SIMULATEUR DE BILLARD RÉALISTE

par

Julien Ploquin

Mémoire présenté au Département d'informatique
en vue de l'obtention du grade de maître ès sciences (M.Sc.)

FACULTÉ DES SCIENCES

UNIVERSITÉ DE SHERBROOKE

Sherbrooke, Québec, Canada, 11 octobre 2012

Library and Archives Canada

Published Heritage Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et Archives Canada

Direction du Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 978-0-494-91690-2

Our file Notre référence
ISBN: 978-0-494-91690-2

Le 16 octobre 2012

le jury a accepté le mémoire de Monsieur Julien Ploquin
dans sa version finale.

Membres du jury

Professeur Jean-Pierre Dussault
Directeur de recherche
Département d'informatique

Professeure Virginie Charette
Membre
Département de mathématiques

Professeur Richard Egli
Président rapporteur
Département d'informatique

Sommaire

Aujourd'hui, il n'existe aucun simulateur de jeu de billard qui puisse bénéficier d'une validation scientifique. Les approximations physiques et les limitations dans la liberté des joueurs sont les principales responsables des problèmes de réalisme qu'il est possible d'observer dans les simulateurs existants. L'objectif de cette maîtrise est donc de créer un simulateur qui soit le plus réaliste possible, justification à l'appui. Pour cela, des formules analytiques de physique mécanique combinées à une simulation par évènements seront utilisées pour mettre à jour les positions et vélocités des billes. La vitesse de calcul et de résolution de coup ne devant pas devenir trop encombrante, des techniques pour accélérer le traitement seront utilisées. Parmi celles-ci, notons un résolveur d'équation quartique s'appuyant sur une méthode non-itérative. Des outils d'optimisation de problèmes de complémentarité linéaire seront également utilisés pour pallier à certains cas problématiques introduits par la liberté d'action dans l'axe z. Le simulateur répond finalement à la majorité des coups réels, sauf certains utilisant des aspects physiques non-traités tels que la déformation des bandes qui devront être traités dans des travaux futurs. Une analyse qualitative de certains coups filmés viendra confirmer la validité des modèles utilisés pour la simulation. Il est aussi montré que la sensibilité d'un joueur est grandement affectée par la liberté et la fidèlité physique proposée grâce à une comparaison avec un simulateur reconnu se limitant à un modèle physique comprenant de nombreuses approximations.

Mots-clés: simulation ; billard ; mécanique ; physique ; résolveur ; optimisation ; hybridation.

Sommaire

Remerciements

Tout d'abord je remercie M. Jean-Pierre Dussault, mon directeur de recherche, pour son intérêt dans la réussite du projet, sa disponibilité et son aide précieuse. Je remercie également Jean-François Landry, collègue du laboratoire d'optimisation travaillant sur un joueur de billard, pour me l'avoir fait partager pour tester le simulateur et avoir toujours été présent lorsque j'avais besoin de ses connaissances. Je dis également merci à tous mes collègues du laboratoire d'optimisation pour leur aide et leur bonne humeur : Vincent Ducharme, Maxime Toussaint, Bilel Kchouk, Martin Guay, Emilie Joannopoulos, Catherine Simard et Nicolas Bureau. Je remercie aussi l'ESEO (Ecole Supérieure d'Electronique de l'Ouest) de m'avoir permis de venir faire une maîtrise à l'Université de Sherbrooke.

Je remercie mes parents pour leur soutien et leur confiance sans quoi rien de tout cela n'aurait été possible.

Remerciements

Table des matières

Liste des figures

Introduction

Le billard tel qu'on le connaît aujourd'hui est le fruit d'une longue évolution du jeu de croquet datant du moyen-âge. Le croquet étant un jeu d'extérieur, les nobles étaient agacés de devoir se plier au temps pour pratiquer leur activité. Ainsi les premières apparitions de croquet en intérieur ont lieu dans les royaumes d'Europe de l'ouest. C'est à l'époque de Louis XI que ce sport commence sa première vraie transformation vers le billard tel qu'on le connaît aujourd'hui. Le roi souffrant de problème de dos, il ne pouvait pas se pencher pour jouer. Il demanda donc le «réhaussement» de la zone de jeu, ce qui plus simplement correspond à disposer un jeu de croquet sur une table. Il faut ensuite attendre la fin du 17^e siècle pour que l'évolution continue : les tables sont munies de six poches, d'un arceau et de deux billes. Les règles variaient énormément mais gardaient une constante : il fallait rentrer les billes dans les poches. L'activité est alors officiellement réservée à la noblesse, elle est cependant beaucoup pratiquée et observée dans des « académies » (en particulier à Paris) où joueurs maîtres et amateurs se côtoyaient. Le 18^e siècle voit disparaître l'arceau et le tapis de jeu fait son entrée. Les bandes en bois se voient elles aussi recouvertes par le tapis pour améliorer les rebonds. Le bâton de bois courbé du croquet est abandonné pour les queues de billard que l'on connaît aujourd'hui. Les billes se font aussi plus nombreuses sur la table, rallongeant la durée des parties. C'est aussi à cette époque que deux styles de jeu se distinguent : le carambolage et l'empochage. Le premier se concentre sur les collisions pouvant se produire entre les billes et avec les bandes, le but est de réussir une série de collisions. Les poches sont retirées pour ce style de jeu. L'empochage est, quand à lui, basé sur le fait de rentrer les billes dans le poches. Le premier style de jeu sera plus ancré en France tandis que le second sera plus populaire en Angleterre. Le 19^e siècle va voir les dernières évolutions en donnant un bout en

cuir au bout de la queue et en lui ajoutant de la craie bleue qui permet d'effectuer des coups exploitant bien plus les effets qu'auparavant. A noter aussi le remplacement du bois des bandes par du caoutchouc. La maîtrise de plus en plus impressionnante des joueurs fait naître de nouveaux styles de jeu : le snooker pour l'empochage et le 3 bandes pour le carambolage. Malgré les démonstrations techniques impressionnantes au carambolage, c'est le style de l'empochage qui sera le plus démocratisé, étant plus accessible pour les joueurs amateurs [34][16][28]. L'étude du comportement physique des billes dans le jeu du billard date elle aussi de plusieurs siècles. Coriolis [10] fut un des premiers à s'y lancer, étant passionné par le jeu. Son analyse est très poussée et est régulièrement citée par les études plus récentes comme un travail assez exceptionnel vis-à-vis des moyens de l'époque. Les connaissances en mathématiques et physiques mécanique se sont depuis bien améliorée et certains modèle de Coriolis ont été retravaillés. Les études les plus remarquables dans le domaine sont maintenant signées de la main de Marlow [22], Shepard [29] ou encore Alciatore [1]. Ces ouvrages couvrent la majorité des interrogations que l'on peut se poser sur les lois physiques impliquées dans le jeu de billard. On peut aussi citer Petit [26] qui s'adresse aux joueurs en particulier mais approfondit l'aspect scientifique des astuces et conseils qu'il prodigue. Avec tous ces travaux sur le billard, on est en droit de croire qu'aujourd'hui le monde de la simulation de jeu de billard devrait être fidèle à la réalité. Ceci n'est pourtant pas encore le cas. Il n'existe aujourd'hui aucun programme de jeu ou de simulation dont la validité aurait été prouvée scientifiquement. En effet la majorité des jeux sont des logiciels propriétaires, ce qui implique qu'il est impossible de « regarder à l'intérieur » afin de vérifier la validité des opérations effectuées, l'ampleur des approximations ou encore les possibles « bricolages » pour obtenir un rendu qui paraîsse réaliste. Un bémol tout de même, les jeux sont créés pour le plaisir et n'ont généralement pas la prétention d'être des références scientifiques. Du côté des simulateurs, ayant eux pour objectif de se rapprocher au maximum de la réalité, il n'y a pas non plus de réelle référence. Certains sont plus connus que d'autres, comme par exemple *Billards* [25], *FastFiz* [19] ou *FooBillard* [8], mais ils ont tous un point commun : soit ils ne sont pas approuvés scientifiquement (il faut comprendre ici que la démarche d'explication des règles physiques n'est pas faite ou pas assez testée) soit pas assez réalistes pour reproduire une majorité de coups fidèles à la physique du billard. Pour *Billiards*,

qui utilise le moteur physique ODE [31], des tests ont été effectués montrant que certains coups pouvait déboucher sur des bugs assez dérangeants (le souvenir d'une bille décollant de la table après une collision de bande pour ne plus réatterrir est gênant pour finir une partie). D'un autre côté, *FastFiz* développe son propre moteur physique mais le simplifie trop, ce qui fait que les effets ne sont pas pris en compte lors des collisions, gros défaut pour qui connaît l'importance de ce paramètre dans une partie entre joueurs professionnels. L'absence de véritable simulateur fidèle est la première motivation à la création d'un nouveau qui comblerait les défauts des autres. Parallèlement à ces simulateurs plus ou moins défaillants, une autre raison a motivé ce projet. Depuis quelques années, des robots dont la fonction est d'exécuter des coups au billard sont développés un peu partout dans le monde, comme DeepGreen [20][15], le bras développé pour les tests de [23], celui développé par les Allemands de l'Université technique de Munich [24] ou encore RobotShark [2]. Ces installations ont pour but de devenir des adversaires de taille pour des joueurs humains professionels. Pour bien fonctionner, elles ont donc besoin d'appréhender la façon dont les billes vont se comporter sur la table avant de pouvoir frapper l'une d'entre elles, soit un simulateur réaliste, et d'une intelligence artificielle créant un arbre de décision elle aussi reliée au simulateur. Des joueurs sont en développement dans de nombreuses écoles, comme par exemple *PoolMaster* développé à l'Université de Sherbrooke [18] ou PickPocket [30]. Il n'y a pas cependant de travaux particuliers pour améliorer la simulation, qui est pourtant la pièce centrale pour l'efficacité d'un joueur robotique. Ceci constitue une raison de plus de se lancer dans la création d'un simulateur le plus réaliste possible.

Dans un premier temps, la partie théorique du simulateur sera exposée, regroupant les règles physiques utilisées mais aussi le type de simulation choisi parmi ceux existants. L'algorithme général du simulateur clôturera cette partie. Dans un deuxième temps, les outils développés et les moyens mis en œuvre pour la résolution de nouveaux problèmes seront abordés. Enfin, les résultats obtenus seront analysés dans une dernière partie.

Introduction

Chapitre 1

Simulation physique réaliste d'un jeu de billard

Dans ce chapitre, nous passerons en revue toute la théorie nécessaire à une simulation la plus fidèle et rapide possible pour un jeu de billard. Tout d'abord les connaissances en physique mécanique seront abordées puis les différents types de simulation à disposition et le choix effectué entre ceux-ci. Enfin, l'algorithme général du simulateur sera détaillé.

1.1 Physique impliquée

Le défaut de beaucoup de simulateurs existants est leur fâcheuse tendance à simplifier à outrance (voire ignorer) certains aspects de la physique mécanique qui sont pourtant des éléments clés du jeu de billard. Il est fait référence aux frottements entre les billes lors des collisions, à la prise en compte du spin de la bille dans son déplacement et ses chocs contre un autre objet ou encore les déplacements selon l'axe z (le fait qu'une bille puisse décoller du tapis). Le simulateur développé tiendra compte de ces propriétés physiques, car celles-ci permettent une plus grande liberté d'action à un joueur. En effet, le panel de coups disponibles en enlevant les effets de la vélocité angulaire (définition de celle-ci prise dans [22][29][1][26]) en z des billes ou leur faculté de décollage (les coups appelés « massé » en sont le parfait exemple) se voit grande-

ment diminuer. Si l'on faisait jouer une IA contrôlant un robot contre une joueur humain professionnel, l'abstraction de ces libertés serait très contraignante.

Une connaissance du comportement des billes dans l'espace de jeu est donc primordiale pour obtenir un simulateur répondant aux objectifs fixés. Beaucoup d'études existent sur la physique mécanique à l'œuvre dans le jeu de billard [22][29][1][26]. Plusieurs seront utilisées dans les développements suivants. Il y aura bien entendu certaines simplifications de la physique mécanique réelle pour certaines situations. Si toutes les règles de la physique mécanique devaient être prises en compte à la lettre, ce simulateur serait très lourd et sûrement toujours en développement. Si l'on veut permettre à une IA de jouer sur le simulateur, il faut qu'elle puisse tester un panel de coups confortable pour prendre une décision, le simulateur se doit donc d'être léger tout en étant précis. Ainsi les simplifications sont indispensables, le tout étant de ne pas trop en faire.

