BEGUE Adrien
BROSSARD Florian
L1 I1 A1

Sujet N°4

Mode d'emploi

Dans les différents menus, il est proposé des choix à faire. Si l'on entre un choix qui n'est pas proposé, le programme s'arrête. La simulation quant à elle ne s'arrête que sur fermeture de la fenêtre Billard.

Menu de paramétrage de la simulation :

- Choix 1 : Une bille seule avec une direction aléatoire.

- Choix 2 : n bille placé sur la ligne verticale centrale du billard avec une direction aléatoire chacune et gestion des chocs bille/bille. n est un entier saisis par l'utilisateur et compris entre 1 et 16 inclus. Une valeur entrée qui n'est pas dans cet intervalle entraînera l'extinction du programme.

- Choix 3 : n billes placées sur la ligne verticale centrale du billard avec une direction aléatoire chacune sans gestion des chocs bille/bille. n est un entier saisis par l'utilisateur et compris entre 1 et 16 inclus. Une valeur entrée qui n'est pas dans cet intervalle entraînera l'extinction du programme.

- Choix 4 : Une bille avec une direction aléatoire vers les 15 autres billes placés sous forme triangulaire. Les chocs bille/bille sont gérés.

- Choix 5 : Une bille allant tout droit vers les 15 autres billes placées sous forme triangulaire. Les chocs bille/bille sont gérés.

Une fois un de ces 5 choix effectué, le programme montreras la/les bille(s) dans leurs positions initiales et proposeras un second menu avec un choix unique :

- Choix 1 : Lancement de la simulation.

Dans ce menu également, tout autre choix entraînera l'extinction du programme.

Résolutions des problèmes

Les différents problèmes posés étaient :

1. La représentation des différents éléments de la simulation
2. Le paramétrage de la simulation
3. Le calcul de la position et du déplacement d'une bille
4. Le placement initial des billes
5. Calcul d'une direction aléatoire pour une bille
6. La gestion des chocs bille/côté
7. La gestion des chocs bille/bille
8. L'affichage graphique du billard

1. La représentation des différents éléments de la simulation

Afin de simuler les déplacements des billes sur une table de billard, on a dû poser la manière de représenter les différents éléments :

- Les billes
  Elles sont représentées par un cercle de rayon R, un numéro, une couleur ayant les composantes r, v, b (rouge, vert, bleue), un centre de coordonnées (cx,cy), une vitesse v (relative), une direction. La direction d'une bille est représentée par un vecteur unitaire d ayant comme point d'application le centre de la bille et comme coordonnées (dx,dy). dx et dy sont des points appartenant au périmètre d'un cercle trigonométrique, la norme de ce vecteur est donc toujours la même quelle que soit la bille.

- La table
  La table de billard est représentée par un rectangle de couleur verte est de dimension lng (longueur) et lrg (largeur) définies manuellement dans le code.

- Les évènements
  Les évènements sont représentés par leurs type (représenté par un entier allant de 0 à 5 chacun représentant un type de choc différents), le temps t pour lequel l'évènement à lieu, le numéro des billes impliquées dans l'évènement.

- Les paramètres
  Les paramètres sont représentés par un booléen indiquant si l'on prend en compte les chocs bille/bille (dans le cas contraire elles se traversent), un booléen indiquant si on lance la simulation (ce booléen est initialisé à faux et devient vrai s'il n'y a pas eu d'erreur pendant le paramétrage), un booléen représentant le fait de placer les billes en position classique (une bille vers les 15 autres en triangle) ou non, le nombre de bille présentes sur la table.

2. Le paramétrage de la simulation

Le problème du paramétrage consistait à pouvoir choisir le 'type' de la simulation et le placement initial des billes sur la table.
Nous avons résolus ce problème en développant un menu modifiant une variable de type table passée en paramètre.
Ce menu est représenté par un texte présentant les différents choix possibles et par l'attente de la saisie du numéro du choix de l'utilisateur.
Nous avons choisis de programmer une fonction 'dans le cas de' exécutant les fonctions adéquates en fonction de la saisie.

