Projet

Savez-vous dessiner en bash ?

1 ASCII Art

On veut écrire un script bash capable de produire un dessin composé de :

- points
- droites horizontales
- rectangles pleins

On considèrera que l'image est un tableau de L par H pixels. Un pixel est ici un simple caractère qui sera affiché à l'écran. On utilisera par exemple l'espace pour le blanc et le # pour le noir. Tout autre caractère est bien sûr utilisable.

- Quelle structure de données faut-il utiliser pour l'image ? On l'appellera `Image`
- Ecrire un script bash `Point` dont le but est de remplir un pixel donné avec le caractère donné
- Ecrire un script bash `Trait` dont le but est de tracer une droite horizontale d'ordonnée Y, commençant en X1 et finissant en X2. Faire appel au script `Point`
- Ecrire un script `Rectangle` qui trace un rectangle de diagonale (X1,Y1)-(X2,Y2) en utilisant le script `Trait`
- Faire, dans un script bash, un petit dessin utilisant tous ces types de primitives (points, traits, rectangles).

2 Fichiers séquentiels

Plutôt que d'avoir à écrire un script pour chaque nouveau dessin, nous allons décrire la scène à dessiner dans un petit langage très simple, qui sera écrit dans un fichier texte. Ce fichier sera lu et analysé par la nouvelle version de votre script. Si le fichier de description le spécifie, l'image sera elle aussi écrite dans un fichier plutôt que d'être affichée à l'écran.

Le fichier de description est un fichier textuel dont chaque ligne est soit :

- une commande de tracé de point, avec la syntaxe :
  P x y c
  Où x et y sont des entiers et c, le caractère qui doit être dessiné
- une commande de tracé de rectangle, avec la syntaxe :
  R x1 y1 x2 y2 c
  Où x1 ... y2 sont des entiers et c, le caractère qui doit être dessiné

- une commande indiquant le nom du fichier de sortie :
  S nom-fichier

Chaque ligne commence donc par un caractère qui décrit le type de commande, suivi d'un certain nombre de paramètres séparés par un nombre quelconque de blancs. Les lignes commençant par d'autres caractères que P, R ou S sont simplement ignorées (elles peuvent ainsi être utilisées pour des commentaires). Les lignes vides doivent elles aussi être ignorées.

1.

Ecrire le script correspondant aux lignes commençant par P et R et pour chaque ligne lue, appeler le script correspondant déjà écrit dans l'exercice précédent.

2.

Traiter ensuite le cas de la commande S. Si elle n'existe pas dans le fichier de description, on affiche le dessin à l'écran. Si elle existe, on crée un nouveau fichier et le résultat est produit dans ce fichier.

3.

Si plusieurs commandes S sont présentes dans le fichier, on considère que les commandes R et P entre chaque S s'appliquent à des dessins différents qui seront produits dans des fichiers distincts.

3 Fichier graphique à la norme PPM

Une image peut être considérée comme une grille de points ou pixels ayant chacun sa propre couleur, mélange de rouge, de vert et de bleu. Le nombre de pixels par ligne est appelé largeur de l'image et le nombre de lignes sa hauteur. Un fichier PPM est un fichier de texte qui doit comporter un en-tête comportant 4 indications séparées par un ou plusieurs espaces ou bien un saut de ligne :

- Un code qui sera toujours P3 (il s'agit d'un fichier texte);
- La largeur de l'image (nombre de pixels par ligne)
- La hauteur de l'image (nombre de lignes)
- Le nombre de niveaux de couleurs (Ex : 10. Avec cette valeur, on pourra représenter 103=1000 couleurs)

puis une série de lignes décrivant (largeur * hauteur) pixels. Chaque pixel contient 3 valeurs décimales entre 0 et le nombre de niveaux de couleurs (ex : 10). On part du coin supérieur gauche de la grille de pixels, et on écrit de gauche à droite les 3 valeurs de chacun des pixels, en séparant chaque valeur d'au moins un espace et en allant à la ligne dans le fichier PPM à chaque changement de ligne dans la grille de pixels. Les trois valeurs pour chaque pixel représentent le dosage de rouge, de vert et de bleu, respectivement. Une valeur 0 indique que la couleur n'est pas utilisée, et la valeur maximale (ex : 10) indique que la

couleur est saturée. Par exemple, si le nombre de niveaux de couleurs est fixé à 10, la couleur noire sera représentée par 0 0 0, la couleur rouge par 10 0 0, la couleur verte par 0 10 0 et le blanc par 10 10 10.

