Introduction à la Programmation

G1: monasse*(at)*imagine.enpc.fr G2: nicolas.audebert*(at)*onera.fr
G3: alexandre.boulch*(at)*onera.fr G4: maxime.ferrera*(at)*onera.fr
G5: laurent.bulteau*(at)*u-pem.fr G6: pierre-alain.langlois*(at)*enpc.fr

— TP #4 —

1 Structures

Avertissement : Dans ce TP, nous allons faire évoluer des corps soumis à la gravitation, puis leur faire subir des chocs élastiques. Il s'agit d'un long TP qui nous occupera plusieurs séances. En fait, le TP suivant sera une réorganisation de celui-ci. Les sections 1.1.4 et 1.1.5 ne sont données que pour les élèves les plus à l'aise et ne seront abordées qu'en deuxième semaine. En section 1.2 sont décrites quelques-unes des fonctions à utiliser, et en 1.3 leur justification physique.

1.1 Etapes

1.1.1 Mouvement de translation

1. *Pour commencer, étudier le projet :*
   Télécharger le fichier `TP4.zip` sur la page habituelle, le décompresser et lancer votre environnement de développement. Parcourir le projet, en s'attardant sur les variables globales et la fonction main (inutile de regarder le contenu des fonctions déjà définies mais non utilisées). Le programme fait évoluer un point (`x`, `y`) selon un mouvement de translation constante (`vx`, `vy`), et affiche régulièrement un disque centré en ce point. Pour ce faire, afin de l'effacer, on retient la position du disque au dernier affichage (dans `ox` et `oy`) ; par ailleurs, deux instructions commençant par `noRefresh` sont placées autour des instructions graphiques afin d'accélérer l'affichage.
2. *Utiliser une structure :*
   Modifier le programme de façon à utiliser une structure Balle renfermant toute l'information sur le disque (position, vitesse, rayon, couleur).
3. *Fonctions d'affichage :*
   Créer (et utiliser) une fonction void AfficheBalle(Balle D) affichant le disque D, et une autre void EffaceBalle(Balle D) l'effaçant.
4. *Faire bouger proprement le disque :*
   Pour faire évoluer la position du disque, remplacer les instructions correspondantes déjà présentes dans main par un appel à une fonction qui modifie les coordonnées d'une Balle, en leur ajoutant la vitesse de la `Balle` multipliée par un certain pas de temps défini en variable globale ($dt = 1$ pour l'instant).

1.1.2 Gravitation

5. *Évolution par accélération :*
   Créer (et utiliser) une fonction qui modifie la vitesse d'une `Balle` de façon à lui faire subir une attraction constante $a_x = 0$ et $a_y = 0.0005$. Indice : procéder comme précédemment, c'est-à-dire ajouter $0.0005 * dt$ à v_y...
6. *Ajouter un soleil :*
   On souhaite ne plus avoir une gravité uniforme. Ajouter un champ décrivant la masse à la structure `Balle`. Créer un soleil (de type `Balle`), jaune, fixe (ie de vitesse nulle) au milieu de la fenêtre, de masse 10 et de rayon 4 pixels (la masse de la planète qui bouge étant de 1). L'afficher.
7. *Accélération gravitationnelle :*
   Créer (et utiliser à la place de la gravitation uniforme) une fonction qui prend en argument la planète et le soleil, et qui fait évoluer la position de la planète. Rappel de physique : l'accélération à prendre en compte est $-G\,m_S/r^3\,\overrightarrow{r}$, avec ici $G = 1$ (Vous aurez sans doute besoin de la fonction double sqrt(double x), qui retourne la racine carrée de `x`). Ne pas oublier le facteur dt... Faire tourner et observer. Essayez diverses initialisations de la planète (par exemple $x =$ `largeur`$/2$, $y =$ `hauteur`$/3$, $v_x = 1$, $v_y = 0$). Notez que

l'expression de l'accélération devient très grande lorsque r s'approche de 0 ; on prendra donc garde à ne pas utiliser ce terme lorsque r devient trop petit.

