exemple, il ne devra pas implémenter les boucles si les objets de base ne fonctionnent pas. Cela vous permettra de progresser graduellement en complexité. N'hésitez pas non plus à ajouter des fonctionnalités si vous le pouvez. Certaines instructions sont volontairement vagues pour que vous fassiez des choix de conception sur votre langage.

Le rapport est la partie la plus importante du projet et ne doit donc pas être négligée au profit du code. Le rapport ne doit pas simplement décrire votre implémentation, mais plutôt votre démarche. Vous devez décrire le fonctionnement de votre analyseur lexical et syntaxique, la grammaire du langage que vous avez créé et les choix que vous avez faits, les structures de données que vous avez utilisées et les algorithmes implémentés pour les parcourir.

N'hésitez pas à demander de l'aide à l'enseignant de TP pour l'implémentation en Ocaml ou toute autre indication. Des indications et exemples de code Ocaml seront donnés lors des séances.

Références

[1] Page SVG de Wikipedia : `http://fr.wikipedia.org/wiki/Svg`

[2] Cours sur le SVG : `http://www.liafa.jussieu.fr/~carton/Enseignement/XML/Cours/SVG/index.html`