Voici maintenant un exemple de fichier PPM, dans lequel vous remarquerez que les lignes commençant par le symbole # (dièse) sont des commentaires.

```
P3
#exemple.ppm
4 4
8
0 0 0 0 0 0 0 0 0 0 10 0 10
0 0 0 0 10 5 0 0 0 0 0 0
0 0 0 0 0 0 0 10 5 0 0 0
10 0 10 0 0 0 0 0 0 0 0 0
```

L'extension du nom des fichiers PPM sera .ppm.

4.

Soit, dans un repère (O, i, j) du plan, un cercle C de centre $G(x_0, y_0)$ et de rayon R. Un point M quelconque du plan, de coordonnées (x, y), appartient au cercle C si et seulement si ses coordonnées vérifient l'équation du cercle, c'est à dire si et seulement si :

$$(x - x_0)^2 + (y - y_0)^2 = R^2$$

Pour cet exercice, nous allons donc considérer que le rayon R est fixé (on essaiera les valeurs 50, 100 et 150). L'origine du repère, O, correspond au coin supérieur gauche de l'image. Notez que l'axe des ordonnées est dirigé vers le bas par commodité (pour ne considérer que des points de coordonnées positives). La largeur (ainsi que la hauteur) de l'image est $2 * R + 2 * marge$, où $marge$ correspond au bord de l'image (on pourra prendre $marge = R/10$, par exemple). Le centre G du cercle sera considéré au milieu de l'image, en $(R + marge, R + marge)$.

Pour tracer le cercle, il suffit maintenant de considérer chaque pixel, un par un et de regarder si ses coordonnées (x, y) vérifient l'équation du cercle. Si c'est le cas, le pixel est colorié en vert, sinon en noir :

```
Algo tracer
Debut
  Pour y allant de 1 a hauteur faire
    Pour x allant de 1 a largeur faire
      Si verifie_equation(x, y) alors colorier le pixel en vert
      Sinon colorier le pixel noir
      Fin Si
    Fait
  Fait
Fin algo tracer
```

Pour palier au problème de pointillés éventuels sur le bord du cercle, vous utiliserez l'équation suivante pour savoir si un point (x, y) est sur le bord du cercle ou non :

$$R^2 - \varepsilon \leq (x - x_0)^2 + (y - y_0)^2 \leq R^2 + \varepsilon$$

où ε est une valeur petite par rapport à R^2 (prendre $\varepsilon = 2 * R$ donne un bord assez épais, essayez plusieurs valeurs pour ε).

Ecrire un script `Cercle` qui trace un cercle de centre G et de rayon R en utilisant ce qui précède.

5.

Modifiez vos scripts afin de tracer des droites horizontales et/ou verticales, des rectangles pleins ou creux, des triangles pleins ou creux, des cercles pleins ou creux et même du remplissage au hasard si vous le souhaitez.

Enfin, écrivez un script utilisant tous les scripts précédants et permettant de générer une image PPM représentant une scène précise complexe (paysage, nature morte, objets, ...)

4 A rendre

Ce projet est à faire en binôme.

Ce projet doit contenir un rapport, dans lequel vous devrez décrire le sujet du projet, les algorithmes utilisés dans vos scripts et les dessins obtenus. Votre projet (et pas le rapport) doit également contenir vos scripts commentés et les images obtenues. Le rapport est à rendre sous forme de fichier (.txt, .doc, .tex ou .html et .pdf dans tous les cas). Les fichiers (rapport, scripts et images) sont à rendre sur l'ENT, dans une archive zip, tar.gz ou tgz, le vendredi 28 avril 2017 au plus tard.