8. *Initialisation aléatoire :*
   Créer (et utiliser à la place des conditions initiales données pour le soleil) une fonction initialisant une `Balle`, sa position étant dans la fenêtre, sa vitesse nulle, son rayon entre 5 et 15, et sa masse valant le rayon divisé par 20. Vous aurez probablement besoin de la fonction Random...
9. *Des soleils par milliers...*
   Placer 10 soleils aléatoirement (et en tenir compte à l'affichage, dans le calcul du déplacement de l'astéroïde...).
10. *Diminuer le pas de temps de calcul :*
    Afin d'éviter les erreurs dues à la discrétisation du temps, diminuer le pas de temps dt, pour le fixer à 0.01 (voire à 0.001 si la machine est assez puissante). Régler la fréquence d'affichage en conséquence (inversement proportionnelle à dt). Lancer plusieurs fois le programme.

1.1.3 Chocs élastiques simples

11. *Faire rebondir l'astéroïde :*
    Faire subir des chocs élastiques à l'astéroïde à chaque fois qu'il s'approche trop d'un soleil, de façon à ce qu'il ne rentre plus dedans (fonction ChocSimple), et rétablir dt à une valeur plus élevée, par exemple 0.1 (modifier la fréquence d'affichage en conséquent). Pour savoir si deux corps sont sur le point d'entrer en collision, utiliser la fonction Collision.

1.1.4 Jeu de tir

(figure 1 droite)

12. *Ouvrir un nouveau projet :*
    Afin de partir dans deux voies différentes et travailler proprement, nous allons ajouter un nouveau projet Imagine++, appelé `Duel`, dans cette même solution. Recopier (par exemple par copier/coller) intégralement le contenu du fichier `Tp4.cpp` dans un fichier `Duel.cpp`. Une fois cette copie faite, modifier le fichier `CMakeLists.txt` en ajoutant deux lignes indiquant que l'exécutable `Duel` dépend de `Duel.cpp` et utilise la bibliothèque `Graphics` d'Imagine++.
13. *À vous de jouer !*
    Transformer le projet `Duel`, à l'aide des fonctions qui y sont déjà présentes, en un jeu de tir, à deux joueurs. Chacun des deux joueurs a une position fixée, et divers soleils sont placés aléatoirement dans l'écran. Chaque joueur, à tour de rôle, peut lancer une `Balle` avec la vitesse initiale de son choix, la balle subissant les effets de gravitation des divers soleils, et disparaissant au bout de 250 pas de temps d'affichage. Le gagnant est le premier qui réussit à atteindre l'autre... Conseils pratiques : positionner symétriquement les joueurs par rapport au centre, de préférence à mi-hauteur en laissant une marge d'un huitième de la largeur sur le côté ; utiliser la fonction GetMouse pour connaître la position de la souris ; en déduire la vitesse désirée par le joueur en retranchant à ces coordonnées celles du centre de la boule à lancer, et en multipliant par un facteur 0.00025.
14. *Améliorations :*
    Faire en sorte qu'il y ait systématiquement un gros soleil au centre de l'écran (de masse non nécessairement conséquente) afin d'empêcher les tirs directs.
15. *Initialisation correcte :*
    Modifier la fonction de placement des soleils de façon à ce que les soleils ne s'intersectent pas initialement, et qu'ils soient à une distance minimale de 100 pixels des emplacements des joueurs.

1.1.5 Chocs élastiques

(figure 1 gauche)

16. *Tout faire évoluer, tout faire rebondir :*
    On retourne dans le projet `Gravitation`. Tout faire bouger, y compris les soleils. Utiliser, pour les chocs élastiques, la fonction Chocs (qui fait rebondir les deux corps). Faire en sorte que lors de l'initialisation les soleils ne s'intersectent pas.