Il sera décrit dans une première partie les déplacements des billes sur la table, leur comportement et leur trajectoire théorique résultante. Les collisions seront ensuite étudiées en deuxième partie, à commencer par le choc queue-bille, puis les collisions bille-bille et bille-bande. Enfin, les points non-traités dans le simulateur seront abordés.

1.1.1 Déplacement des billes

Types de déplacement

Les billes de billard évoluent sur un tapis. Ce dernier va exercer une action bien précise sur les billes : une force de friction. Celle-ci peut faire accélérer une bille, la ralentir et modifier sa trajectoire en fonction de la vélocité linéaire et angulaire de cette bille. Cette friction va créer des types de déplacement distincts pour une bille. En effet, il est possible d'observer qu'une bille peut rouler ou glisser sur le tapis de la table de billard. Chronologiquement, une bille en mouvement va commencer par glisser sur la table, puis rouler et enfin s'immobiliser.

La trajectoire d'une bille en mouvement n'est pas forcément rectiligne (fig. 1.1). En effet lorsque celle-ci glisse sur le tapis, elle peut dévier sur le côté pour donner une trajectoire curviligne. Le chemin que peut emprunter une bille est donc divisé en

deux parties : une partie plus ou moins curviligne suivi d'une partie rectiligne où la bille roule. L'étude de ces deux trajectoires est exposée par la suite. Il sera rajouté une partie sur les spins des billes et les mouvements aériens.

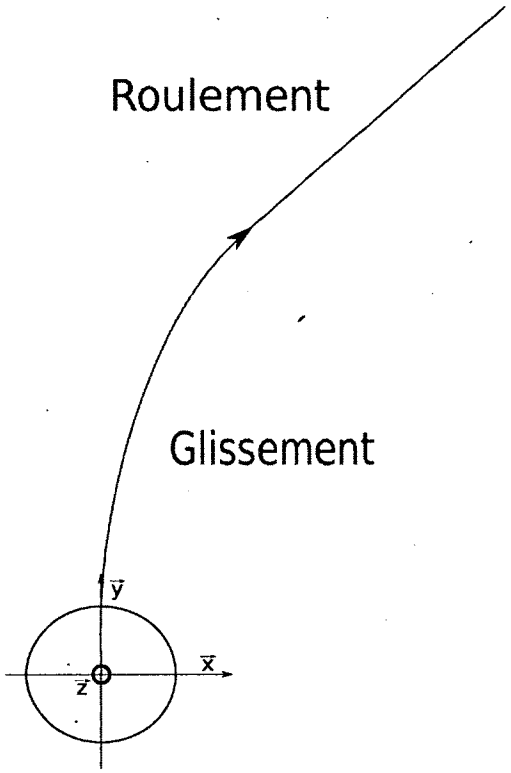

figure 1.1 – Exemple de trajectoire possible d'une bille.

- **Le Roulement** :
  Quand une bille roule sur le tapis, la bille se déplace d'un périmètre quand elle a fait un tour sur elle même. Il y a donc proportionnalité entre la vélocité angulaire et linéaire.

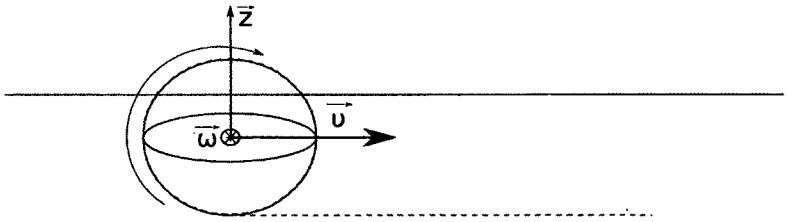

figure 1.2 – Bille en état de roulement.

Quand une bille roule sur le tapis, elle se déplace selon une ligne droite et est ralentie par une faible friction exercée par le tapis (fig. 1.2). Une bille en roulement finit par s'immobiliser pour rentrer en état stationnaire.

La vélocité linéaire est régie par : $\vec{v}(t) = \vec{v_0} - \mu_r\, g\, t\, \hat{v_0}$,
la vélocité angulaire par : $\vec{\omega}(t) = \frac{\vec{z} \times \vec{v}(t)}{R}$,

avec $\vec{v_0}$ qui est la vélocité linéaire à l'instant initial ($t = 0$), μ_r le coefficient de friction de roulement, g la constante de gravité et $\hat{v_0}$ la vélocité linéaire initiale normalisée [22], $\vec{z}$ le vecteur unitaire représentant l'axe z partant du centre de la bille dirigé vers le haut et R le rayon de la bille.

- **Le Glissement** :
  Lorsqu'une bille est en glissement sur le tapis, elle n'a pas la proportionnalité entre vélocité linéaire et angulaire, ce qui fait que la friction exercée par le tapis est bien plus élevée qu'en état de roulement. En fonction de la vélocité angulaire, la bille peut gagner ou perdre en vélocité linéaire et inversement pour la vélocité angulaire.

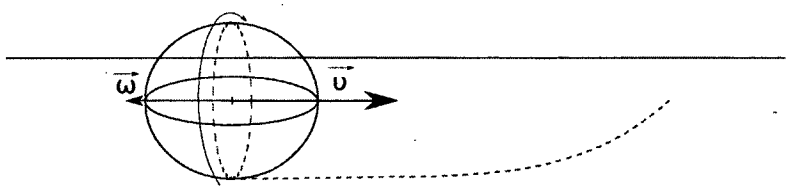

figure 1.3 – Bille en état de glissement.

1.1. Physique impliquée

En fonction de la vélocité au point de contact entre la bille et le tapis, la trajectoire de la bille peut devenir curviligne et totalement dévier de sa ligne de visée initiale (fig. 1.3). Cette vélocité est appelée vélocité relative au point de contact. Elle est calculée avec la vélocité linéaire ET angulaire. La bille sera très influencée par cette vélocité car son accélération linéaire et angulaire dépend de celle-ci.

Vélocité relative : $\vec{u}\,(t) = \vec{v}(t) + R\vec{z} \times \vec{\omega}(t)$.

Vélocité linéaire : $\vec{v}\,(t) = \vec{v_0} - \mu_s\, g\, t\, \hat{u_0}$.

Vélocité angulaire : $\vec{\omega}\,(t) = \vec{\omega_0} - \frac{5\mu_s g}{2R} t\,(\hat{z} \times \hat{u_0})$.

Dans les équations ci-dessus, μ_s est le coefficient de friction de glissement, $\hat{u_0}$ est la vélocité relative initiale normalisée, R est le rayon de la bille, $\hat{z}$ est le vecteur normalisé représentant l'axe z pointé vers le haut, $\times$ est le produit vectoriel [22].

La vélocité linéaire va se rapprocher de plus en plus de la relation de proportionnalité du régime de roulement au fur et à mesure que la bille évolue sur le tapis (fig. 1.4). Cette proportionnalité est atteinte lorsque la vélocité relative au point de contact devient nulle.
La vélocité relative est régie par : $\vec{u}(t) = \vec{u_0} - \frac{7}{2}\mu_s g t\, \hat{u_0}$.

Le régime de glissement représente finalement un état de « stabilisation » où la bille modifie ses vélocités afin que sa rotation soit en relation avec son déplacement sur la table. C'est sur ce régime de mouvement que misent les joueurs expérimentés pour faire des coups plus ou moins complexes et impressionnants.

- **Spin selon l'axe z :**
  Un autre type d'état de mouvement peut se manifester : le spin selon l'axe z (correspondant en anglais au terme de "right/left English"). Celui représente la rotation de la bille selon l'axe z. Cet état est spécial car il " s'additionne

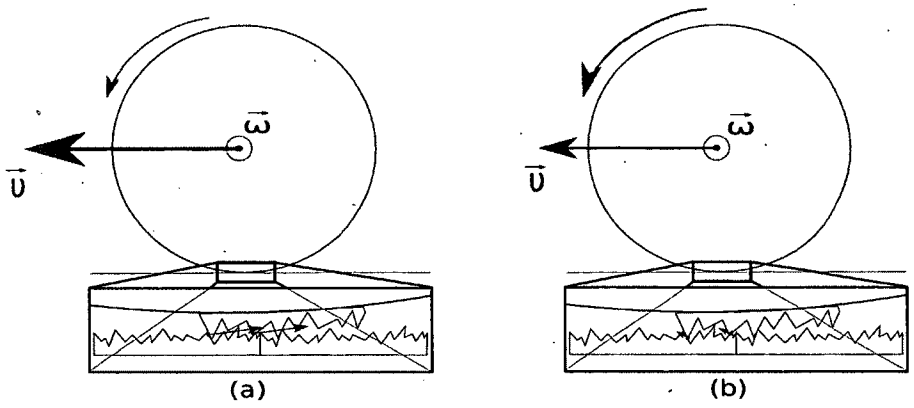

figure 1.4 – (a) La bille glisse sur l'autre objet : la friction est importante ; (b) La bille roule sur la surface de l'autre objet, il y a très peu de frottement.

" aux états vu précédemment. En effet une bille peut très bien se trouver en glissement ou en roulement avec du spin selon l'axe z. Il arrive même d'observer une bille stationnaire avec du spin selon l'axe z (la bille tourne sur elle même en restant sur place, comme pourrait le faire une toupie) (fig. 1.5).
Cet état est indépendant du régime de mouvement de la bille, ce qui veut dire que l'évolution de la vélocité angulaire selon l'axe z est indépendante des équations régissant la vélocité angulaire selon x et y dans les deux régimes de mouvement.

Vélocité angulaire en z : $\omega_z(t) = \omega_z(0) - \frac{5\mu_{sp}g}{2R}t$,

avec $\omega_z(0)$ la vélocité angulaire en z initiale, μ_{sp} le coefficient de frottement de spin selon l'axe z (ce dernier est très petit car le frottement ne s'exerçant qu'en un point, la bille est très peu ralentie par le tapis) [22].

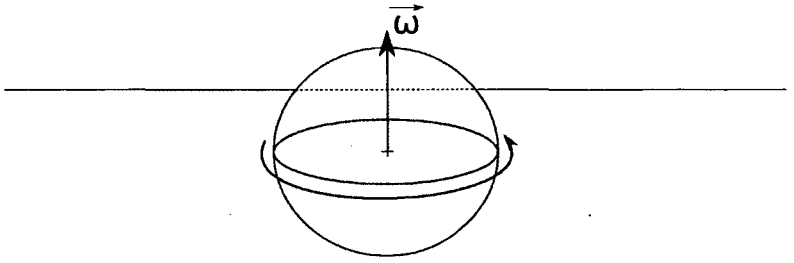

figure 1.5 – Cas d'une bille stationnaire ayant une composante z de vélocité angulaire non nulle (effet toupie).

Le spin est responsable de l'écartement des trajectoires résultantes des billes rentrant en collision avec d'autres billes/bandes. En effet cette vélocité angulaire de côté va « accrocher » les billes entre elles (pour prendre l'exemple d'une collision bille-bille). Ce phénomène sera vu plus en détails dans la partie sur les collisions.

Formes des trajectoires et effets simples

Comme vu précédemment, la trajectoire d'une bille en mouvement peut être de deux natures : linéaire si celle-ci est en régime de roulement ou curviligne si elle est en régime de glissement. Les équations de position de la bille sont pourtant de second degré dans les deux cas.

Équation de position en régime de glissement : $\vec{P}(t) = \vec{P_0} + \vec{v_0}t - \frac{1}{2}\mu_s\, g\, t^2\, \hat{u_0}$.
Équation de position en régime de roulement : $\vec{P}(t) = \vec{P_0} + \vec{v_0}t - \frac{1}{2}\mu_r\, g\, t^2\, \hat{v_0}$.

