3 La génération de PDF

À partir du programme qui permet de générer du SVG, nous voulons maintenant construire un document PDF. Nous réutiliserons les classes d'objets géographiques, la lecture des données au format MIF et le découpage en quartiles vus lors des cours précedents. Nous ajouterons des méthodes permettant d'afficher les objets géographiques au format PDF ainsi que l'utilisation de rectangles englobants et le changement de repère.

3.1 Pourquoi générer du PDF ?

Le PDF est un format binaire créé par Adobe Systems qui permet d'intégrer des images bitmap, du dessin vectoriel et du texte dans le même document. Il s'agit d'un format propriétaire dont les spécifications sont publiques. Le format PDF est utilisable librement.

Les deux formats SVG et PDF sont assez complémentaires. Pour intégrer un document vectoriel dans une page web, pour réutiliser le contenu d'un document ou pour intégrer de la dynamicité dans le document, il est préférable d'utiliser le SVG. Cependant, le format PDF constitue une meilleur solution dans le cas où il faut que le document soit imprimable, visualisable simplement et que le contenu soit visible par tous de la même façon,

Pour un site internet utilisant la cartographie, il pourrait être intéressant d'utiliser les deux formats. SVG et HTML pour visualiser les informations et pour interagir avec l'utilisateur et PDF pour une sortie d'un document à imprimer.

Le format RML (ReportML) de ReportLab permet de décrire un document en XML et le transformer automatiquement en PDF, cela peut constituer un niveau intermédiaire de document qui possède les avantages du XML tout en pouvant être transformé en PDF.

3.2 Le format PDF

Nous allons utiliser la bibliothèque REPORTLAB pour générer directement des documents PDF. Le site web *reportlab.org* donne accès à une documentation très complète de cette bibliothèque. Dans ce cours, nous n'utiliserons qu'une petite partie des fonctions mises à notre disposition.

Le code de la figure 13 permet de voir comment générer un fichier PDF de taille A4 dans lequel il est écrit *Hello World* à l'emplacement (100mm, 100mm) à partir du coin en bas à gauche de la page.

Dans ce code, $myCanvas$ représente le document PDF. La ligne 5 permet de déclarer ce document. La ligne 6 affiche la chaine de caractère dans le document. La ligne 7 déclare une fin de page (ce qui n'est pas nécessaire pour ce document car il ne contient qu'une seule page). La ligne 8 génère le fichier *hello.pdf* dans le répertoire courant.

L'exemple de la figure 14 illustre l'utilisation des chemins (*paths*) pour dessiner des polygones. La méthode `beginPath` créée un nouveau chemin. Le début du chemin est défini grâce à l'intruction `moveTo` (ligne 8). Ensuite les lignes sont traçées grâce à l'instruction `lineTo` (ligne 13). La méthode `close` de la ligne 14 permet de fermer le chemin. Un chemin qui reste ouvert représenterait une polyligne. Enfin, la méthode `drawPath` permet de dessiner le chemin dans le canvas.

La ligne 23 change la couleur de remplissage. Les trois paramètres sont de nombres entre 0 et 1 qui définissent respectivement les trois composantes : rouge, vert et bleu. Toutes les couleurs peuvent être définies à partir de ces trois composantes. Les lignes 24 et 25 servent respectivement à dessiner le disque et le cadre.

La figure 15 montre le résultat de l'exécution du code précédent.

3.3 Le changement de repère

L'inconvénient du format PDF est qu'il faut transformer les coordonnées originales (en Lambert 2) vers les dimensions de la feuille de travail (qui peut être par exemple au format A4). Dans ce cas, il faut donc procéder à un changement de repère.

Rectangles englobants et changement de repère

Le changement de repère sera effetué en 3 étapes :
— calcul des dimensions du cadre à représenter ;
— calcul des dimensions du cadre vers lequel représenter les données ;

— création d'un objet permettant de transformer les coordonnées d'un objet en fonction des dimensions précedemment calculées.