1.2 Aide

Fonctions fournies :

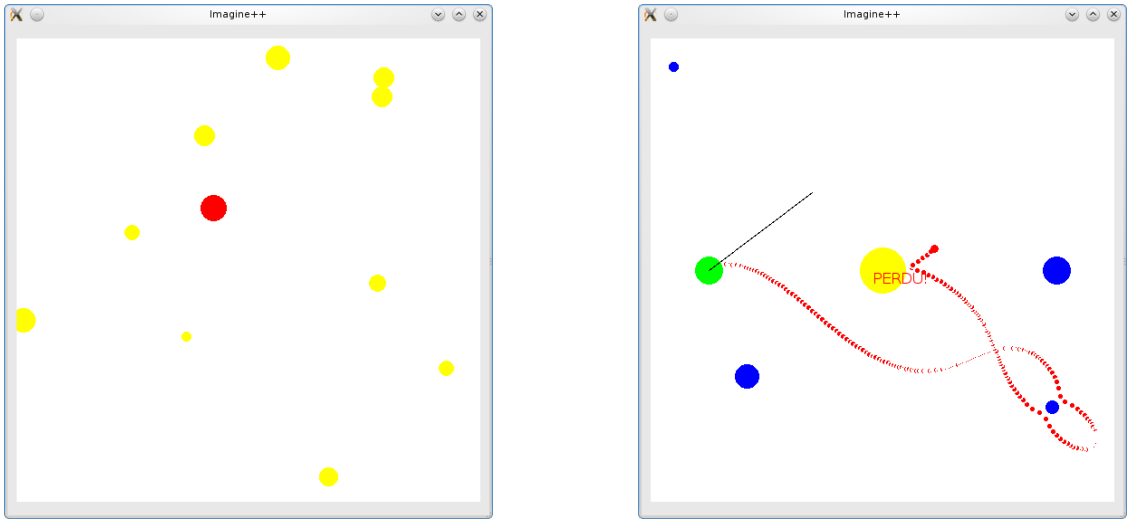

FIGURE 1 – Corps célestes et jeu de tir...

```
void InitRandom();
```

est à exécuter une fois avant le premier appel à Random.

```
double Random(double a, double b);
```

renvoie un double aléatoirement entre a et b (compris). Exécuter *une fois* InitRandom(); avant la première utilisation de cette fonction.

```
void ChocSimple(double x, double y, double &vx, double &vy, double m,
                double x2, double y2, double vx2, double vy2);
```

fait rebondir la première particule, de coordonnées (`x`, `y`), de vitesse (`vx`, `vy`) et de masse `m`, sur la deuxième, de coordonnées (`x2`, `y2`) et de vitesse (`vx2`, `vy2`), sans déplacer la deuxième.

```
void Choc(double x, double y, double &vx, double &vy, double m,
          double x2, double y2, double &vx2, double &vy2, double m2);
```

fait rebondir les deux particules l'une contre l'autre.

```
bool Collision(double x1, double y1, double vx1, double vy1, double r1,
               double x2, double y2, double vx2, double vy2, double r2);
```

renvoie `true` si le corps de coordonnées (`x1`, `y1`), de vitesse (`vx1`, `vy1`) et de rayon `r1` est sur le point d'entrer en collision avec le corps de coordonnées (`x2`, `y2`), de vitesse (`vx2`, `vy2`) et de rayon `r2`, et `false` sinon.

1.3 Théorie physique

NB : Cette section n'est donnée que pour expliquer le contenu des fonctions pré-programmées fournies avec l'énoncé. Elle peut être ignorée en première lecture.

1.3.1 Accélération

La somme des forces exercées sur un corps A est égale au produit de sa masse par l'accélération de son centre de gravité.

$$\sum_i \overrightarrow{F}_{i/A} = m_A \overrightarrow{a}_{G(A)}$$

1.3.2 Gravitation universelle

Soient deux corps A et B. Alors A subit une force d'attraction

$$\overrightarrow{F}_{B/A} = -Gm_A m_B \frac{1}{d_{A,B}^2} \overrightarrow{u}_{B\to A}.$$

1.3.3 Chocs élastiques

Soient A et B deux particules rentrant en collision. Connaissant tous les paramètres avant le choc, comment déterminer leur valeur après ? En fait, seule la vitesse des particules reste à calculer, puisque dans l'instant du choc, les positions ne changent pas.

