Projet Ada

Nathan GUYOT Vincent LEFOULON

Novembre 2015

1 Généralités

Le plus simple pour comprendre comment s'organise le code est de suivre les grandes lignes de son exécution.

Le point d'entrée de l'application est le fichier `boites.adb`. On l'appelle en lui fournissant sur `stdin` les paramètres de la commande.

On va alors dans un premier temps faire appel au module `io` pour analyser les paramètres fournis et créer, à l'aide du module `boite`, une représentation abstraite de cette commande, sous forme d'enregistrement.

Puis, toujours avec le module `boite`, on crée une représentation abstraite de la boite à partir de la commande, toujours sous forme d'enregistrement, même si une classe serait plus appropriée.

Enfin, on fait derechef appel au module `io` pour enregistrer la boite sous forme de fichier SVG, à l'aide du module `svg`.

2 Gestion des E/S

Pour lire la commande, il suffit de constater que les paramètres vont par paires : un intitulé et une valeur. On les récupère donc deux par deux puis stocke la valeur dans la variable correspondant à l'intitulé.

3 Représentation d'une boite

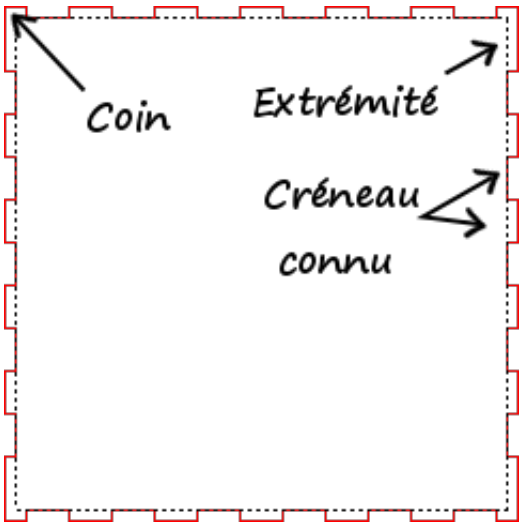
Comme en compilation, il a été décidé de passer par une représentation intermédiaire de la boite en vue d'étendre le programme. Par exemple, `pandoc` fonctionne de cette manière : il construit un arbre syntaxique à partir du fichier d'entrée avant de générer celui de sortie.

Une boite est un ensemble de trois pièces de deux types : extérieure ou intérieure. On peut donc se contenter de deux attributs.

Une pièce est un ensemble de cinq facettes de trois types : selon la longueur, selon la largeur et le (pla)fond. Là encore, trois attributs suffisent.

Une facette se caractérise par quatre coins, chacun étant plein ou creux, et par quatre côtés. Un côté est constitué d'un créneau (une encoche ou un trou)

d'une longueur à déterminer, d'une suite de créneaux de longueur définie par l'utilisateur et d'un dernier créneau de taille à calculer. Par exemple :



Comme il est compliqué de travailler avec des types non contraints, on se contente de stocker la longueur et le type des créneaux (plein ou creux) aux extrémités et le nombre de créneaux de taille connue ainsi que le type majoritaire, pour savoir par lequel on commence.

On stocke les coins indépendamment des côtés pour qu'il n'y ait pas de redondance (un coin appartiendrait à deux côtés sinon).

4 Génération du SVG

Le SVG est généré sous forme de chaînes de caractères afin de pouvoir l'afficher de multiples manières (à l'écran, dans un fichier, etc.). Seulement, comme le taille de ces chaînes n'est pas définie à l'avance, il a fallu utiliser des *unbounded strings*.

Afin de limiter leur manipulation, nous nous sommes basés sur le fonctionnement du module `matplotlib` en Python et avons utilisé une machine à état. Concrètement, un fichier SVG est représenté par une variable `contents` appartenant au module et de type `Unbounded_String`. On la complète par concaténation au fur et à mesure des appels aux fonctions (`header`, `polygon`, etc.).