Pour représenter les dimensions, nous utiliserons des rectangles englobants aussi appelé *bounding box* (cf : figure 16). Les cotés d'un rectangle englobant sont parallèles aux axes. Ils peuvent donc être définis par 4 valeurs : $x_{min}, y_{min}, x_{max}$ et y_{max}.

L'interface de la classe *BoundingBox* est décrite dans la figure 17. Ces *BoundingBox* doivent pouvoir être calculés par les objets géographiques et les géométries. Par conséquent, il faut rajouter à la classe `OjbetGeographique` et aux classes `Geometrie` des méthodes `getBoundingBox` qui retournent les rectangles englobants de ces objets.

La boundingbox de la zone dans laquelle dessiner peut être définie manuellement en utilisant par exemple sur la taille du format A4 (21cm * 29.7cm). En se basant sur les deux rectangles englobants, il est possible de définir un objet effectuant le changement de repére automatiquement. Ce transformeur de coordonnées (cf. figure 18 implémente une méthode `transforme` qui prend en argument une coordonnée dans le repère défini par la première *boundingbox* et retourne l'image de cette coordonnée dans le repère défini par la seconde *boundingbox*. La méthode `transforme` préserve les proportions originales. L'algorithme de cette méthode est donné par la figure 19

La génération du document PDF

L'algorithme de la figure 20 décrit une méthode qui permet de dessiner un ensemble d'objets géographiques sur une feuille au format A4. Cet algorithme se base les méthodes `getBoundingBox` et `toPDF` de la classe `ObjetGeographique`.

Exercice de TD. Écrivez le code :

— de la classe `BoundingBox`;
— de la classe `Transformeur`;
— de la fonction `ecrirePDF`;
— des méthodes `getBoundingBox` pour la classe `ObjetGeographique` et pour les classes de type `Geometrie` (surtout pour la classe `Polygone`);
— des méthodes `toPDF` pour la classe `ObjetGeographique` et pour les classes de type `Geometrie` (surtout pour la classe `Polygone`).

Exercice de TP. Implémentez les questions de la partie précedente.

Rajoutez le titre, le nom de l'auteur, et la direction du nord dans le document généré.

Vous pourrez aussi rajouter l'échelle du document et bien entendu la génération automatique de la légende.

Optimisez la méthode de transformation des coordonnées de la classe `Transformeur`.

```
1 from reportlab.pdfgen import canvas
2 from reportlab.lib.pagesizes import A4
3 from reportlab.lib.units import mm
4
5 myCanvas = canvas.Canvas("hello.pdf", pagesize=A4)
6 myCanvas.drawCentredString(100*mm,100*mm,"Hello World")
7 myCanvas.showPage()
8 myCanvas.save()
```

FIGURE 13 – Génération d'un *hello world* en pdf.

```
from reportlab.pdfgen import canvas
from reportlab.lib.pagesizes import A4
from reportlab.lib.units import mm
from math import pi, cos, sin

def pentagone(aCanvas, xcenter, ycenter, radius, fill=False):
    p = aCanvas.beginPath()
    p.moveTo(xcenter+radius,ycenter)
    for i in xrange(5):
        angle = (2*pi*i)/5
        x = xcenter+cos(angle)*radius
        y = ycenter+sin(angle)*radius
        p.lineTo(x, y)
    p.close()
    aCanvas.drawPath(p, fill=fill)

myCanvas = canvas.Canvas("pentagone.pdf", pagesize=(120*mm,120*mm))
pentagone(myCanvas, 60*mm, 60*mm, 50*mm)
pentagone(myCanvas, 60*mm, 60*mm, 40*mm)
pentagone(myCanvas, 60*mm, 60*mm, 30*mm)
pentagone(myCanvas, 60*mm, 60*mm, 20*mm)
pentagone(myCanvas, 65*mm, 65*mm, 5*mm, fill=True)
myCanvas.setFillColorRGB(0.5,0.5,0.5)
myCanvas.circle(15*mm, 15*mm, 5*mm, fill=True)
myCanvas.rect(0, 0, 120*mm, 120*mm)
myCanvas.showPage()
myCanvas.save()
```

