TD10 – Infographie

Langage C (LC4)

On utilise les définitions suivantes.

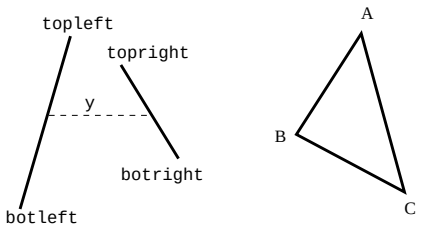
```
typedef struct { int x; int y; } pixel;
typedef pixel triangle[3];
typedef struct { int R; int G; int B; } color;

// valeur maximale d'une coordonnee RGB de couleur
const int max_color = 255;

// affiche un pixel
void putpixel(int x, int y, color c)};
```

L'objectif de ce TD est, dans la première partie, d'écrire une fonction qui parcourt tous les pixels contenus dans un triangle, et dans la suite d'utiliser cette fonction pour diverses applications.

1 Remplissage d'un triangle

Question 1. Écrire une fonction `int interpolx(pixel A, pixel B, int y)` qui renvoie l'abscisse du point d'ordonnée `y` sur le segment `A` – `B`.

Question 2. Écrire une fonction `scanx` qui parcourt de gauche à droite les pixels d'un segment horizontal entre deux segments non horizontaux qui ne s'intersectent pas, comme sur la figure de gauche ci-dessous.
Cette fonction prend les paramètres suivants :
– quatre valeurs de type `pixel` et de noms `topleft`, `botleft`, `topright`, `botright`, désignant deux segments comme indiqués sur la figure ;
– un entier `y` indiquant l'ordonnée du segment horizontal à parcourir.
La fonction affichera les coordonnées de chaque pixel parcouru.



Il « suffit » maintenant d'utiliser cette fonction autant de fois que nécessaire entre les arêtes AB et AC du triangle, puis entre BC et AC (figure de droite). La principale difficulté réside

dans les diverses configurations possibles du triangle : ordre des points, sommet du bas situé à gauche ou à droite du sommet du milieu, segments horizontaux etc.

Question 3. Écrire une fonction `swapy(pixel *, pixel *)` qui échange les pixels pointés si l'ordonnée du premier est plus grande que celle du deuxième.

Question 4. Écrire une fonction :
`void top_mid_bot(triangle *adrT, pixel *ptop, pixel *pmid, pixel *pbot)`
qui affecte aux pixels pointés les valeurs des trois sommets du triangle de sorte que `*ptop` est le (ou un des) sommet(s) le plus haut, `*pbot` est le (ou un des) plus bas et `*pmid` est entre les deux.

Question 5. Écrire une fonction :
`int mid_is_left(pixel top, pixel mid, pixel bot)`
qui renvoie 1 si `mid` est à gauche du segment `top` – `bot` et 0 sinon.

Question 6. Écrire la fonction principale `void scan_triangle(triangle *adrT)` qui affiche les coordonnées de tous les pixels contenus dans le triangle dont l'adresse est passée en paramètre. Les étapes sont les suivantes :
- définir trois pixels `top`, `mid` et `bot` et les positionner grâce à la fonction `top_mid_bot`) ;
- déterminer la position horizontale de `mid` avec `mid_is_left` ;
- utiliser la fonction `scanx` sur toutes les lignes entre les segments `top` – `mid` et `top` – `bot`, puis entre les segments `mid` – `bot` et `top` – `bot`. Dans les deux cas, vérifier auparavant que `mid` n'est pas à la même hauteur que `top` ou `bot`, auquel cas l'étape correspondante doit être sautée.

Question 7. Modifier la fonction précédente pour qu'elle prenne en paramètre un pointeur sur fonction, de prototype `void (*do_pixel)(int, int)`, qui donnera les opérations à effectuer pour chaque pixel.

Question 8. Écrire le code nécessaire pour appeler la nouvelle version de la fonction en obtenant le même comportement que la première version (affichage des coordonnées).

2 Coordonnées barycentriques

On souhaite afficher une palette de couleurs à l'intérieur d'un triangle équilatéral. Les trois sommets correspondront aux trois couleurs rouge, vert, bleu respectivement.
On utilisera les définitions suivantes, pour lesquelles un point est défini par ses coordonnées barycentriques à l'intérieur d'un triangle :

```
struct st_point
{
  double alpha, beta, gamma;
  triangle tri;
};
typedef struct st_point point;

pixel pixel_from_point(point);
point point_from_pixel(pixel, triangle *);
```

La fonction `point_from_pixel` permet donc de récupérer les coordonnées barycentriques d'un pixel à partir de ses coordonnées cartésiennes.

L'idée des coordonnées barycentriques dans un triangle est d'indiquer en quelles proportions un point est proche de chacun des trois sommets. Ainsi, le barycentre du triangle a ses trois coordonnées égales, les sommets ont chacun deux coordonnées à 0. On prend souvent comme convention que la somme des coordonnées vaut 1.

Question 9. Écrire une fonction `color color_from_bary(double alpha, double beta, double gamma)` qui construit une couleur à partir de trois coordonnées barycentriques de sorte que trois coordonnées égales produisent du gris et deux coordonnées à 0 produisent une couleur primaire.

Question 10. Utiliser la fonction précédente et la fonction `scan_triangle` pour afficher la palette de couleurs.

3 Images de synthèse

Pour manipuler et afficher un objet en trois dimensions, on procède souvent par triangulation : on décompose sa surface en plein de petits triangles et pour chaque opération effectuée sur l'objet (déplacement, affichage, éclairage…), on itère cette opération sur tous les triangles qui le composent.
On supposera ici qu'on dispose :
– d'un type `triangle3D` ;
– d'une fonction `triangle projette(triangle3D)` qui calcule la projection en 2D d'un triangle en 3D ;
– d'une fonction `triangle3D *visibles(triangle3D *)` qui, à partir d'un tableau de triangles en 3D, renvoie un tableau contenant exactement tous les triangles visibles (non cachés par d'autres triangles) ;
– d'une fonction `double intensity(triangle3D T)` qui renvoie l'intensité lumineuse réfléchie sur le triangle `T` par la lumière ambiante [1], sous la forme d'une valeur entre 0 (aucune lumière) et 1 (pleine lumière).

Question 11. Écrire une fonction `color attenuated_color(color c, float intensity)` qui diminue la couleur passée en paramètre en fonction de l'intensité lumineuse.

Question 12. Écrire une suite d'instructions permettant, à partir d'un objet triangulé et de sa couleur, d'afficher la projection de tous ses triangles visibles en tenant compte de l'intensité lumineuse renvoyée par chacun d'eux. Vous utiliserez pour cela la fonction `scan_triangle`.

Question 13. Qu'est-ce qui empêche de faire de cette suite d'instructions une fonction, prenant comme paramètres la liste des triangles en 3D et la couleur ? Proposez une modification de la fonction `scan_triangle` qui le permette.

1. en pratique, c'est une combinaison de produits vectoriels faisant intervenir le vecteur normal au triangle, la position de la source lumineuse, la position de l'observateur, et éventuellement la position d'autres objets autour…