3. Le calcul de la position et du déplacement d'une bille

On a le système d'équation suivant :

$$E\begin{cases} cx = px + t * dx * v \\ cy = py + t * dy * v \end{cases}$$

(cx,cy) représente les coordonnées du centre de la bille, t le temps actuel de la simulation, (dx,dy) le vecteur unitaire de direction de la bille, v la vitesse de la bille. (px,py) représente les coordonnées du centre de la bille l'instant du dernier évènement.
On peut donc calculer facilement la position du centre 'actuelle' de chaque bille.
On peut déduire, d'après ces équations, que l'on incrémente la position actuelle de $t * \begin{Bmatrix} dx \\ dy \end{Bmatrix} * v$
Et donc que l'on n'a pas besoin de stocker les variables px et py.
On obtient donc

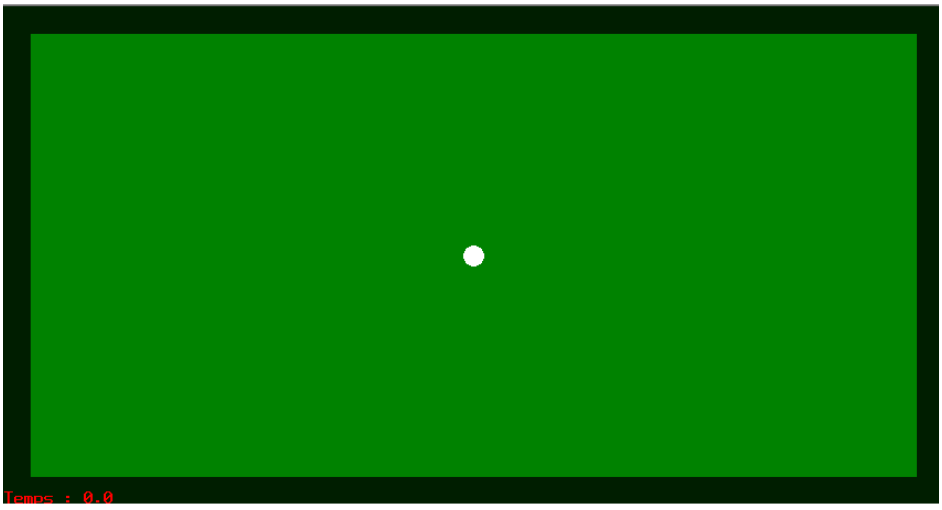
$$E\begin{cases} cx = cx + t * dx * v \\ cy = cy + t * dy * v \end{cases}$$

Nous avons donc programmé une fonction qui, pour chaque bille, calcul l'évènement dans lequel la bille est impliquée puis l'on prend celui dont la date est la plus proche et on avance toute les bille jusqu'à cette date : t est la différence t2-t1 ou t1 est la date actuel et t2 la date de l'évènement le plus proche puis on recalcule le vecteur directionnel de la/des bille(s) impliquée(s).

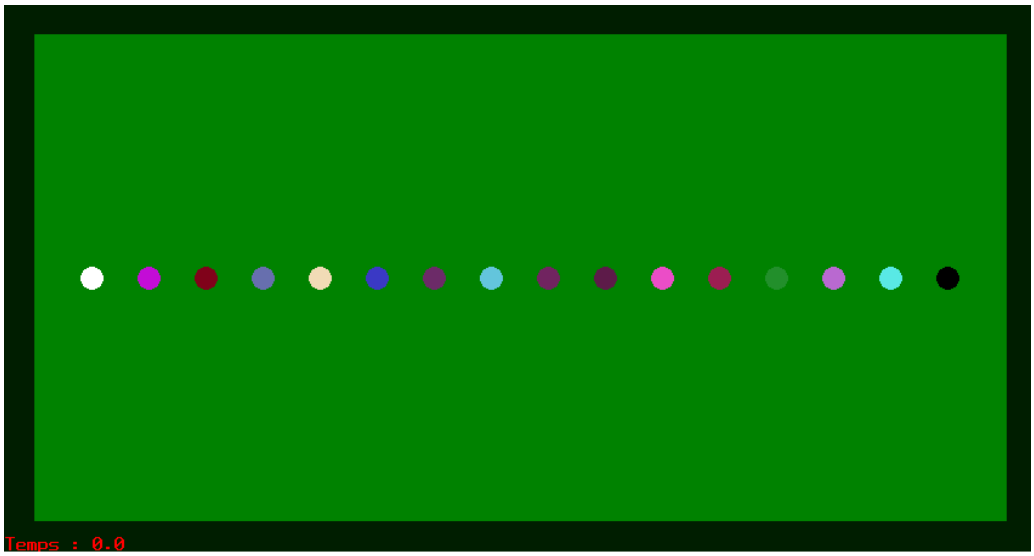
4. Le placement initial des billes

On donne le choix à l'utilisateur de placer les billes selon trois configurations :