Pourquoi une trajectoire linéaire en régime de roulement dans ce cas ? Cela s'explique simplement par le fait que le coefficient $\frac{1}{2}\mu_r\, g\, \hat{v_0}$ est intimement lié à $\vec{v_0}$. L'accélération (coefficient en t^2) ne changera pas la direction de la vélocité $\vec{v_0}$, seulement sa norme. En régime de glissement, la vélocité relative n'est pas reliée seulement à la vélocité linéaire mais aussi à la vélocité angulaire, ce qui aura un effet plus ou moins conséquent sur la trajectoire dépendamment de cette dernière.

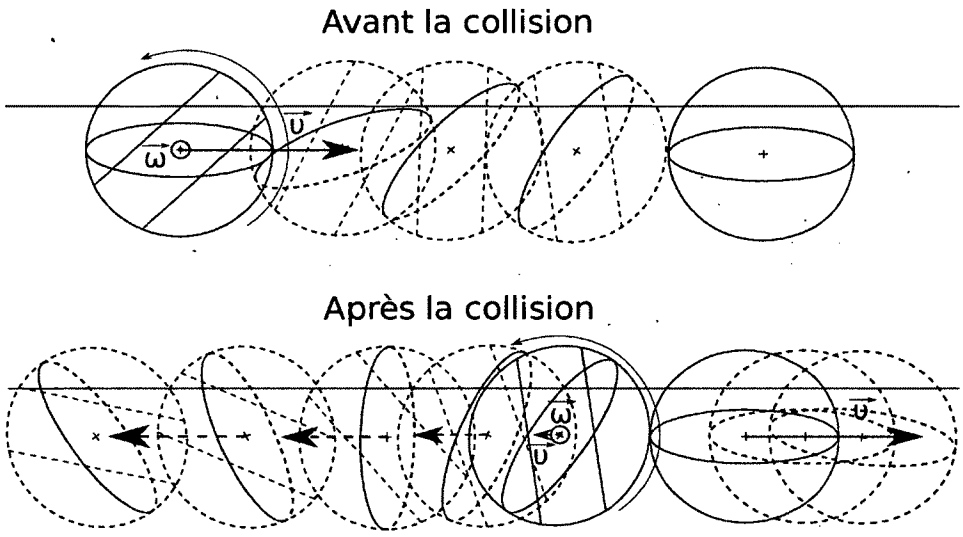

figure 1.6 – Principe du Draw Shot.

Les joueurs expérimentés savent se servir de la phase de glissement pour obtenir des effets plus ou moins complexes. Les collisions font généralement parties intégrantes de ces coups car elles permettent à une bille de « perdre » une partie de sa vélocité linéaire, tout en laissant quasiment intacte (plus ou moins dépendamment de l'objet que rencontre la bille, une autre bille ou une bande) la vélocité angulaire. Il est possible de courber ou de faire repartir une bille dans un sens opposé à sa vélocité pré-collision. Voici quelques exemples d'effets connus simples :

- Le *Draw Shot* :
  la bille a suffisamment de backspin (vélocité angulaire contraire à la vélocité linéaire, visuellement la bille tourne dans le sens opposé du régime de roulement) pour changer de sens après sa collision avec une autre bille (fig. 1.6).

- Le *Stun Shot* :
  la bille arrive en collision sans vélocité angulaire, elle s'arrête donc complètement avec une collision frontale avec une autre bille. Il s'agit ni plus ni moins d'un Draw Shot dont la vélocité angulaire est nulle lors de la collision.

- Le *Follow* :

la bille rentre en collision avec une vélocité angulaire qui va la pousser à suivre la bille avec laquelle elle est entrée en collision (phénomène observable avec une bille rentrant en collision en régime de roulement). Plus la bille aura une vélocité angulaire élevée, plus elle accélèrera. Il s'agit ni plus ni moins d'un Draw Shot avec une vélocité angulaire opposée.

Rentrer en collision en régime de roulement entraîne toujours un décalage de la bille de sa trajectoire linéaire idéale [26][1] (fig. 1.7). C'est le régime de glissement qui suit la collision qui est responsable de cette déviation : la vélocité angulaire change la trajectoire pour finalement trouver la proportionnalité entre celle-ci et la vélocité linéaire.

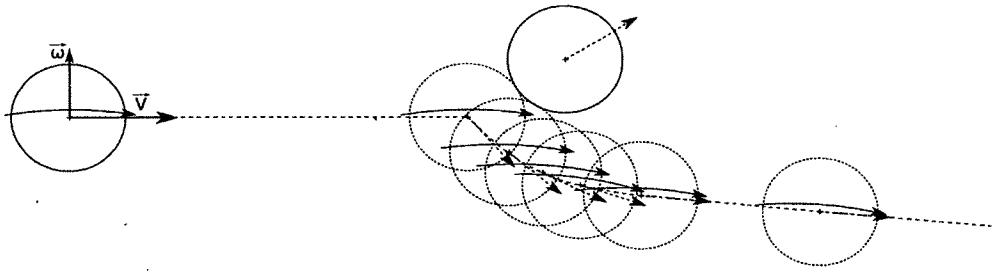

figure 1.7 – Exemple d'une modification de trajectoire après une collision en régime de roulement.

Déplacement aérien

Lorsqu'une bille quitte le tapis, une seule force agit sur son mouvement : la gravité. Plus aucun frottement n'est exercé sur la bille. Ceci implique l'ajout d'une hypothèse : les frottements de l'air sur les billes sont négligeables (fig. 1.8). Ainsi sa vélocité angulaire reste inchangée tant qu'elle n'a pas retouchée le tapis, de même que sa vélocité linéaire dans le plan horizontal. L'accélération n'agira que sur la composante z de la vélocité linéaire :

$$\text{Vélocité linéaire en } z : v_z(t) = v_z(0) - gt.$$
$$\text{Position en } z \text{ de la bille : } p_z(t) = p_z(0) + v_z(0)t - gt^2.$$

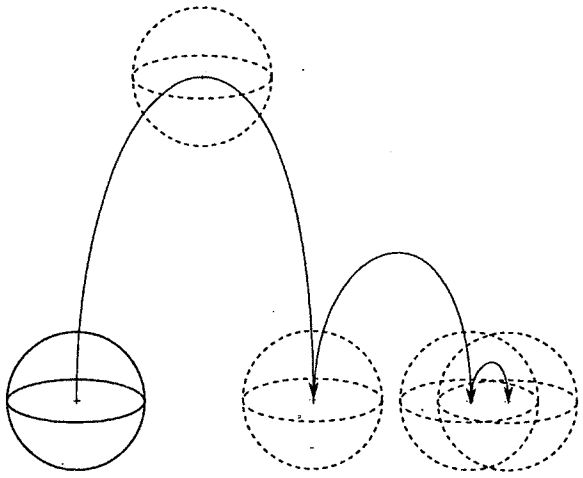

figure 1.8 – Trajectoire d'une bille en vol.

Les coups utilisant la composante aérienne du jeu de billard sont rares. Les plus utilisés sont les coups « massés » qui consistent à frapper la bille blanche avec la queue très inclinée vers la table, tout en communiquant une bonne vélocité pour que la bille rebondisse sur le tapis (quelques exemples dans [26][1][29]) (fig. 1.9). Ce sont des coups rarement utilisés car difficiles à prévoir et à « doser ». De plus, si la bille est ratée, il y a une bonne probabilité que la table en subisse les conséquences (tapis écrasé, déchiré voire une table dotée d'un nouveau trou).

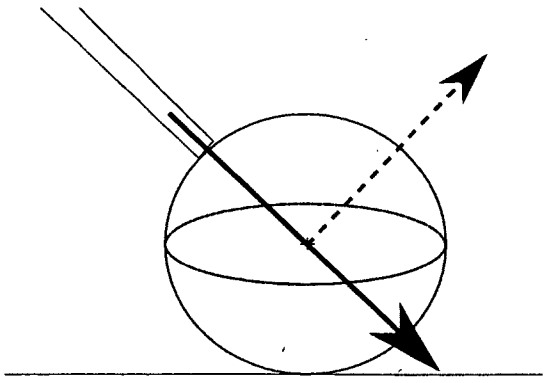

figure 1.9 – Exemple d'un coup massé.

Une bille peut aussi décoller après une collision violente entre une bille et une bande (fig. 1.10). En effet, la pointe formée par la bande au point où la bille rentre en collision se situe au-dessus du centre de la bille. On a donc une force dont la composante z est non-nulle. La bille va donc subir une force la poussant dans le tapis de jeu, ce qui provoquera donc une collision avec le tapis pouvant la faire décoller, dépendamment de la magnitude de la force exercée.

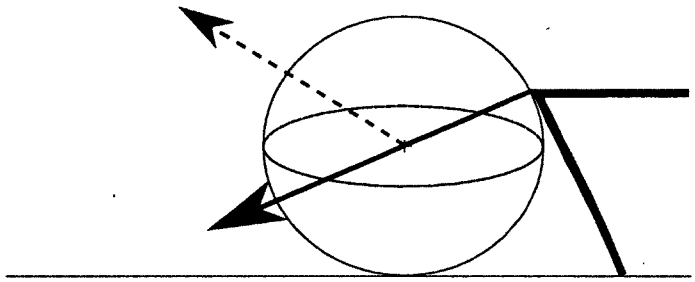

figure 1.10 – Exemple d'une collision entre une bille et une bande provoquant le décollage de la bille.

1.1.2 Collisions

Les collisions sont les évènements les plus importants du billard. Il est très essentiel de les représenter du mieux possible, le jeu n'étant finalement qu'une succession de collisions. Dans un premier temps, la collision queue-bille qui initie tout coup sera étudiée. Par la suite une étude des collisions entre deux objets rigides sera exposée, suivie de ses applications et spécifications dans le cadre du billard.

Collision queue-bille

Pour jouer son tour, un joueur utilise une grande baguette de bois dotée d'un bout en cuir, nommée queue de billard, pour frapper la bille blanche et lui communiquer un mouvement. Il y a beaucoup de possibilités de coups, une infinité pour être plus réaliste, mais ils peuvent tous être définis comme dépendants de 5 paramètres [19] (fig. 1.11) :

- a : l'écartement horizontal du bout de la queue avec le centre de la bille, c'est le paramètre qui est responsable de la quantité de spin que l'on communiquera à la blanche.

- b : l'écartement vertical du bout de la queue avec le centre de la bille, désignant la quantité de back/top spin que la blanche emmagasinera lors de la collision.

- θ : angle vertical que forme la queue avec la table, il s'agit de l'inclinaison de celle-ci. Plus l'angle est grand, plus la table est visée.

- ϕ : angle horizontal, c'est l'angle que la queue forme par rapport à un axe d'origine sur le plan de la table. Il s'agit de l'angle de visée.

- V : vitesse de la queue lors de l'impact.

Avec ces 5 paramètres, on peut trouver le point $Q(a, c, b)$ le point d'impact entre la queue et la bille (c s'obtient en utilisant $c = |\sqrt{R^2 - a^2 - b^2}|$). Avec ce point, il est possible de déterminer la force exercée par la queue sur la bille, et donc d'en déduire les vélocités communiquées à la blanche.

Tout d'abord, il faut considérer que le temps de collision est très petit ce qui permet de traiter la force exercée comme une impulsion. $\vec{F}$ étant la force exercée par la queue sur la blanche, en intégrant la Seconde Loi de Newton, il est possible d'écrire : $\vec{F} = m\vec{v}$ [29]. Soit le repère $\vec{x}$,$\vec{y}$, $\vec{z}$, dont l'origine est le centre de la bille, avec $\vec{y}$ sur la droite de « visée » de la queue. Il est possible d'exprimer $\vec{v}$ la vélocité linéaire résultante de la collision :

$$\vec{v} = \left(0,\ \tfrac{-F}{m}\cos\theta,\ \tfrac{-F}{m}\sin\theta\right),$$

avec m la masse de la bille, θ l'un des cinq paramètres et F la force calculée selon :

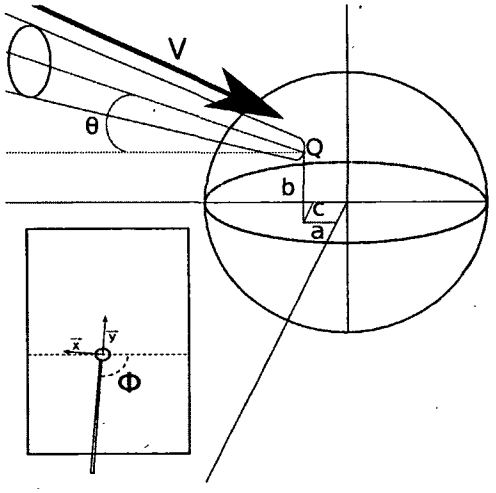

figure 1.11 – Les 5 paramètres et le point d'impact Q

$$F = \frac{2mV}{1+\frac{m}{M}+\frac{5}{2R^2}\left(a^2+b^2\cos^2\theta+c^2\sin^2\theta-2bc\cos\theta\sin\theta\right)},$$

où V est l'un des cinq paramètres, (a, b, c) sont les coordonnées du point de contact Q et M est la masse de la queue de billard [19]. La vélocité angulaire s'obtient en calculant :

$$\vec{\omega} = \frac{1}{I}\left(-cF\sin\theta + bF\cos\theta, aF\sin\theta, -aF\cos\theta\right),$$

avec $I = \frac{2}{5}mR^2$ le moment d'inertie d'une bille [19].

