Introduction à la Programmation

—

Examen final sur machine

—

G1: monasse*(at)*imagine.enpc.fr G2: facciolg*(at)*imagine.enpc.fr
G3: alexandre.boulch*(at)*onera.fr G4: theophile.dalens*(at)*inria.fr
G5: bourkia(at)imagine.enpc.fr G6: fernandl*(at)*imagine.enpc.fr

—

20/01/16

—

1 Enoncé

1.1 Parcours de tunnel

On va programmer une variante du jeu Flappy Bird, sans effet de gravité. On dirige un vaisseau (représenté par un rectangle rouge) qui doit naviguer le plus loin possible dans un tunnel qui défile de la droite vers la gauche, le vaisseau restant toujours à mi-largeur de la fenêtre et pouvant se déplacer uniquement verticalement en réponse aux touches du clavier.

On stockera les coordonnées dans une grille de L cases en largeur et de H cases en hauteur (des constantes), mais on affichera tout en multipliant les coordonnées par un facteur `zoom` (une autre constante).

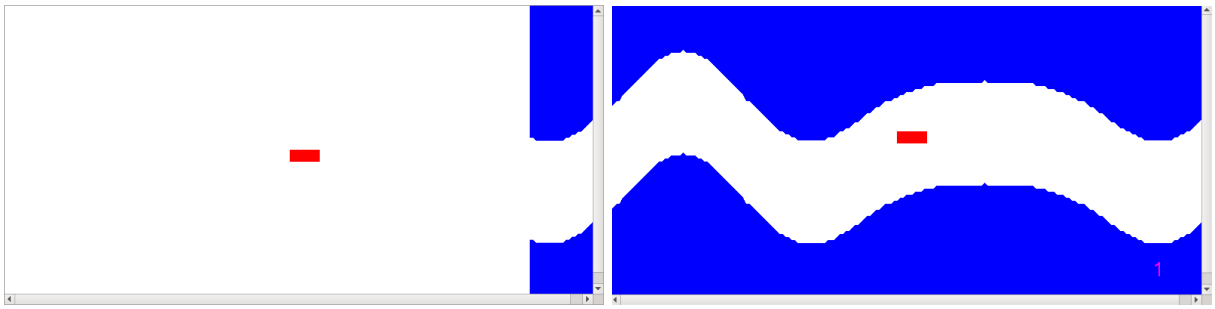

FIGURE 1 – À gauche, le tunnel avec les parois en bleu arrive de la droite vers le vaisseau en rouge. À droite, en cours de jeu, notez le score qui s'affiche.

1.2 Travail demandé

Il est plus important de livrer un code clair (commenté et indenté) et qui compile sans warning, même s'il ne répond pas à toutes les questions. Pour cela, vérifiez à chaque étape que votre programme compile et se lance correctement.

Créez un projet Imagine++ avec un fichier contenant le `main`. Prenez le `CMakeLists.txt` d'un projet existant ou celui de `[IMAGINEPP_ROOT]/test/Graphics/` et adaptez-le[1].

1. `[IMAGINEPP_ROOT]` est le chemin où Imagine++ est installé, typiquement `/usr/share/Imagine++` sous Mac et Linux, et `C:/Program Files (x86)/Imagine++` sous Windows

1.2.1 Tunnel

1. Dans des fichiers spécifiques, définissez la classe `Tunnel`, stockant une longueur n et des tableaux *dynamiques* `haut` et `bas` de valeurs entières (taille maximale des tableaux L).
2. Écrire un constructeur de `Tunnel`, celui-ci étant initialement de longueur 0.
3. Écrire le destructeur de `Tunnel`.
4. Ajouter une méthode `defile` prenant deux valeurs h et b. h et b s'ajoutent à la fin de `haut` et `bas`. Si ces tableaux atteignent la limite L, faire de la place en supprimant les plus vieilles valeurs (en indice 0, `haut[0]` et `bas[0]`).
5. Écrire une méthode `affiche` : le bord haut est déterminé par la courbe `haut` et le bord bas est sèparé du bas de la fenêtre par la courbe `bas`. Dans tout affichage, on multipliera les coordonnées logiques par une constante `zoom`. On pourra utiliser la fonction

   ```
   fillPoly(int x[], int y[], int n, Color col);
   ```

   où n est la taille commune des tableaux x et y et les sommets du polygone sont $(x[i], y[i])$ pour $0 \leq i < n$.
6. Écrire une méthode `vivant` qui indique si le rectangle de coins (x_0, y_0) et (x_1, y_1) (arguments de la méthode) est entièrement dans le passage du tunnel.

1.2.2 Vaisseau

7. Dans d'autres fichiers, définir une classe `Vaisseau` gardant en mémoire son centre (x, y).
8. Programmer le constructeur de `Vaisseau`.
9. Le vaisseau a des largeur et hauteur, qui sont des constantes. Écrire des méthodes retournant ces valeurs, ainsi que des accesseurs pour la position.
10. Écrire sa méthode `affiche`.
11. Écrire une méthode `deplace` qui indique la translation en y à appliquer au vaisseau.

1.2.3 Jeu

12. Faire défiler (dans une fonction appelée depuis le main) le tunnel, dont l'évolution est donnée par les fonctions suivantes :

$$h(x) = H/3 + cos(2x\pi/L) * H/8 + sin(3x\pi/L) * H/20 + \eta(x)$$
$$b(x) = H/3 - cos(2x\pi/L) * H/8 - sin(3x\pi/L) * H/20 + \nu(x)$$

    η et ν prenant des valeurs aléatoires entre $-H/30$ et $H/30$.
13. Gèrer le déplacement du vaisseau : il réagit aux flèches haut et bas. De plus, il a le temps de réagir deux fois au clavier avant chaque pas de défilement. On utilisera la fonction `Clavier` (cf par exemple le TP Tron) :

```
int Clavier() {
    Event e;
    do {
        getEvent(0,e);
        if (e.type==EVT_KEY_ON)
            return e.key;
    } while (e.type!=EVT_NONE);
    return 0;
}
```

14. Le score augmente de 1 chaque fois que le défilement s'est effectué L fois. Afficher le score dans un coin de la fenêtre.

Important : Quand vous avez terminé, créez une archive du projet *à votre nom ou au nom du binôme* en ZIP, RAR, TGZ ou 7z. N'incluez que les fichiers source et le CMakeLists.txt, pas les fichiers binaires créés par Cmake. Ne partez pas avant que le surveillant ne soit passé vous voir pour copier cette archive sur clé USB.