1. On place 1 seule bille sur la table, cette bille est blanche et son centre se situe au centre du rectangle représentant la partie jouable de la table. On a donc
   cx = 25 + (lng / 2) et cy = 25 + (lrg / 2) avec lng et lrg respectivement la longueur et largeur de la table. La valeur 25 est la taille de la bande noire (partie non jouable de la table).

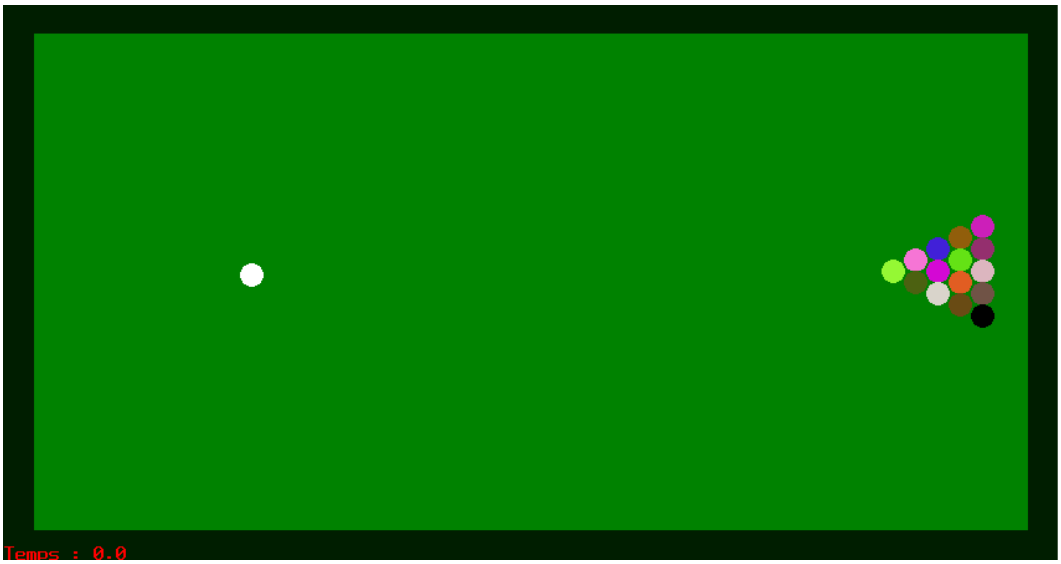


2. On place n bille sur la table et on les aligne toute sur la ligne centrale en largeur de la table. L'espace entre deux bille est le même pour chaque bille. On obtient donc :
   Pour i de 1 à n faire
   t[i].cx = ((lng / (n + 1)) * (i + 1)) + 25.0
   t[i].cy = (lrg / 2) + 25.0
   Fait
   Avec t le tableau de bille et n le nombre de bille à placer.



3. On place 16 billes sur la table. Une est placé sur la ligne centrale avec
   cx = 200 et cy = (lrg / 2) + 25.0
   R représente le rayon d'une bille et 200 est un nombre choisis arbitrairement.

   On place les 15 autres sous forme de triangle (ou de pyramide) : pour chaque colonne on incrémente le nombre de billes présentes sur cette colonne de 1 afin d'obtenir cette forme



   On doit donc imaginer un curseur qui se déplace comme suit :
   En x, la première colonne se place en x = lng + 25.0 – (14 * R).
   14 * R représente le fait que la première bille du triangle (celle du sommet) est à 7 fois le diamètre d'une bille, éloignée du côté droit de la table : il y a donc un espace d'un diamètre de bille entre la base du triangle et le côté droit de la table. A chaque colonne suivant la première, on 'avance' le curseur d'un diamètre de bille soit 2*R. Le curseur avance à droite.