Collision entre deux corps rigides

Il est possible d'assimiler les collisions entre les objets au billard comme des cas spécifiques de collision entre deux objets rigides (fig. 1.12). Une étude générale de collision entre deux corps rigides permet d'avoir une base solide sur laquelle s'appuyer, puis de l'adapter en simplifiant certains aspects quand on la replace dans les collisions bille-bille ou bille-bande.

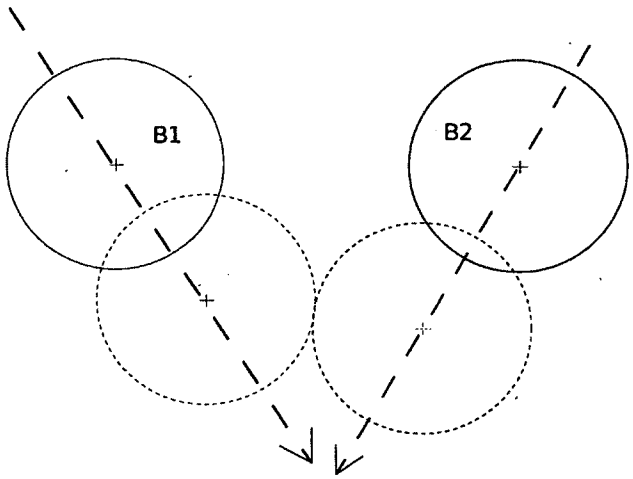

figure 1.12 – Collision entre deux billes

L'analyse de Régis Petit [26] prend en compte le cas général pour raffiner au cas du billard. Son analyse est développée par la suite.

– **Mécanique générale des chocs entre deux corps rigides** :

Il est supposé que les corps sont des objets de type sphérique.
En se servant des lois standards de la mécanique des collisions, il est possible de déterminer les vélocités linéaires et angulaires résultantes de deux objets rentrés en contact. La première de ces lois est la suivante : la quantité de mouvement lors de ce choc est conservée, ce qui se traduit par :

$$M_1\vec{v}_{1final} = M_1\vec{v}_{1init} + \vec{P},$$
$$M_2\vec{v}_{2final} = M_2\vec{v}_{2init} - \vec{P},$$

où $\vec{P}$ est le vecteur de percussion du choc, s'exerçant au point de contact et provenant du corps choqué, et M_x est la masse de l'objet x.
Le lien entre les vélocités linéaires finales et initiales est établi. La loi de la conservation du moment cinétique permet quand à elle de faire le lien entre les vélocités angulaires finales et initiales :

$$I\vec{\omega}_{1final} = I\vec{\omega}_{1init} + \overrightarrow{G_1C} \times \vec{P},$$
$$I\vec{\omega}_{2final} = I\vec{\omega}_{2init} + \overrightarrow{G_2C} \times (-\vec{P}),$$

avec I le moment d'inertie d'une bille, $\overrightarrow{G_xC}$ est le vecteur reliant le centre de l'objet x au point de contact C. Le vecteur $\vec{P}$ est donc en grande partie responsable des changement de vélocités dans la collision.
Pour obtenir ce vecteur $\vec{P}$, la loi de Coulomb sur les frottements dans le cas d'une collision entre deux solides, représentant le fait que le frottement s'exercera dans le sens opposé au mouvement au point de contact (fig. 1.13), doit être prise en compte :

$$\vec{P} = P\left(\vec{n} - \mu \frac{\vec{V}_{c\,init}}{\|\vec{V}_{c\,init}\|}\right),$$

$$P = kM\left(\vec{v}_{2init} - \vec{v}_{1init}\right).\vec{n},$$

$$\vec{V}_{c\,init} = (\vec{v}_{1init} - \vec{v}_{2init}) - [(\vec{v}_{1init} - \vec{v}_{2init}) \cdot \vec{n}]\,\vec{n} + \vec{n} \times (R_1\vec{\omega}_{1init} + R_2\vec{\omega}_{2init}).$$

Dans les équations ci-dessus, μ est le coefficient de frottement entre les deux objets, $\vec{n}$ est le vecteur unitaire normal à C dirigé vers le corps 1, $\vec{V}_{c\,init}$ est le vecteur vitesse de glissement du point C dans le mouvement relatif du corps 1 par rapport au corps 2. Il y a une constante à déterminer : k .
Pour compléter l'ensemble d'équations pour déterminer les vélocités résultantes des deux objets, il faut la règle des collisions de Newton :

$$(\vec{v}_{1final} - \vec{v}_{2final}) \cdot \vec{n} = -N \cdot (\vec{v}_{1init} - \vec{v}_{2init}) \cdot \vec{n},$$

avec N le coefficient d'élasticité entre les deux corps, et le théorème de l'énergie cinétique :

$$E_{c\,init} - E_{c\,final} = e\,E_{c\,init},$$

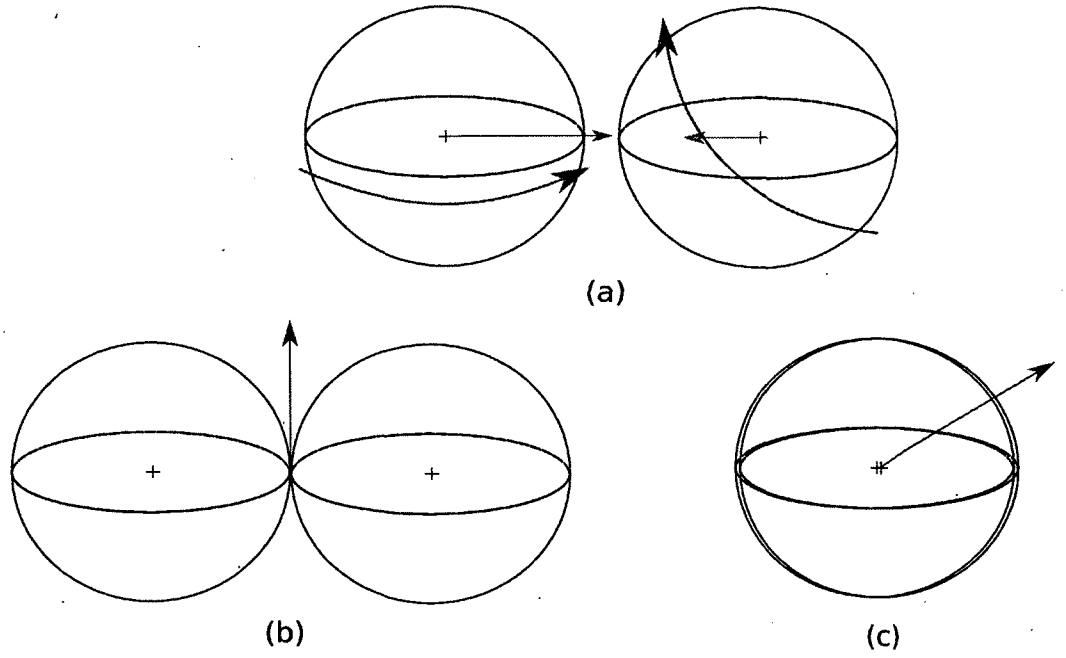

figure 1.13 – Représentation de la loi de Coulomb sur les frottements. En (a) deux billes ayant des vélocités angulaires et linéaires différentes vont entrer en collision. A l'instant (b), les billes sont en contact, le vecteur vitesse de glissement $\vec{V}_{cinit}$ d'un corps sur l'autre est déterminé en prenant en compte les vélocités angulaires et linéaires des deux solides. Ce vecteur est représenté en mauve ici et est associé au corps de gauche sur celui de droite. La loi de Coulomb impose que le frottement appliqué au premier corps (ici celui de gauche) soit opposé au vecteur vitesse de glissement de ce dernier sur l'autre (celui de droite). Le deuxième corps subira un frottement opposé. En (c), un autre point de vue de la scène est exposé. On reconnaît l'influence des vélocités angulaires des deux corps.

avec e le pourcentage d'énergie perdue pendant le choc. Pour rappel, l'énergie cinétique du système se calcule selon :

$$E_c = \frac{1}{2}\left(M_1\vec{v_1}^2 + I\vec{\omega_1}^2 + M_2\vec{v_2}^2 + I\vec{\omega_2}^2\right)$$

L'ensemble d'équations est constitué de 5 équations et 5 inconnues ($\vec{v}_{1final}$, $\vec{v}_{2final}$, $\vec{\omega}_{1final}$, $\vec{\omega}_{2final}$, k), avec une résolution vectorielle, la solution unique obtenue est :

$$\begin{cases} \vec{v}_{1final} = \vec{v}_{1init} + \frac{P}{M_1}\left(\vec{n} - \mu\frac{\vec{V}_{cinit}}{\|\vec{V}_{cinit}\|}\right) \\ \vec{\omega}_{1final} = \vec{\omega}_{1init} + \frac{5}{2R_1}\mu\frac{P}{M_1}\left(\vec{n} \times \frac{\vec{V}_{cinit}}{\|\vec{V}_{cinit}\|}\right) \\ \vec{v}_{2final} = \vec{v}_{2init} + \frac{M_1}{M_2}\left(\vec{v}_{1init} - \vec{v}_{1final}\right) \\ \vec{\omega}_{2final} = \vec{\omega}_{2init} - \left(\frac{M_1}{M_2}\right)\left(\frac{R_1}{R_2}\right)\left(\vec{\omega}_{1init} - \vec{\omega}_{1final}\right) \\ k = \frac{1+N}{1+\frac{M_1}{M_2}} \end{cases}$$

Il y a cependant un point supplémentaire à prendre en compte : l'adhérence. En effet lorsque la vélocité au point de contact $\vec{V}_{cinit}$ n'est pas suffisante pour vaincre le frottement, les deux objets sont en contact étroit sans qu'ils puissent glisser l'un sur l'autre. Ce phénomène est repérable grâce à la relation suivante : $\vec{V}_{cfinal} = u\vec{V}_{cinit}$.
Un indice d'adhérence est donc introduit pour déterminer si ce phénomène se produit ou non.

$$u = 1 - \frac{7}{2}\mu\frac{\left(1+\frac{M_1}{M_2}\right)\left(\frac{P}{M_1}\right)}{\|\vec{V}_{cinit}\|}.$$

La détermination de u est explicitée en annexe A. Si $u < 0$, alors le frottement n'est pas vaincu par $\vec{V}_{cinit}$ et il y a adhérence. La solution va donc changer en prenant en compte cet effet : il faut remplacer $\mu\left(\frac{P}{M_1}\right)\left(\frac{1}{\|\vec{V}_{cinit}\|}\right)$ par $\frac{\frac{2}{7}}{\left(1+\frac{M_1}{M_2}\right)}$.
Le coefficient de frottement n'est plus présent. La solution devient donc :

$$\begin{cases} \vec{v}_{1final} = \vec{v}_{1init} + \frac{P}{M_1}\vec{n} - \left(\frac{2}{7}\right)\frac{\vec{V}_{cinit}}{1+\frac{M_1}{M_2}} \\ \vec{\omega}_{1final} = \vec{\omega}_{1init} + \frac{5}{7R_1}\left(\vec{n} \times \frac{\vec{V}_{cinit}}{1+\frac{M_1}{M_2}}\right) \\ \vec{v}_{2final} = \vec{v}_{2init} + \frac{M_1}{M_2}\left(\vec{v}_{1init} - \vec{v}_{1final}\right) \\ \vec{\omega}_{2final} = \vec{\omega}_{2init} - \left(\frac{M_1}{M_2}\right)\left(\frac{R_1}{R_2}\right)\left(\vec{\omega}_{1init} - \vec{\omega}_{1final}\right) \\ k = \frac{1+N}{1+\frac{M_1}{M_2}} \end{cases}$$

