Travaux dirigés d'infographie n°7

Cours de synthèse d'image I

—IMAC première année —

Lancer de rayons/Ray Tracing

Le TP est noté par binôme. La durée est de 3 heures. Tous les documents sont autorisés.

▸ **Exercice 1.** ***Lecture du fichier de description de la scène***

```
typedef struct{
  point3_t position;
  vecteur3_t visee;
  vecteur3_t haut;
  float distance_focale;
  float dimension_u;
  float dimension_v;
  int du;
  int dv;
}camera_t;
```

Étant données les structures du TP précédent et la définition du type `camera_t` *ci-dessus, écrire les fonctions,*

+ `char *objet_generique_vers_string(objet_generique_t *plan, char *buffer);`

+ `char *plan_vers_string(plan_t *plan, char *buffer);`

+ `char *sphere_vers_string(sphere_t *sphere, char *buffer);`

+ `char *camera_vers_string(camera_t *sphere, char *buffer);`

, qui "écrivent" à l'emplacement pointé par `buffer`*, une chaîne de caractères, représentant la structure, dont l'adresse est le premier paramètre de la fonction. Plus précisement, la conversion d'un* `plan_t` *est une chaîne de type* `"p a b c d"`*, d'une* `sphere_t` *une chaîne de type* `"s x y z r"`*, et d'une* `camera_t`*, une chaîne de type* `"c x y z vx vy vz hx hy hz f dimension_u dimension_v du dv"`*. Le premier caractère d'une chaîne est un code, les autres sont les valeurs des champs de la structure. On suppose que le buffer transmis, pointé par* `buffer`*, est de taille suffisante. Toutes les*

fonctions renvoient l'adresse `buffer`*.*

Écrire les fonctions réciproques des fonctions précédentes, retournant des structures initialisées à partir de leurs représentations "chaîne de caractères",

+ `objet_generique_t *string_vers_objet_generique(char *buffer);`

+ `plan_t *string_vers_plan(char *buffer);`

+ `sphere_t *string_vers_sphere(char *buffer);`

+ `camera_t *string_vers_camera(char *buffer);`

, on suppose que la chaîne de caractère transmise représente une structure valide.

Écrire la fonction,

```
+ void charger_scene(char *nom_fichier,
    objet_generique_t **objets, camera_t *camera);
```

qui lit dans le fichier `nom_fichier` *un ensemble d'objets génériques et une caméra. On suppose que le fichier ne contient qu'une seule caméra. Par la suite, on appellera l'ensemble objets/caméra, la scène.*

Écrire le programme **rtrace** *qui prend en paramètre (de la ligne de commande) un nom de fichier décrivant la scène.* **rtrace** *affiche sur la sortie standard les differents éléments de la scène, en commençant par la caméra. Un fichier de test* `incredible.scene` *se trouve sur ma page.*

▸ Exercice 2. *Calculs d'intersections*

Écrire les fonctions,

+ `int intersecte(objet_generique_t obj, rayon_t rayon);`

+ `int intersecte_plan(plan_t plan, rayon_t rayon);`

+ `int intersecte_sphere(sphere_t sphere, rayon_t rayon);`

, qui renvoient `1` *si* `rayon` *intersecte l'objet passé en paramètre,* `0` *sinon.*