En y, le curseur se place sur la ligne centrale de la table puis, pour chaque colonne suivante on augmente cette composante de R (le curseur se déplace vers le bas de l'écran) on place le nombre de bille adéquate (égal au numéro de la colonne) chacune espacé de 2*R par rapport à leurs centre.
On doit également prendre en compte les erreurs de précisions dû aux calculs on espace donc les bille (on avance le curseur) de (2 + $\frac{1}{1000}$) * R en x et en y.

5. Calcul d'une direction aléatoire pour une bille

Comme précisé avant, la direction d'une bille est représentée par un vecteur unitaire $d = \begin{Bmatrix} dx \\ dy \end{Bmatrix}$
Où les composantes dx et dy appartiennent au cercle trigonométrique (de rayon 1).
On obtient donc ces formules :
d = random() * 2.0 * π
dx = sin (d)
dy = cos (d)
d est une valeur aléatoire comprise entre 0 et $2\ \pi$

6. Gestion des chocs bille/côté

La gestion d'un choc entre une bille est un côté est faite de manière suivante :

Si la coordonnée en x de la bille plus son rayon est égale à la coordonnées en x du côté droit, il y a choc, on avance la bille jusqu'au moment t où cx = x et on inverse dx ce qui auras pour effet d'inverser le mouvement en x de la bille. L'évènement est de type 1.

Si la coordonnée en x de la bille moins son rayon est égale à la coordonnées en x du côté gauche, il y a choc, on avance la bille jusqu'au moment t où cx = x et on inverse dx ce qui auras pour effet d'inverser le mouvement en x de la bille. L'évènement est de type 2.

Si la coordonnée en y de la bille moins son rayon est égale à la coordonnées en y du côté supérieur, il y a choc, on avance la bille jusqu'au moment t où cy = y et on inverse dy ce qui auras pour effet d'inverser le mouvement en y de la bille. L'évènement est de type 3.

Si la coordonnée en y de la bille plus son rayon est égale à la coordonnées en y du côté inférieure, il y a choc, on avance la bille jusqu'au moment t où cy = y et on inverse dy ce qui auras pour effet d'inverser le mouvement en y de la bille. L'évènement est de type 4.

En revanche si aucun évènement 'de choc' ne survient avant un laps de temps T, alors on avance toutes les billes jusqu'au temps t + T et on recalcule les évènements. L'évènement est de type 0.

Nous avons développés une fonction gestionChoc qui appel pour chaque bille la fonction prochainEvenement. La fonction prochainEvenement retourne une variable de type evenement qui calcul l'évènement pour la bille passée en paramètre le plus proche. La fonction gestionChoc elle calcul parmi tous ces évènement celui qui seras le plus proche et avance les bille jusqu'à ce dernier puis, par le biais d'une fonction 'dans le cas de' appelée avec le type de l'évènement change la direction de la bille impliquée.

7. Gestion des chocs bille/bille

Dans la cadre d'un choc bille/bille (type 5), on doit calculer le moment t exact où la bille b1 entre en collision avec la bille b2. Nous avons donc développé une fonction dateChoc qui prend deux variables de type bille en paramètre et qui est appelée dans la fonction prochainEvenement sur tous les couples de billes possible (à l'exception d'une bille avec elle-même).

On sui ce résonnement :

On veut le temps t pour lequel la distance entre les deux billes b1 et b2 est égale à 2.0 * R
R est le rayon de ces billes.
Pour aider à la compréhension, les différentes appellations "utiles" sont mises en commentaires dans la fonction dateChoc dans le code source

On appel respectivement :
(cx1,cy1) ; (cx2,cy2) les coordonnées des billes b1 et b2 à t
(dx1,dy1) ; (dx2,dy2) les directions des billes b1 et b2
(px1,py1) ; (px2,py2) les coordonnées des billes b1 et b2à t=0
v1 et v2 les vitesses de b1 et b2
On a:
Les systèmes d'équations suivants :

$$E1\begin{cases} cx1 = px1 + t * dx1 * v1 \\ cy1 = py1 + t * dy1 * v1 \end{cases}$$

$$E2\begin{cases} cx2 = px2 + t * dx2 * v2 \\ cy2 = py2 + t * dy2 * v2 \end{cases}$$