- **Collision entre deux billes** :
  Lors d'une collision entre deux billes, il est possible de simplifier les solutions obtenues : en effet les deux objets sont ici sphériques, de même dimension et de même masse. Cela nous donne donc les simplifications suivantes pour un choc avec adhérence :

$$\begin{cases} \vec{v}_{1final} = \vec{v}_{1init} + \frac{P}{M}\vec{n} - \frac{\vec{V}_{cinit}}{7} \\ \vec{\omega}_{1final} = \vec{\omega}_{1init} + \frac{5}{7R}\left(\vec{n} \times \frac{\vec{V}_{cinit}}{2}\right) \\ \vec{v}_{2final} = \vec{v}_{2init} + \left(\vec{v}_{1init} - \vec{v}_{1final}\right) \\ \vec{\omega}_{2final} = \vec{\omega}_{2init} - \left(\vec{\omega}_{1init} - \vec{\omega}_{1final}\right) \\ k = \frac{1+N}{2} \end{cases}$$

Les simplifications $M_1 = M_2$, $R_1 = R_2$ s'appliquent de la même façon pour un choc sans adhérence :

$$\begin{cases} \vec{v}_{1final} = \vec{v}_{1init} + \frac{P}{M_1}\left(\vec{n} - \mu\frac{\vec{V}_{cinit}}{\|\vec{V}_{cinit}\|}\right) \\ \vec{\omega}_{1final} = \vec{\omega}_{1init} + \frac{5}{2R_1}\mu\frac{P}{M_1}\left(\vec{n} \times \frac{\vec{V}_{cinit}}{\|\vec{V}_{cinit}\|}\right) \\ \vec{v}_{2final} = \vec{v}_{2init} + \left(\vec{v}_{1init} - \vec{v}_{1final}\right) \\ \vec{\omega}_{2final} = \vec{\omega}_{2init} - \left(\vec{\omega}_{1init} - \vec{\omega}_{1final}\right) \\ k = \frac{1+N}{2} \end{cases}$$

Les vélocités linéaires et angulaires peuvent donc être calculées facilement et donc rapidement.

- **Collision entre une bille et une bande (ou le tapis) :**
  Dans le cas du choc entre une bille et un objet de type « obstacle inamovible », le premier objet représente la bille et le deuxième cet obstacle. Ce dernier ne bougera pas après le choc (ses vélocités linéaires et angulaires resteront nulles) ce qui réduira le nombre de calculs à effectuer. De plus le deuxième objet a une masse beaucoup plus importante que la bille, ainsi le rapport $\frac{M_1}{M_2}$ sera nul.
  La solution d'un coup sans adhérence sera donc du type :

$$\begin{cases} \vec{v_{1final}} = \vec{v_{1init}} + \frac{P}{M}\left(\vec{n} - \mu\frac{\vec{V_{cinit}}}{\|\vec{V_{cinit}}\|}\right) \\ \vec{\omega_{1final}} = \vec{\omega_{1init}} + \frac{5}{2R}\mu\frac{P}{M}\left(\vec{n} \times \frac{\vec{V_{cinit}}}{\|\vec{V_{cinit}}\|}\right) \\ k = 1 + N \end{cases}$$

  Les simplifications sont aussi, évidemment, valables pour un coup avec adhérence. Ce qui différenciera les collisions bille/bande et bille/tapis sera le vecteur $\vec{n}$ (dans le cas d'une collision bille/tapis, ce dernier sera toujours ramené au vecteur $\vec{z}$ (0, 0, 1)).

1.1.3 Hypothèses simplificatrices

Le simulateur créé ne sera pas parfait. Il existe en effet des phénomènes dans la physique mécanique réelle qui ne seront pas traités dans le simulateur, ou seront simplifiés. Ces derniers sont les suivants :

- Collisions simultanées : les collisions simultanées sont évitées dans le simulateur ce qui pourrait affecter la crédibilité du simulateur dans certaines situations (fig. 1.14). Il n'en est pourtant rien ! En effet les situations impliquant des collisions simultanés ont une probabilité égale à 0 sur le plan de la table. Il existera

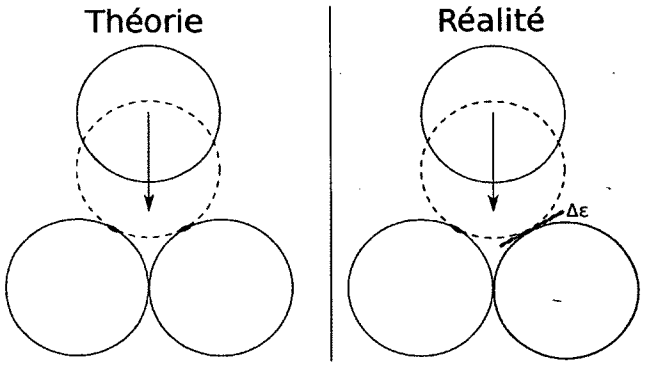

figure 1.14 – Illustration de collisions simultanées.

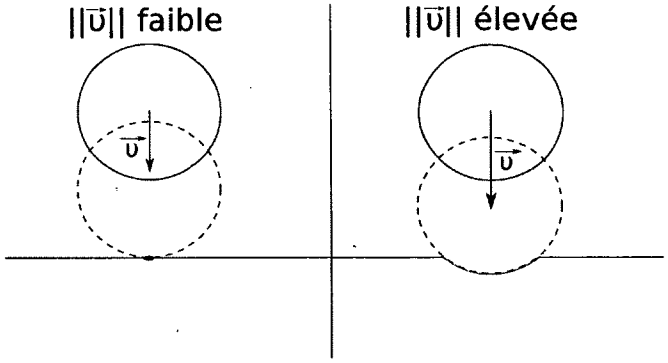

figure 1.15 – Illustration de la déformation de la bande.

toujours une distance $\Delta\epsilon$ qui permet de hiérarchiser chronologiquement toutes les collisions.

- Déformation des bandes : le gros point faible quand au respect de la réalité. En effet les bandes ne sont pas rigides ! Elles peuvent se déformer en fonction de la vélocité de la bille incidente. Ceci a pour conséquence que la collision ne s'effectue plus en un seul point mais sur une surface subissant les phénomènes de compression et de décompression (fig. 1.15). Évidemment le traitement est plus complexe en prenant en compte cet aspect.

- Collisions avec des objets déformables : les bandes ne sont pas les seules à se déformer ! Les autres éléments du jeu de billard sont beaucoup plus rigides que

les bandes mais rien n'est parfaitement rigide. Même les billes devraient subir des déformations minimes lors de collisions entre elles.

1.2 Simulation discrète

Il existe plusieurs sortes de simulation, parmi lequelles on retrouve la simulation discrète synchrone et la simulation discrète asynchrone (par évènements)[14][9]. Expliquons ces deux types pour mettre en lumière leur différence. La simulation synchrone sera reliée à un pas de temps (ou time-step) et à chaque pas, avancera les objets selon leur vélocités, tout en vérifiant les règles physiques établies au préalable (on peut faire référence à la règle de non-interpénétration des objets par exemple). Il s'agit donc d'un type de simulation basé sur la «vérification» qu'un évènement se produit ou non à chaque timestep. La simulation asynchrone par contre va prévoir les changements avant d'y être confronté. Les deux principes de simulation sont donc différents dans leur approche de détection d'un changement majeur dans l'environnement simulé. Chaque approche a ses avantages et inconvénients. Dans une première partie, la simulation discrète synchrone sera exposée, suivie dans une deuxième partie par la simulation discrète asynchrone. Enfin, l'algorithme général du simulateur sera exposé.

1.2.1 Simulation discrète synchrone

La simulation discrète synchrone est la plus intuitive. En effet l'idée est simple : il s'agit ni plus ni moins d'avancer d'un temps fixé et de vérifier que les objets de la scène n'enfreignent pas certaines règles.
Ayant les équations de déplacement des billes dans le temps en fonction de leur état (glissement, roulement) il est possible de déterminer leur position, vélocités après un pas de temps. Dans le cas de la collision entre deux objets, il faut vérifier à chaque pas de temps que ces derniers ne rentrent pas l'un dans l'autre. Si c'est le cas, alors une collision se produit. Dans le cadre du billard, il faut donc surveiller pour chaque bille :

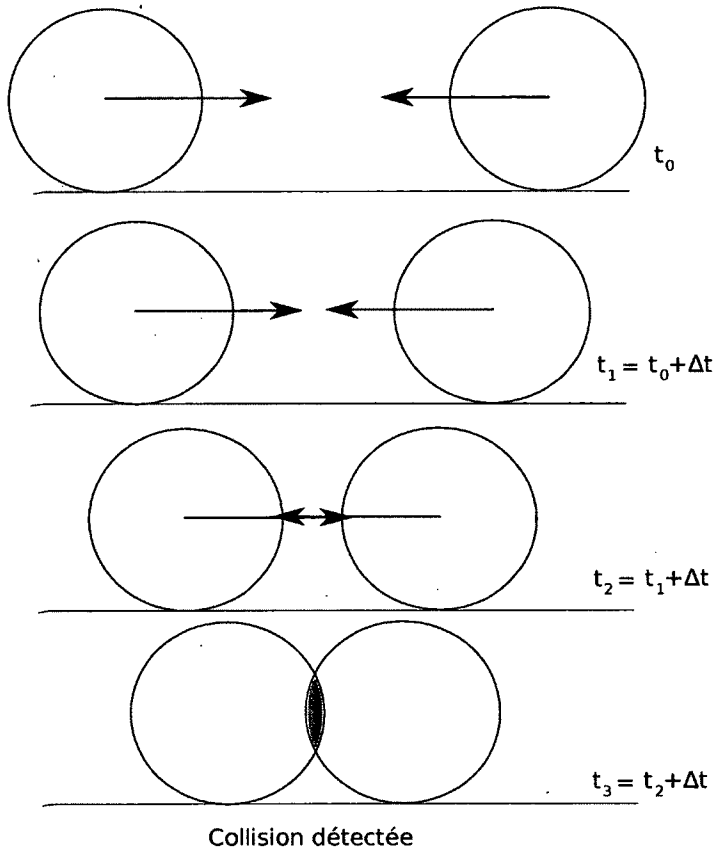

figure 1.16 – Détection d'une collision en simulation discrète synchrone

- Changements d'état : lorsque la vélocité relative devient nulle, passer du glissement au roulement. De même, lorsque la vélocité linéaire devient nulle, passer du roulement au stationnaire.

- Collisions : vérifier que deux billes ne s'interpénètrent pas, en vérifiant qu'il y a une distance d'au moins un diamètre de bille entre leur centre (fig. 1.16. C'est aussi le cas des bandes ou le tapis, pour lesquels la distance est cette fois un rayon.