Durant un choc dit *élastique*, trois quantités sont conservées :

1. la quantité de mouvement $\overrightarrow{P} = m_A \overrightarrow{v}_A + m_B \overrightarrow{v}_B$
2. le moment cinétique $M = m_A \overrightarrow{r}_A \times \overrightarrow{v}_A + m_B \overrightarrow{r}_B \times \overrightarrow{v}_B$ (qui est un réel dans le cas d'un mouvement plan).
3. l'énergie cinétique $E_c = \frac{1}{2} m_A v_A^2 + \frac{1}{2} m_B v_B^2$.

Ce qui fait 4 équations pour 4 inconnues.

1.3.4 Résolution du choc

On se place dans le référentiel du centre de masse. On a alors, à tout instant :

1. $\overrightarrow{P} = 0$ (par définition de ce référentiel), d'où $m_A \overrightarrow{v}_A = -m_B \overrightarrow{v}_B$.
2. $M = (\overrightarrow{r}_A - \overrightarrow{r}_B) \times m_A \overrightarrow{v}_A$, d'où, en notant $\Delta\overrightarrow{r} = \overrightarrow{r}_A - \overrightarrow{r}_B$, $M = \Delta\overrightarrow{r} \times m_A \overrightarrow{v}_A$.
3. $2E_c = m_A(1 + \frac{m_A}{m_B}) v_A^2$.

La constance de E_c nous informe que dans ce repère, la norme des vitesses est conservée, et la constance du moment cinétique que les vitesses varient parallèlement à $\Delta\overrightarrow{r}$. Si l'on veut que les vitesses varient effectivement, il ne nous reste plus qu'une possibilité : mutliplier par -1 la composante des $\overrightarrow{v}_i$ selon $\Delta\overrightarrow{r}$. Ce qui fournit un algorithme simple de rebond.

1.3.5 Décider de l'imminence d'un choc

On ne peut pas se contenter, lors de l'évolution pas à pas des coordonnées des disques, de décider qu'un choc aura lieu entre t et $t + dt$ rien qu'en estimant la distance entre les deux disques candidats à la collision à l'instant t, ni même en prenant en plus en considération cette distance à l'instant $t + dt$, car, si la vitesse est trop élevée, un disque peut déjà avoir traversé l'autre et en être ressorti en $t + dt$... La solution consiste à expliciter le minimum de la distance entre les disques en fonction du temps, variant entre t et $t + dt$.

Soit $N(u) = (\overrightarrow{r}_A(u) - \overrightarrow{r}_B(u))^2$ le carré de la distance en question. On a :

$$N(u) = (\overrightarrow{r}_A(t) - \overrightarrow{r}_B(t) + (u - t)(\overrightarrow{v}_A(t) - \overrightarrow{v}_B(t)))^2$$

Ce qui donne, avec des notations supplémentaires :

$$N(u) = \Delta\overrightarrow{r}(t)^2 + 2(u - t)\Delta\overrightarrow{r}(t) \cdot \Delta\overrightarrow{v}(t) + (u - t)^2 \Delta\overrightarrow{v}(t)^2$$

La norme, toujours positive, est minimale au point u tel que $\partial_u N(u) = 0$, soit :

$$(t_m - t) = -\frac{\Delta\overrightarrow{r}(t) \cdot \Delta\overrightarrow{v}(t)}{\Delta\overrightarrow{v}(t)^2}$$

Donc :

1. si $t_m < t$, le minimum est atteint en t,
2. si $t < t_m < t + dt$, le minimum est atteint en t_m ;
3. sinon, $t + dt < t_m$, le minimum est atteint en $t + dt$.

Ce qui nous donne explicitement et simplement la plus petite distance atteinte entre les deux corps entre t et $t + dt$.