$A = (cx1 - cx2)$
$B = (cy1 - cy2)$
La distance d entre les billes à t :
$\sqrt{A^2 + B^2} = 2 * R$
Posons :
$C = px1 - px2$
$D = py1 - py2$
$E = dx1 * v1 - dx2 * v2$
$F = dy1 * v1 - dy2 * v2$
On trouve alors :
$A = C + t * E$
$B = D + t * F$
Or on a
$\sqrt{A^2 + B^2} = 2 * R \iff A^2 + B^2 = 4 * R^2$
Et
$A^2 = C^2 + 2 * C * t * E + t^2 * E^2$
$B^2 = D^2 + 2 * D * t * F + t^2 * F^2$
D'où
$A^2 + B^2 = C^2 + D^2 + 2t(C * E + D * F) + t^2 * (E^2 + F^2)$
Donc on cherche
$C^2 + D^2 + 2t(C * E + D * F) + t^2 * (E^2 + F^2) = 4 * R^2$
$\iff C^2 + D^2 + 2t(C * E + D * F) + t^2 * (E^2 + F^2) - 4 * R^2 = 0$
Posons :
$H = C^2 + D^2$
$I = (C * E + D * F)$
$J = E^2 + F^2$

On obtient l'équation du second degré suivante :
$H - 4 * R^2 + 2t * I + t^2 * J = 0$
On sait que
$\Delta = b^2 - 4 * a * c$
$a = J$
$b = 2 * I$
$c = H - 4 * R^2$
On considère uniquement les solutions réel ($\Delta \geq 0.0$)
Si Δ ≥ 0.0 ET Δ ≤ EPSILON alors
$t = -b/(2 * a)$
Sinon si Δ > EPSILON alors
$t1 = \left(-b - \sqrt{\Delta}\right)/(2 * a)$
$t2 = (-b + \sqrt{\Delta})/(2 * a)$
Si (t1 < t2) alors
t = t1
Sinon
t = t2
Fin si
Fin si
On suppose qu'une précision EPSILON de 10^{-6}s est suffisante pour ne pas entrainer de bugs avant un certain temps d'exécution de la simulation.

Nous avons également implanté un test juste après le calcul de a (J) afin de vérifier si ce dernier est strictement supérieur à 0.0, dans le cas contraire on pourrait se retrouver face à une division par 0 (car a = J). Cela signifie également qu'il n'y aura pas choc.

8. L'affichage graphique du billard

Nous avons développés 3 fonctions distinctes destinées à afficher la simulation sous forme graphique.

La première, demarrageAffichage, initialise la fenêtre de dessin afin d'avoir la taille de la table définie dans le code et un rectangle vert foncé sur un fond noir qui est plus grand de 25 pixels sur chaque côté. Cette fonction affiche aussi en bas à gauche de la fenêtre une chaîne de caractère indiquant le temps actuel de la simulation.

La seconde, tamponBillard, sert à mettre en tampon la chaîne de caractère indiquant le temps ainsi que le rectangle vert foncé.

La dernière, affichage, appel la fonction tamponBillard puis dessine toutes les billes avec leurs positions actuelles.

Nous avons également développés la fonction gestionCouleur, qui permet d'avoir la bille N°1 blanche, la N°2 noire et toutes les autres sont définies avec des composantes aléatoires.

Répartition des tâches

Adrien BEGUE :

- Définition des constantes
- Implantations des menus textuels
- Réflexion sur le placement des billes en triangle et en ligne
- Implantation des calculs donnés (fonction calculChocEntreBille)

Florian BROSSARD :

- Résolution du problème des chocs bille/bille
- Implantation de l'affichage graphique
- Résolution du problème des chocs bille/bille
- Implantations des placements des billes en triangle et en ligne
- Implantations de la direction aléatoire des billes
- Résolution et Implantation des chocs bille/côté
- Implantations des couleurs des billes
- Implantations des déplacements des billes
- Finalisation du paramétrage de la simulation (type agrégé paramètre)