1.2. Simulation discrète

Cette méthode de simulation est l'approche la plus répandue pour représenter la physique mécanique dans un environnement virtuel. Cette technique est populaire car elle est réaliste visuellement et simple à programmer, mais elle est cependant loin d'être précise et physiquement correcte. À chaque pas de temps des erreurs numériques seront accumulées à cause des approximations linéaires effectuées par le simulateur. En effet, à chaque pas de temps, la nouvelle position d'un objet est calculée en supposant que sa vélocité est constante. De même, la nouvelle vélocité sera calculée en prenant une accélération constante. Cela donne :

- P_0 : Position d'un objet au temps 0 ; V_0 : Vélocité au temps 0 ; a_0 : Accélération au temps 0

- $P_1 = P_0 + V_0.\Delta t$ la position de l'objet au temps 1
- $V_1 = V_0 + a_0.\Delta t$ la vélocité de l'objet au temps 1

Il est supposé que l'accélération est constante durant Δt , de même que la vélocité. Il y a donc un décalage avec la réalité, même si Δt est petit. La précision est donc « sacrifiée » pour la simplicité. Le but d'un simulateur étant de reproduire au mieux la réalité, il est clair que cette imprécision est un point négatif.
Cependant, il existe une autre caractéristique de la simulation discrète synchrone qui va accentuer le besoin d'un autre type de simulation. Malgré le fait que Δt puisse être très petit, il y a peu de chances pour qu'un évènement se produise exactement en $t_0 + \alpha\Delta t$ avec $\alpha \in \mathbb{N}$. Il est quasi certain que celui-ci arrive à un temps $t_0 + \alpha\Delta t + t_\epsilon$, avec $t_\epsilon < \Delta t$, et donc compris dans un intervalle de temps Δt . Les évènements sont donc en grande majorité détectés *après coup* (fig. 1.16). Ceci implique de revenir au temps précis où l'évènement s'est produit. Deux possibilités s'offrent alors : soit on revient en arrière au temps t pour le traiter (entraînant une nouvelle perte de précision et des calculs plus ou moins évidents dans le cas de collisions entre objets), soit on recule de Δt (là aussi la perte de précision est à souligner) et on diminue le pas de temps pour arriver sur l'évènement. Pour résumer, la détection et le traitement des évènements complexifient grandement les calculs tout en continuant de perdre de la précision.

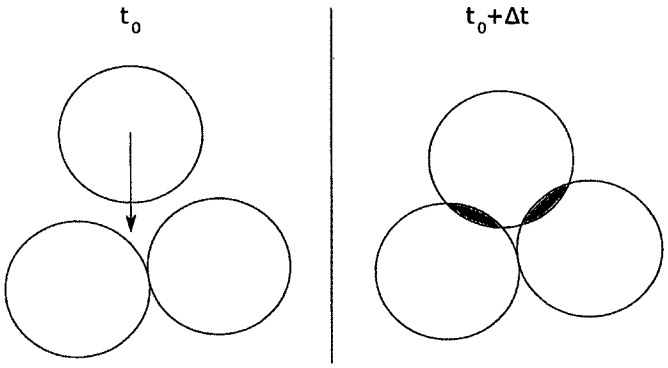

figure 1.17 – Situation délicate pour une simulation discrète synchrone

Parmi les faiblesses de ce type de simulation, il est possible de rajouter le fait que certains évènements seront probablement très difficiles à détecter. Une situation simple pour l'illustrer est la suivante : deux billes sont au repos et en contact, une troisième arrive pour frapper les deux (fig. 1.17), si les deux collisions se produisent dans le même Δt , laquelle traiter en premier ?

Ce problème devient de plus en plus lourd avec le nombre de billes impliquées. Effectuer une casse est donc un problème majeur avec ce type de simulation.
La validité d'une simulation discrète synchrone est donc très sensible au pas de temps Δt choisi. Plus ce dernier sera grand, plus on aura une simulation rapide mais peu précise. A l'inverse, un Δt très petit permettra d'éviter les pertes de précisions et de limiter les cas problématique comme celui de la casse, au prix d'une lenteur due à l'alourdissement des calculs.
Le but du nouveau simulateur étant d'être le plus précis possible, l'approche discrète synchrone est donc un choix risqué. Il est préférable de se porter sur une autre alternative : la simulation discrète asynchrone.

1.2.2 Simulation discrète asynchrone

La simulation discrète asynchrone fonctionne de façon radicalement différente de son homologue synchrone. Ici, les évènements ne seront pas déterminés *après coup*

mais *avant*. En disposant les informations nécessaires à la prédiction des possibles évènements, le simulateur avancera d'évènement en évènement, sans utiliser de pas de temps fixe. Le principal avantage de cette approche est d'avoir des résultats exacts pour les caractéristiques (positions, vélocités, ...) des objets. Dans une majorité des cas, le simulateur sera aussi plus rapide avec cette approche que celle utilisée dans la simulation discrète synchrone (ceux qui ne rentrent pas dans cette majorité seront exposés par la suite).
Le deuxième avantage de ce type de simulation est pour sa simplicité d'utilisation par un joueur IA. En effet ce dernier pourra établir des arbres d'évènements plus facilement qu'avec la simulation synchrone, ce qui facilitera sa prise de décision.

Tout d'abord, il est commode de définir le terme d'« évènement » :

Évènement : Phénomène considéré comme localisé et instantané, survenant en un point et un instant bien déterminé. Il représente un changement brutal dans le système simulé. Ceci implique tout changement de comportement des billes (le passage de l'accélération de glissement à celle du roulement illustre autant un changement brutal qu'une collision entre deux billes).

Pour que le simulateur puisse fonctionner parfaitement, il doit se soumettre à trois hypothèses :

- **Hypothèse 1 :** Les évènements peuvent tous être prédits. Dans le cas contraire, cela impliquerait que des évènements puissent intervenir à tout moment, or le simulateur n'avance que vers des évènements qu'il peut prévoir.

- **Hypothèse 2 :** Les évènements sont tous instantanés. Si un évènement n'était pas instantané, le simulateur devrait donc avancer jusqu'à celui-ci puis le traiter, avec une chance qu'un autre évènement se passe dans cet intervalle de temps.

- **Hypothèse 3 :** Deux évènements ne peuvent être simultanés. Si deux évène-

ments se produisaient au même instant, les changements de l'un sur les objets du monde simulé pourraient invalider l'autre.

Si l'une de ces trois hypothèses est invalidée par un évènement pouvant survenir dans la simulation, alors ce dernier serait probablement à l'origine d'une erreur fatale pour refléter le réalisme du simulateur. Il sera exposé plus loin des cas problématiques, reliés à l'introduction de la 3D, pour le respect de ces hypothèses.

Les évènements traité par le simulateur sont les suivants :

- **CUE STRIKE :**
  Il s'agit du coup de queue initial dans la bille blanche, qui va définir les cinq paramètres utilisés. Ces derniers seront ensuite utilisés pour donner à la bille blanche ses vélocités initiales. Cet évènement est bien entendu le premier de toute séquence d'évènements composant ce qui sera appelé un *SHOT*, entité qui regroupera tous les évènements se produisant entre le coup de queue initial et l'immobilisation de toutes les billes (empochées ou sur la table).

- **CHANGEMENT D'ÉTAT :**
  Une bille va rentrer dans un nouvel état : il peut s'agir de la transition entre un régime de mouvement à un autre. Ces régimes sont :

  - *SLIDING* : la bille glisse sur le tapis de jeu, subissant une forte friction.

  - *ROLLING* : la bille roule sans glisser sur le tapis. La friction subie est minime.

  - *SPINNING* : la bille a une vélocité angulaire en z non nulle. En réalité, le terme « spinning » ne correspond pas forcément à la composante z, mais étant donné que celle-ci est la seule à être régie par des équations indépendantes de l'état de la bille, l'emploi du terme n'est pas choquant. Ce régime de mouvement correspond en fait à un aspect secondaire d'un état principal. Par exem-

ple, une bille peut glisser sur le tapis tout en étant soumise au *SPINNING*. De même pour une bille à l'arrêt (il n'est pas rare de voir une bille tourner telle une toupie alors que sa position sur la tapis reste inchangée). Pour résumer, le *SPINNING* introduit trois états de mouvement : le *SLIDING_SPINNING*, le *ROLLING_SPINNING* et le *SPINNING* (le dernier correspondant précisément au cas de la toupie).

- *FLYING* : la bille vole au dessus de la zone de jeu. Seule sa vélocité linéaire en z est influencée par cet état de mouvement.

- *STATIONARY* : la bille est immobile.

- **COLLISIONS :**
  Les collisions sont l'élément principal du jeu de billard, plusieurs évènements y sont donc rattachés. Ceux-ci traiteront évidemment les changements physiques à opérer lors du choc. Les différentes collisions rencontrées sont les suivantes :

  - *COLLISION BILLE/BANDE* : une bille rentre en contact avec un des rebords de la table (les bandes sont illustrées sur la fig. 1.18. Ceci provoque un changement radical pour les vélocités linéaire et angulaire de cette bille.

  - *COLLISION BILLE/BILLE* : deux billes rentrent en collision l'une avec l'autre.

  - *COLLISION BILLE/TAPIS* : une bille en vol (ou subissant une force l'écrasant sur le tapis) atterrit sur la tapis de jeu.

La liste des évènements étant maintenant complète, il est nécessaire de savoir comment prédire ceux-ci :

- **PRÉDICTION D'UN CHANGEMENT D'ÉTAT** : tout d'abord, il faut savoir que tous les changements d'état ne se prédisent pas avec les mêmes équa-

tions. Il faut donc traiter ceux-ci en fonction des transitions réelles entre états de mouvement (par exemple une bille en *SLIDING* ne pourra que s'orienter vers un régime de roulement).

- *De SLIDING à ROLLING* ou *De SLIDING_SPINNING à ROLLING_SPINNING* : Une bille est en régime de glissement tant que sa vélocité relative est non-nulle. L'équation de la vélocité relative dans le temps étant : $\vec{u}(t) = \vec{u_0} - \frac{7}{2}\mu_s g t\,\hat{u_0}$ [22], il est trivial de définir le temps pour lequel cette dernière s'annule :
$$t_S = \frac{2|\vec{u_0}|}{7\mu_s g}$$

- *De ROLLING à STATIONARY* ou *De ROLLING_SPINNING à SPINNING* : Une bille est en régime de roulement tant que sa vélocité linéaire est non-nulle (sa vélocité angulaire dépendant de celle-ci, elle ne sera donc pas prise en compte). L'équation de la vélocité linéaire dans le temps étant : $\vec{v}(t) = \vec{v_0} - \mu_r\, g\, t\,\hat{v_0}$, il est évident que le temps pour lequel cette dernière s'annule est :
$$t_R = \frac{|\vec{v_0}|}{\mu_R g}$$

- *De state_SPINNING à state* : Une bille est en régime dit SPINNING quand la composante z de sa vélocité angulaire est non-nulle. Cette transition n'est pas comme les précédentes, soit entre deux états précis, car le SPINNING se combine à un autre état de bille. L'équation de la composante z de la vélocité angulaire dans le temps étant : $\omega_z = \omega_{z_0} - \frac{5\mu_{SP} g}{2R} t$ [22], il est évident que le temps pour lequel cette dernière s'annule est :
$$t_{SP} = \frac{\omega_{z_0} 2R}{5\mu_{SP} g}$$

Il est facile de prédire ces évènements : avec les outils fournis dans la partie (physique impliquée), les durées des états correspondent à des équations linéaires simples.

Une remarque peut cependant être observée : il n'existe pas de prédiction de changement d'état impliquant le régime de mouvement *FLYING*. En effet une bille sur le tapis passe par une collision pour s'envoler (collision avec le tapis la

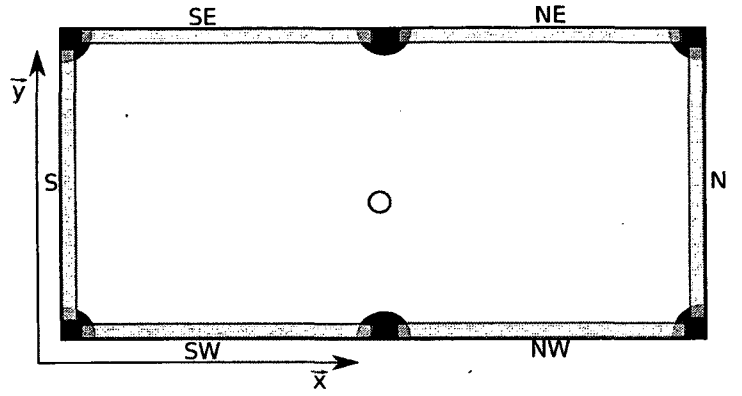

figure 1.18 – Illustration de la table et des différentes bandes

faisant rebondir dans les airs) et à l'inverse, une bille en vol rentrant en contact avec le tapis de jeu provoque une collision qui, si la vélocité en z est petite débouchera vers un régime de mouvement de type *SLIDING*, sinon la bille s'envolera de nouveau.