FIGURE 14 – Génération de pentagone en pdf

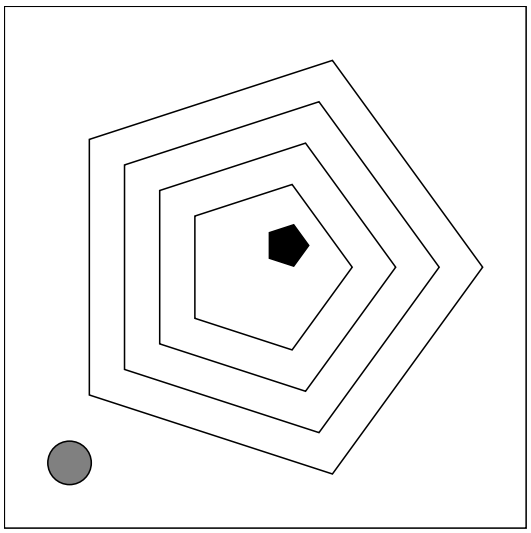

FIGURE 15 – Résultat de la génération des pentagones

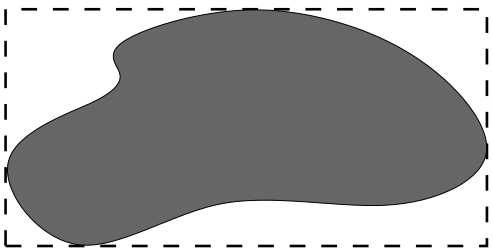

FIGURE 16 – Le rectangle englobant associé à une figure (en pointillé)

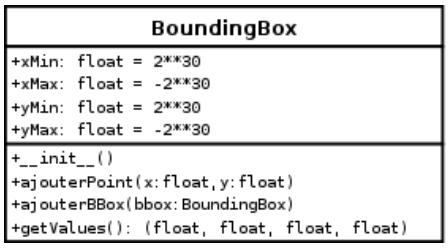

FIGURE 17 – La classe BoundingBox

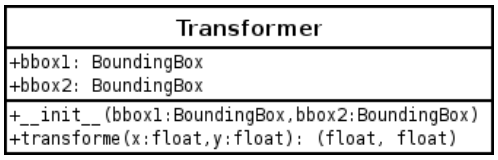

FIGURE 18 – La classe Transformer

```
Fonction: "conversion"
Argument
    bbox1 = la bounding-box representant le repere d'origine
    bbox2 = la bounding-box representant le repere cible
    x, y = la coordonnee a convertir
Retourne
    xc, yc = la coordonnee convertie
Debut
    x1, y1, l1, h1 = bbox1.getValues()
    x2, y2, l2, h2 = bbox2.getValues()
    si l1/h1 > l2/h2 alors
       ratio = l2/l1
    sinon
       ratio = h2/h1
    xc = (x-x1)*ratio + x2
    yc = (y-y1)*ratio + y2
    Retourne (xc, yc)
Fin
```

FIGURE 19 – Algorithme de changement de repère

```
Fonction: "ecrirePDF"
Argument
    documentGeographiques: La liste des objets geographiques a representer.
    methodeDeColoration: La methode de coloration des objets geographiques
                (par exemple par des quartiles)
    nomFichPDF: le nom du fichier PDF a generer
Retourne
    rien
Debut
    bbox1 = BoundingBox()
    PourChaque objGeographique dans lstObjGeographiques:
        bbox = objGeographique.getBoundingBox()
        bbox1.ajouterBBox(bbox)
    bbox2 = BoundingBox()
    bbox2.ajouterPoint(0,0)
    bbox2.ajouterPoint(21cm, 29.7cm)
    transformeur = Transformer(bbox1, bbox2)
    myCanvas = Canvas(nomFich,pagesize=(21cm, 29.7cm))
    PourChaque objGeographique de documentGeographique:
        objGeographique.toPDF(myCanvas, methodeDeColoration, transformeur)
    myCanvas.save()
Fin
```

FIGURE 20 – Algorithme de dessin d'une carte en PDF