move <dx> <dy> <ims> <imd> (p. 2)
scroll <dx> <dy> <ims> <imd> (p. 2)
flip <dir> <ims> <imd> (p. 3)
domain.c (p. 4)
shear <dir> <angle> <ims> <imd> (p. 4)
rotate <x> <y> <angle> <ims> <imd> (p. 5)
wave <tx> <ax> <ty> <ay> <ims> <imd> (p. 6)
sphere <ims> <imd> (p. 7)

move <dx> <dy> <ims> <imd>

Écrire le programme `move.c`

— le programme permet de charger une image donnée en paramètre `ims` et sauve le résultat dans l'image nommée `imd`
— le programme permet de réaliser une translation dans les directions x et y, d'une quantité `dx` et `dy`
— visualiser les images produites via `pvisu`

`./move 0 30 ../data/pepper.ppm a.ppm`

`./move 30 0 ../data/pepper.ppm a.ppm`

`./move 30 30 ../data/pepper.ppm a.ppm`

`./move -30 -30 ../data/pepper.ppm a.ppm`

scroll <dx> <dy> <ims> <imd>

Écrire le programme `scroll.c`

— le programme permet de charger une image donnée en paramètre `ims` et sauve le résultat dans l'image nommée `imd`
— le programme permet de réaliser une translation dans les directions x et y, d'une quantité `dx` et `dy`
— par rapport au programme move, les parties non visibles sont recopiées : les images sont considérées comme périodiques
— visualiser les images produites via `pvisu`

`./scroll 0 30 ../data/pepper.ppm a.ppm`

`./scroll 30 0 ../data/pepper.ppm a.ppm`

`./scroll 30 30 ../data/pepper.ppm a.ppm`

`./scroll -30 -30 ../data/pepper.ppm a.ppm`

flip <dir> <ims> <imd>

Écrire le programme `flip.c`

- le programme permet de charger une image donnée en paramètre `<ims>` et sauve le résultat dans l'image nommée `imd`
- le programme permet de réaliser un effet *miroir* selon la direction donnée par `dir`. `dir`∈`{h, v, t}` est un caractère permettant de déterminer l'orientation de la transformation (c.-à-d., `h` pour un flip horizontal, `v` pour un flip vertical et `t` pour une transposition : ligne vers colonne)
- **le programme accepte différentes combinaisons de paramètres, par exemple : vht, hhh, etc**
- visualiser les images produites via `pvisu`

`./flip h ../data/pepper.ppm a.ppm`

`./flip v ../data/pepper.ppm a.ppm`

`./flip vh ../data/pepper.ppm a.ppm`

`./flip hh ../data/pepper.ppm a.ppm`

`./flip t ../data/pepper.ppm a.ppm`

`./flip ht ../data/pepper.ppm a.ppm`

Fonction `bilinear_interpolation` du module `domain.c`

Écrire la fonction `bilinear_interpolation` du module `domain.c`

— cette fonction permet de réaliser une interposlation bilinéaire. Soit (i', j') un pixel dans l'image transformée I' et (y, x) ses coordonnées inverses dans l'image initiale I. Pour obtenir la nouvelle valeur $I'(i', j')$, la méthode d'interpolation bilinéaire peut utilisée

— soit $(i, j) = (\lfloor y \rfloor, \lfloor x \rfloor)$, les coordonnées entières dans I et si $dx = x - j$ et $dy = y - i$ alors la nouvelle veleur de $I'(i', j')$ donnée l'interpolation bilinéaire est

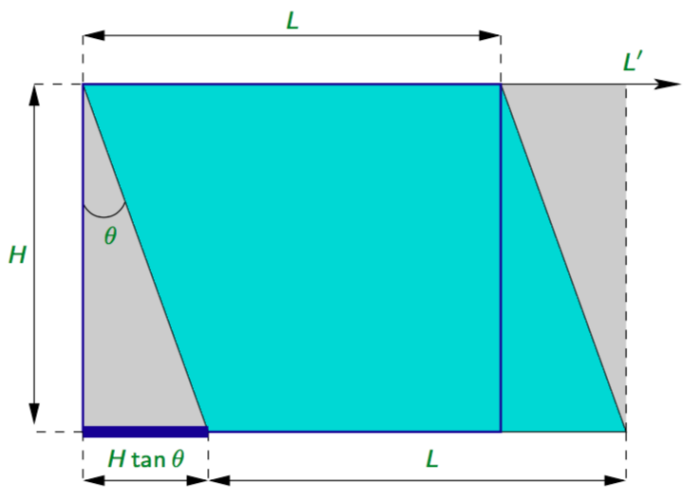
$$I'(i', j') = (1 - dx).(1 - dy).I(i, j) + dx.(1 - dy).I(i, j + 1) + (1 - dx).dy.I(i + 1, j) + dx.dy.I(i + 1, j + 1)$$

`shear <dir> <ims> <imd>`

Écrire le programme `shear.c`

— le programme permet de charger une image donnée en paramètre `<ims>` et sauve le résultat dans l'image nommée `imd`

— le programme permet de réaliser un cisaillement dans la direction `dir`∈`{h, v}` c.-à-d. horizontal (*cf* figure) ou vertical et avec un angle θ exprimé en degré et est un nombre flottant