- **PREDICTION D'UNE COLLISION** : La prédiction de ces évènements sera ici beaucoup moins simple que pour les changements d'état. Les différentes collisions se prédisent de façon différentes et sont explicitées par la suite :

  - *COLLISION BILLE/BANDE* : Pour prévoir ces évènements, il faut calculer quand une bille arrivera sur l'un des bords de la surface de la table. La méthode de détection est cependant plus complexe si la bille est actuellement en vol (état *FLYING*). Il sera d'abord présenté le cas où la bille est sur le tapis. Il est possible de séparer le processus en deux étapes :

    1. Calcul de l'arrivée de la bille sur un bord : il suffit de prendre une une des composantes x ou y dépendamment de la bande ciblée et de calculer quand la bille arrivera à une distance de $bord - RAYON$. L'exemple suivant illustre ce calcul :

       Pour trouver si la bille va entrer en collision avec les bandes SE/NE, il suffit de calculer le $t_{SE/NE} \geq 0$ le temps pour lequel $p_y = Table_{LARGEUR} - RAYON$. Sachant que $p_y = p_{y_0} + v_{y_0} t - \frac{1}{2}\mu_s\, g\, t^2\, \hat{u_{y_0}}$ (il est donc supposé

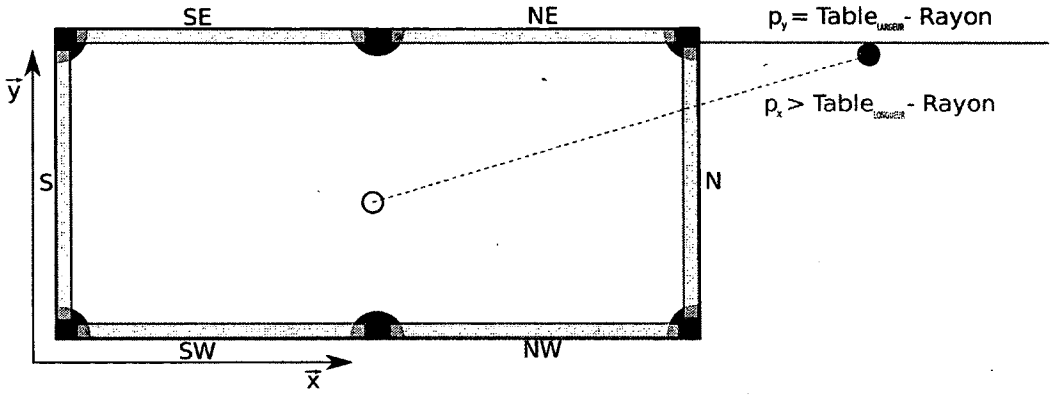

figure 1.19 – La détection de la collision avec la bande NE est invalidée par la position en x de la bille

que la bille est en régime de glissement), $t_{SE/NE}$ est donc solution de cette dernière quadratique avec $p_y = Table_{LARGEUR} - RAYON$. Si les solutions sont négatives, la bille ne rentrera pas en collision avec ce côté de la table. Sinon, la vérification de l'étape 2 peut avoir lieu.

2. Vérification de l'évènement : il a été confirmé que la bille allait entrer en contact avec ce côté de la table, cependant il faut vérifier deux conditions pour que l'évènement soit en effet une collision avec une des bande NE ou SE : la première consiste à vérifier que la bille est encore sur la table. La quadratique a très bien pu renvoyer une solution où la bille se trouve en dehors de la surface de jeu (fig. 1.19), il faut donc vérifier pour cet exemple que $p_x \leq Table_{LONGUEUR} - RAYON$ et $p_x \geq RAYON$ (avec la bande S correspondant à $x = 0$) (voir figure) ; la deuxième représente la possibilité que l'évènement soit en fait l'empochage de la bille : si $p_x > Poche_{E_xLeft}$ et $p_x < Poche_{E_xRight}$ alors l'évènement détecté est $POCKET$ (l'empochage de la bille) et non la collision avec la bande.

Pour résumer, il faut donc résoudre une quadratique pour une des composantes x ou y de la bille et vérifier que l'autre est comprise sur la table et pas dans une poche (fig. 1.20).

Pour le cas d'une bille en vol, la détection se complexifie : il ne s'agit plus de

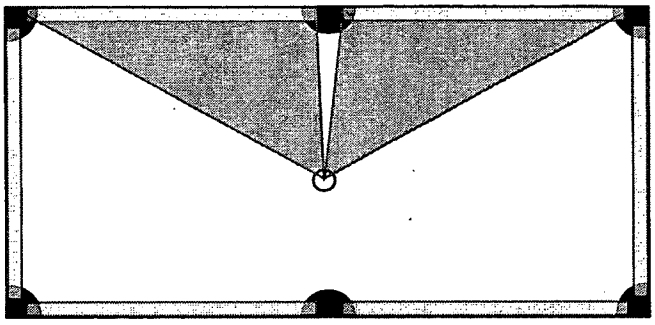

figure 1.20 – Détection validée d'une collision de bande (position finale de la bille sur la table et en dehors d'une poche)

détecter quand une des composantes x ou y atteint une bordure de la zone de jeu, il faut aussi vérifier que la bille ne passe pas au dessus de cette dernière (fig. 1.21). Pour expliciter les nouvelles difficultés, l'exemple d'une bille en vol testée pour une collision avec la bande SE/NE est repris :

1. Calcul du temps pour lequel la bille arrive sur la bande. Ici il ne s'agira plus d'une quadratique car on a besoin de deux composantes : y et z. Avec $\vec{P}_{SE/NE} = \begin{bmatrix} P_y \\ P_z \end{bmatrix}$ la position de la bande, $\vec{P}_{bille} = \begin{pmatrix} P_{0_y} + v_{0_y} t \\ P_{0_z} + v_{0_z} t - \frac{1}{2} g t^2 \end{pmatrix} = \left(\vec{P}_0 + \vec{v}_0 t + \frac{1}{2}\vec{a}_0 t^2\right)$ la position de la bille selon y et z, le temps t_{col} de la collision correspond à $\|\vec{P}_{bille} - \vec{P}_{SE/NE}\|^2 = RAYON^2$. En développant :

$$\|\vec{P}_{bille} - \vec{P}_{SE/NE}\|^2 = RAYON^2$$

$$\left(\vec{P}_{bille} - \vec{P}_{SE/NE}\right)^T \left(\vec{P}_{bille} - \vec{P}_{SE/NE}\right) = RAYON^2$$

$$\left(\left(\vec{P}_0 - \vec{P}_{SE/NE}\right)^T + \vec{v}_0^T t + \tfrac{1}{2}\vec{a}_0^T t^2\right)\left(\left(\vec{P}_0 - \vec{P}_{SE/NE}\right) + \vec{v}_0 t + \tfrac{1}{2}\vec{a}_0 t^2\right) = RAYON^2$$

$$\left(\vec{P}_0 - \vec{P}_{SE/NE}\right)^T \left(\vec{P}_0 - \vec{P}_{SE/NE}\right) + \left(\vec{P}_0 - \vec{P}_{SE/NE}\right)^T \vec{v}_0 t + \left(\vec{P}_0 - \vec{P}_{SE/NE}\right)^T \tfrac{1}{2}\vec{a}_0 t^2 + \vec{v}_0^T t \left(\vec{P}_0 - \vec{P}_{SE/NE}\right) + \vec{v}_0^T \vec{v}_0 t^2 + \tfrac{1}{2}\vec{v}_0^T \vec{a}_0 t^3 +$$

$$\frac{1}{2}\vec{a}_0^T\left(\vec{P}_0 - \vec{P}_{SE/NE}\right)t^2 + \frac{1}{2}\vec{a}_0^T\vec{v}_0 t^3 + \frac{1}{4}\vec{a}_0^T\vec{a}_0 t^4 - RAYON^2 = 0$$

$$\begin{gathered}\frac{1}{4}\|\vec{a}_0\|t^4 + \left(\frac{1}{2}\vec{v}_0^T\vec{a}_0 + \frac{1}{2}\vec{a}_0^T\vec{v}_0\right)t^3 + \\ \left[\frac{1}{2}\left(\vec{P}_0 - \vec{P}_{SE/NE}\right)^T\vec{a}_0 + \frac{1}{2}\vec{a}_0^T\left(\vec{P}_0 - \vec{P}_{SE/NE}\right) + \|\vec{v}_0\|\right]t^2 + \\ \left[\left(\vec{P}_0 - \vec{P}_{SE/NE}\right)^T\vec{v}_0 + \vec{v}_0^T\left(\vec{P}_0 - \vec{P}_{SE/NE}\right)\right]t + \\ \left(\|\vec{P}_0 - \vec{P}_{SE/NE}\| - RAYON^2\right) = 0\end{gathered}$$

Il est possible de reconnaître une équation quartique (équation d'ordre 4) de la forme $ax^4 + bx^3 + cx^2 + dx + e = 0$ avec :

$$a = \frac{1}{4}\|\vec{a}_0\|$$

$$b = \left(\frac{1}{2}\vec{v}_0^T\vec{a}_0 + \frac{1}{2}\vec{a}_0^T\vec{v}_0\right) = \vec{v}_0^T\vec{a}_0$$

$$\begin{gathered}c = \left[\frac{1}{2}\left(\vec{P}_0 - \vec{P}_{SE/NE}\right)^T\vec{a}_0 + \frac{1}{2}\vec{a}_0^T\left(\vec{P}_0 - \vec{P}_{SE/NE}\right) + \|\vec{v}_0\|\right] = \\ \left[\left(\vec{P}_0 - \vec{P}_{SE/NE}\right)^T\vec{a}_0 + \|\vec{v}_0\|\right]\end{gathered}$$

$$d = \left[\left(\vec{P}_0 - \vec{P}_{SE/NE}\right)^T\vec{v}_0 + \vec{v}_0^T\left(\vec{P}_0 - \vec{P}_{SE/NE}\right)\right] = 2\left(\vec{P}_0 - \vec{P}_{SE/NE}\right)^T\vec{v}_0$$

$$e = \left(\|\vec{P}_0 - \vec{P}_{SE/NE}\| - RAYON^2\right)$$

Cette équation donnera entre 0 et 4 racines réelles, la résolution de quartiques sera vue plus tard. S'il n'y a aucune racine réelle, ou si les racines réelles sont toutes négatives, alors il n'y a pas de collision. Dans le cas contraire, les vérifications de l'étape deux peuvent avoir lieu.

2. Vérification de l'évènement : comme sur le plan de la table, il faut tester si la bille est sur la table. Ici la vérification peut aboutir à trois résultats différents : si la composante x de la bille est calculée comme étant en dehors de la table au moment de la collision, alors la bille devient OUT

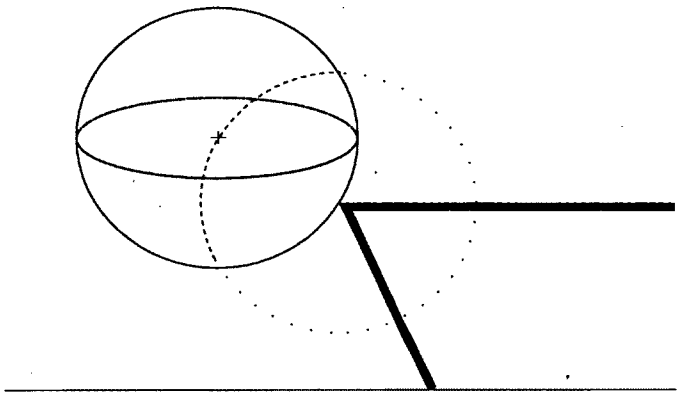

figure 1.21 – Exemple d'une collision Bille/Bande

(la bille a sauté par dessus une bande pour sortir de la table) ; si la composante x correspond à une poche (à l'instar de l'exemple sur le plan de la table) alors la bille est empochée ; si les deux conditions précédentes sont invalidées, alors il y a une collision avec la bande SE/NE.