▸ Exercice 3. *Placement de la caméra*

Écrire la fonction,

+ `vecteur3_t produit_vectoriel(vecteur3_t a, vecteur3_t b);`

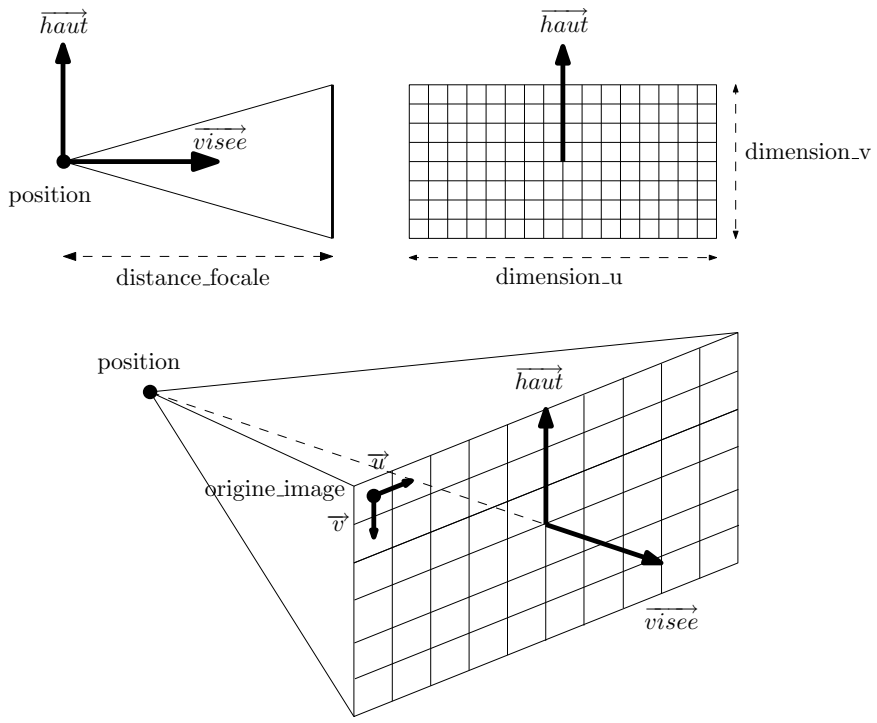

Figure 1: Géométrie de la caméra.

, *qui renvoie le produit vectoriel entre* `a` *et* `b`.

Écrire la fonction,

```
+ void repere_image(camera_t camera,
    point3_t *origine_image, vecteur3_t *u, vecteur3_t *v);
```

, *qui renseigne les structures* `*origine_image`, `*u` *et* `*v` *de sorte qu'elles représentent les grandeurs de la figure 1: le point* `origine_image` *est le centre du pixel d'indice* $(0, 0)$ *dans le système de coordonées de la scène et* $\overrightarrow{u}$ *et* $\overrightarrow{v}$ *sont deux vecteurs tels que,*

```
x = origine_image.x + i*u.x + j*v.x;
y = origine_image.y + i*u.y + j*v.y;
z = origine_image.z + i*u.z + j*v.z;
```

, *sont les coordonnées du centre du pixel d'indice* (i, j) *dans le système de coordonnées de la scène. On supposera que les vecteurs* $\overrightarrow{haut}$ *et* $\overrightarrow{visee}$ *sont transmis normalisés.*

Écrire la fonction,

```
+ rayon_t engendre_rayon(camera_t camera, int i, int j);
```

, *qui renvoie un* `rayon_t` *normalisé, dont l'origine est la position de la caméra dans le système de coordonnées de la scène, et dont la direction est donné par le vecteur de la position de la caméra à la position du pixel d'indice* `(i,j)` *dans le système de coordonnées de la scène .*

▸ **Exercice 4.** ***Lancer de rayon***

Écrire la fonction,

```
+ void set_pixel(unsigned char *pixels, int w, int h,
     int i, int j, unsigned char ndg);
```

, *qui met le pixel d'indice* `(i,j)`, *dans le tableau linéaire* `pixels`, *représentant une image en niveaux de gris et de résolution* `(w,h)`, *au niveau de gris* `ndg`.

Écrire la fonction,

```
+ unsigned char *ray_trace(camera_t camera, objet_generique_t *objets);
```

, *qui renvoie une image "ray-tracé" de la scène passée en paramètre. L'image est synthétisée de la façon suivante. Si le rayon associé à un pixel intersecte un objet, son niveau est mis à blanc. Sinon, il est mis à noir.*

Modifier le programme `rtrace` *afin que, s'il y a deux paramètres, le premier soit le nom du fichier de description de la scène et le deuxième un nom de fichier dans lequel une image "ray-tracé" de la scène sera sauvegardé au format PGM.*