— le programme utilise la fonction d'interpolation bilinéaire et une transformation inverse

— visualiser les images produites via `pvisu`

`./shear h 30 ../data/pepper.ppm a.ppm`

`./shear h -30 ../data/pepper.ppm a.ppm`

`./shear v 30 ../data/pepper.ppm a.ppm`

`./shear v -30 ../data/pepper.ppm a.ppm`

rotate <x> <y> <angle> <dir> <ims> <imd>

Écrire le programme `rotate.c`

— le programme permet de charger une image donnée en paramètre `ims` et sauve le résultat dans l'image nommée `imd`

— le programme permet de réaliser un rotation de centre donné par `x`, `y` et d'angle `angle` exprimé en degré et est un nombre flottant

— La rotation d'un point (x, y) autour du point (x_0, y_0) avec un angle θ peut être exprimée sous la forme matricielle suivante

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x - x_0 \\ y - y_0 \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} \tag{1}$$

— le programme utilise la fonction d'interpolation bilinéaire et une transformation inverse

— visualiser les images produites via `pvisu`

`./rotate 0 0 50 ../data/pepper.ppm a.ppm`

`./rotate 0 0 -50 ../data/pepper.ppm a.ppm`

`./rotate 50 50 -50 ../data/pepper.ppm a.ppm`

`./rotate 128 64 -30 ../data/pepper.ppm a.ppm`

Rotation avec multiple `shear` : `rotate 128 64 -30 pepper.ppm` $\simeq 3\times$`shear`

La rotation peut également se réaliser avec le programme shear. Tester l'enchaînement proposé

`./shear h -15 ../data/pepper.ppm a.ppm`

`./shear v 26.57 b.ppm c.ppm`

`./shear h -15 d.ppm e.ppm`

wave <tx> <ax> <ty> <ay> <ims> <imd>

Écrire le programme `wave.c`

— le programme permet de charger une image donnée en paramètre `ims` et sauve le résultat dans l'image nommée `imd`

— le programme permet de réaliser une transformation de type *vague* où les périodes et amplitudes sont données par les paramètres `tx`, `ax`, `ty`, `ay` pour les directions x et y

— le programme utilise une transformation inverse et peut être exprimée sous la forme suivante

$$x = x' + a_x \sin(\frac{2\pi.y'}{t_x}) \quad \text{et} \quad y = y' + a_y \sin(\frac{2\pi.x'}{t_y})$$

— le programme utilise la fonction d'interpolation bilinéaire

— visualiser les images produites via `pvisu`

— la rotation peut également se réaliser avec le programme shear (*cf* le fichier de résultat). Tester l'enchaînement proposé

`./wave 120 10 250 15 ../data/lena-color.ppm a.ppm`

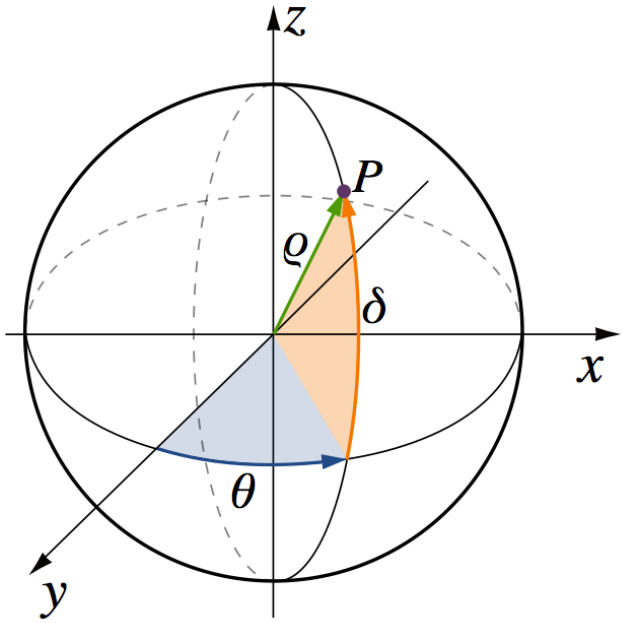
`./wave 128 20 1 1 ../data/lena-color.ppm a.ppm`

`./wave 1 1 128 20 ../data/lena-color.ppm a.ppm`

sphere <ims> <imd>

Écrire le programme `sphere.c`

- le programme permet de charger une image donnée en paramètre `ims` et sauve le résultat dans l'image nommée `imd`
- le programme permet de réaliser une projection de l'image source sur une sphere (*cf* le fichier de résultat pour un exemple). Trouver comment utiliser les coordonnées sphérique
- le programme utilise la fonction d'interpolation bilinéaire et une transformation inverse
- visualiser les images produites via `pvisu`
- si vous générez une série d'images, une animation `gif` peut être créée en utilisant la commande `convert` (par ex. `earth.gif`). Écrire un script shell `animate.sh` qui utilise vos programmes et qui permet de réaliser cette animation. Un exemple de résultat est visible via un lien sur la page `Moodle` du cours.

`../data/earth.ppm`

`./sphere ../data/earth.ppm a.ppm`