TME numéro 8

(Intersections avec les obstacles)

Le but de ce TME est d'implémenter les fonctions `get_cube_inter_segments` et `inter_cubes`. La première renvoie la liste des segments (les côtés des cubes) qui pourraient avoir une intersection avec un vaisseau spatial donné. La deuxième calcule si, pour une position donnée du vaisseau, voisine de sa position actuelle, le vaisseau a une intersection avec l'un des segments mentionnées ci-dessus. Comme nous l'avons vu en cours, il y a différentes manières de calculer si les balles ou les vaisseaux spatiaux s'intersectent avec les segments. Suivant celle que vous déciderez d'utiliser, vous aurez ou non besoin des fonctions des 3 premiers exercices. Je rappelle que tous ceux-ci ne sont qu'une indication pour vous aider à développer votre projet, vous pouvez vous départir de ces indications si vous le souhaitez (à vos risques et périls).

intersections de base

Exercice 1 Écrivez une fonction :
`left : float * float -> float * float -> float * float -> int`
telle que `(left a b c)` renvoie le nombre « 1 » si `c` est sur la gauche de la droite (`a`,`b`) (vue de a vers b), elle renvoie « -1 » s'il est sur la droite en regardant de a vers b, et « 0 » si les vecteurs $\overrightarrow{\texttt{ac}}$ et $\overrightarrow{\texttt{ab}}$ sont colinéaires. Pour activer votre fonction, pensez à utiliser les « register_xxx » :

```
register_left := left;;
```

Exercice 2 Écrivez une fonction :
`between : float * float -> float * float -> float * float -> bool` telle que `(between a b c)`, où $\overrightarrow{\texttt{ac}}$ et $\overrightarrow{\texttt{ab}}$ sont des vecteurs colinéaires, renvoie `true` si `c` se situe sur le segment [`a`,`b`].

Exercice 3 Écrivez une fonction : `intersect_segment : float * float -> float * float -> float * float -> float * float -> bool` telle que `(intersect_segment a b c d)` renvoie un booléen indiquant si les segments [`a`,`b`] et [`c`,`d`] ont une intersection non vide.

intersections vaisseaux/obstacles

Comme nous l'avons vu en cours, afin de faire glisser les vaisseaux spatiaux le long des obstacles, le serveur est souvent amené à tester si un certain nombre de petits déplacements sont possibles (pas d'intersection avec les obstacles) ou non. Afin de limiter le nombre de tests, on récupère avant d'effectuer ceux-ci une liste de tous les segments (bords des cubes) qui pourraient potentiellement avoir une intersection avec le vaisseau spatial après que celui-ci se soit déplacé. Comme, à un instant donné, un vaisseau ne peut se déplacer de plus de 3 pixels, il suffit de noter les segments voisins du vaisseau (cf. ce que l'on a vu en cours). L'obtention de cette liste se fait via un appel à la fonction `get_cube_inter_segments`. Une fois la liste des segments récupérés (nous avons vu qu'il y en a 8 au maximum), il suffit pour chaque petit déplacement d'effectuer les tests d'intersection entre ces segments et le vaisseau spatial. C'est le rôle de la fonction `inter_cubes`.

Exercice 4 Écrivez une fonction :
`get_cube_inter_segments : float -> float -> ((float * float) * (float * float)) list`
qui, étant donné les coordonnées du centre d'un vaisseau spatial, renvoie la liste des segments des bords des obstacles potentiellement en intersection avec le vaisseau après un petit déplacement (moins de 3

pixels). Dans le fichier `aide_serveur`, une variable globale `cubes : Types_serveur.cube liste ref` est déclarée, qui stocke la liste de tous les obstacles du jeu.

Exercice 5 Écrivez une fonction :
`inter_cubes : float -> float -> ((float * float) * (float * float)) list -> bool`
qui, étant donné les coordonnées du centre d'un vaisseau spatial, et la liste des segments qui lui sont voisins, renvoie un booléen indiquant si le vaisseau spatial a une intersection avec au moins un des segments de la liste.

création/destruction

Exercice 6 Écrivez une fonction `cree_vaisseau : Graphics.color -> bool` qui essaye de rajouter dans le plateau de jeu un vaisseau ayant la couleur passée en argument. S'il existe déjà un autre vaisseau ayant cette couleur, la fonction ne crée pas le vaisseau et renvoie `false`, sinon le vaisseau est rajouté à la liste des vaisseaux du plateau de jeu et la fonction renvoie `true`. Dans ce cas, le vaisseau est placé aléatoirement dans l'espace de jeu. Vous pourrez supposer que, lorsque cette fonction est appelée, un mutex a été locké de manière à éviter d'obtenir des résultats exotiques et improbables. On rappelle que la liste de tous les vaisseaux actuellement dans le jeu est stockée dans la variable `vaisseaux : Types_serveur.vaisseau list ref`.

Exercice 7 Écrivez une fonction `detruit_vaisseau : Graphics.color -> unit` supprimant de la liste des vaisseaux (variable globale `vaisseaux : Types_serveur.vaisseau list ref`) celui dont la couleur est passée en argument. Là encore, vous pourrez supposer que, lorsque cette fonction est appelée, un mutex a été locké.