- *COLLISION BILLE/BILLE* : Pour calculer le temps où deux billes vont se rencontrer, il faut déterminer t_{BB} pour lequel $\|\vec{P}_{B1} - \vec{P}_{B2}\|^2 = 4RAYON^2$ avec $\vec{P}_{B1}$ la position de la bille 1 dans le temps et $\vec{P}_{B2}$ la position de la bille 2 dans le temps.

$$\vec{P}_{B1} = \left(\vec{P}_{B1_0} + \vec{v}_{B1_0}t + \tfrac{1}{2}\vec{a}_{B1_0}t^2\right)$$
$$\vec{P}_{B2} = \left(\vec{P}_{B2_0} + \vec{v}_{B2_0}t + \tfrac{1}{2}\vec{a}_{B2_0}t^2\right)$$

Pour simplifier la notation, on prends : $\vec{P}_{B0} = \vec{P}_{B1_0} - \vec{P}_{B2_0}$, $\vec{v}_{B0} = \vec{v}_{B1_0} - \vec{v}_{B2_0}$ et $\vec{a}_{B0} = \frac{1}{2}\vec{a}_{B1_0} - \frac{1}{2}\vec{a}_{B2_0}$.

En développant :

$$\|\vec{P}_{B1} - \vec{P}_{B2}\|^2 = 4 * RAYON^2$$

$$\left(\vec{P}_{B1} - \vec{P}_{B2}\right)^T \left(\vec{P}_{B1} - \vec{P}_{B2}\right) = 4 * RAYON^2$$

$$\left(\vec{P}_{B0}^{T} + \vec{v}_{B0}^{T}t + \vec{a}_{B0}^{T}t^2\right)\left(\vec{P}_{B0} + \vec{v}_{B0}t + \vec{a}_{B0}t^2\right) = 4RAYON^2$$

$$\|\vec{P}_{B0}\| + \left(\vec{P}_{B0}\right)^T(\vec{v}_{B0})\,t + \left(\vec{P}_{B0}\right)^T(\vec{a}_{B0})\,t^2 + (\vec{v}_{B0})^T\left(\vec{P}_{B0}\right)t + \|\vec{v}_{B0}\|t^2 + (\vec{v}_{B0})^T(\vec{a}_{B0})\,t^3 + (\vec{a}_{B0})^T\left(\vec{P}_{B0}\right)t^2 + (\vec{a}_{B0})^T(\vec{v}_{B0})\,t^3 + \|\vec{a}_{B0}\|t^4 - 4RAYON^2 = 0$$

$$\|\vec{a}_{B0}\|t^4 + \left[(\vec{a}_{B0})^T(\vec{v}_{B0}) + (\vec{v}_{B0})^T(\vec{a}_{B0})\right]t^3 + \left[\left(\vec{P}_{B0}\right)^T(\vec{a}_{B0}) + (\vec{a}_{B0})^T\left(\vec{P}_{B0}\right) + \|\vec{v}_{B0}\|\right]t^2 + \left[\left(\vec{P}_{B0}\right)^T(\vec{v}_{B0}) + (\vec{v}_{B0})^T\left(\vec{P}_{B0}\right)\right]t + \left(\|\vec{P}_{B0}\| - 4RAYON^2\right) = 0$$

Ici aussi il est possible de reconnaître une équation quartique avec :

$$a = \|\vec{a}_{B0}\|$$

$$b = \left[(\vec{a}_{B0})^T(\vec{v}_{B0}) + (\vec{v}_{B0})^T(\vec{a}_{B0})\right]$$

$$c = \left[\left(\vec{P}_{B0}\right)^T(\vec{a}_{B0}) + (\vec{a}_{B0})^T\left(\vec{P}_{B0}\right) + \|\vec{v}_{B0}\|\right]$$

$$d = \left[\left(\vec{P}_{B0}\right)^T(\vec{v}_{B0}) + (\vec{v}_{B0})^T\left(\vec{P}_{B0}\right)\right]$$

$$e = \left(\|\vec{P}_{B0}\| - 4RAYON^2\right)$$

Si aucune racine n'est réelle ou positive, alors les billes ne se rencontreront pas. A l'inverse, si une solution réelle et positive existe alors les deux billes rentreront en collision à ce temps précis. Il faut tout de même faire une vérification supplémentaire, les erreurs numériques pouvant se glisser dans la résolution de l'équation quartique. Celle-ci se décompose comme suit :

1. Calculer les vélocités linéaires au temps t_{BB} calculé.
2. Projeter ces vélocités sur le vecteur normal reliant les deux billes (de la bille 1 vers la bille 2). v_{1proj} sera la vélocité projeté de la bille 1 et v_{2proj}

celle de la bille 2.

3. Si $v_{1proj} * v_{2proj} < 0$ et $v_{1proj} < 0$ alors il n'y a pas de collision (les vélocités partent dans des directions opposées). La vérification s'arrête ici si la collision est invalidée.
4. Si $v_{1proj} < 0$ et $v_{1proj} < v_{2proj}$ alors il n'y a pas de collision (les vélocités sont dans la même direction mais la bille 1 va plus vite que la bille 2, entraînant leur séparation). La vérification s'arrête ici si la collision est invalidée.
5. Si $v_{1proj} \geq 0$ et $v_{1proj} < v_{2proj}$ alors il n'y a pas de collision (même cas que précédemment, mais dans la direction opposée).

Si toutes les vérifications sont passées avec succès, la collision est validée.

- *COLLISION BILLE/TAPIS* : Cette collision est très simple à prédire : il suffit de résoudre une équation quadratique : $P_z(t) = P_{0_z} + v_{0_z}t - \frac{1}{2}gt^2$ avec $P_z(t) = RAYON$. Si l'une des racines est positive, alors la plus petite indique le temps la bille touche la zone de jeu. Afin d'éviter de tomber sur une racine nulle erronée, le test de la vélocité linéaire de la bille en z est efficace : si celle-ci est non-négative, alors la collision est invalide.

Maintenant que tous les évènements possibles sont connus et prédictibles, l'algorithme général du simulateur peut être présenté.

1.3 Algorithme général

Chaque coup représente une collision entre la queue de billard et la bille de choc (la blanche) qui rentrera en collision avec un ou plusieurs obstacles, bandes ou billes objets, ces dernières pourront par la suite en heurter d'autres, jusqu'à ce que toutes les billes s'immobilisent sur la table ou soient empochées. Le but étant de rentrer toutes les billes dans les poches, ce processus sera répété jusqu'à ce qu'il n'y ai plus

d'autres bille sur la table que la bille de choc.

Pour représenter ce processus, le simulateur s'appuie sur deux entités :

- *TABLE* : la table représente tous les éléments de jeu à disposition des joueurs. Elles contient les informations de dimension de la zone de jeu, le temps de jeu écoulé, les différentes options activées (le déblocage de l'axe z en faisant partie) et bien sûr les billes. Chaque bille contient les informations de ses positions et de ses vélocités propres. C'est sur cette entité *TABLE* que vont s'opérer TOUS les changements. Les évènements n'auront d'effet que sur celle-ci. Tous les éléments "matériels" du billard seront compris à l'intérieur. C'est le cœur sur lequel va s'appuyer toute la simulation. les autres entités vont se concentrer sur des actions visant à modifier *TABLE*.

- *SHOT* : comme vu précédemment, un *SHOT* représente une succession d'évènements, du premier coup de queue de billard dans la bille blanche jusqu'à l'immobilisation de la dernière bille en mouvement sur la table. En créant un *SHOT*, un *CUE STRIKE* est traité avec les cinq paramètres a, b, θ, ϕ et v, puis, tant que l'une des billes de la table est en mouvement, le prochain évènement sera déterminé et traité. Ce dernier découle du simple procédé suivant :

  1. Détermination du prochain évènement pour chaque bille : les billes seront testées séparément avec les formules de prédictions d'évènement vues précédemment pour savoir quand et quelle sera la nature du prochain évènement. Il s'agit bien évidemment de l'évènement ayant le temps le plus proche du temps de jeu actuel, le plus proche chronologiquement.

  2. Détermination de l'évènement retenu : ayant stockés les prochains évènements des billes, il reste à isoler celui dont le temps avant son apparition est le plus court. Il s'agit ni plus ni moins de choisir le t le plus petit parmi les évènements générés pour chaque bille (fig. 1.22).

Une fois qu'un évènement est détecté comme étant le prochain, *SHOT* avance le

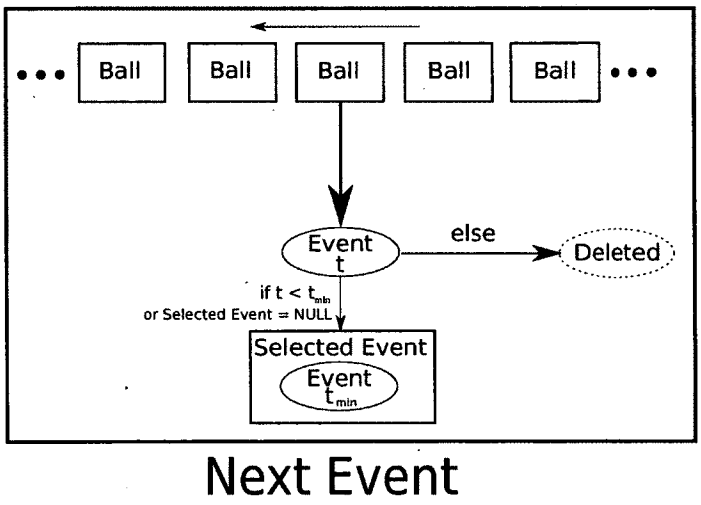

figure 1.22 – Schéma de la détermination du prochain évènement : un évènement (Event) est généré pour chaque bille sur la table. Celui-ci prend la place de l'évènement sélectionné si son temps avant apparition est plus court que celui de l'évènement stocké ($t < t_{min}$) ou si il n'y a pas encore d'évènement stocké. Dans le cas contraire, l'évènement créé est détruit. Ceci correspond au block « Next Event » dans le schéma de l'algorithme général.

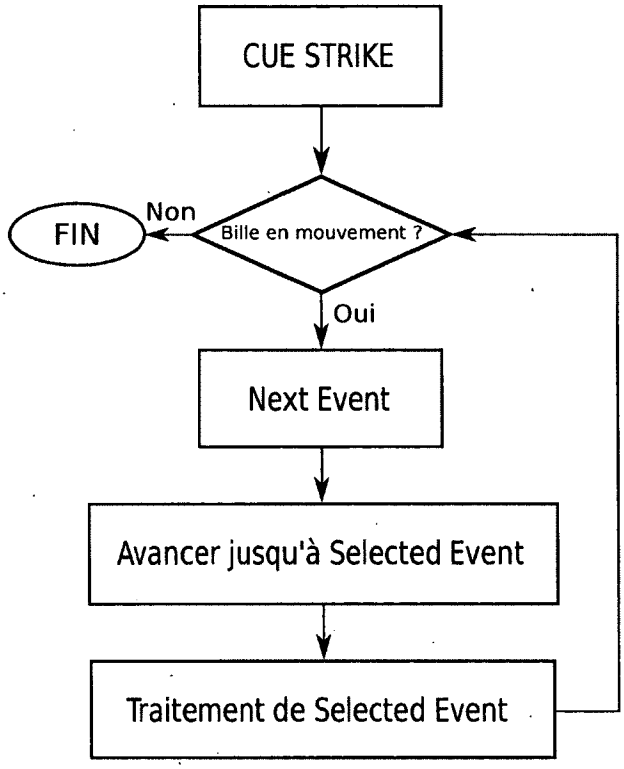

figure 1.23 – Schéma de l'algorithme général de la simulation. Il s'agit entre autres de l'action de *SHOT*.

temps de jeu jusqu'à l'instant où l'évènement se déclare. Les billes sont avancées sur la table selon leur équation de mouvement (dépendamment de leur état de mouvement). Une fois la table mise à jour, l'évènement est traité, impliquant tous les changements nécessaires sur *TABLE*. Une fois cette étape terminée, si au moins une des billes de la table est en mouvement, le processus recommence à la détermination du prochain évènement (fig. 1.23).

Pour optimiser le processus, il est important de se rendre compte que les billes n'étant pas impliquées dans l'évènement sélectionné et traité gardent l'évènement précédemment calculé dans « Détermination du prochain évènement pour chaque bille ». Le recalculer serait une perte de temps non-négligeable, il est donc utile d'éviter ces calculs. Pour ce faire, on utilise une liste d'évènements potentiels. Celle-ci contient

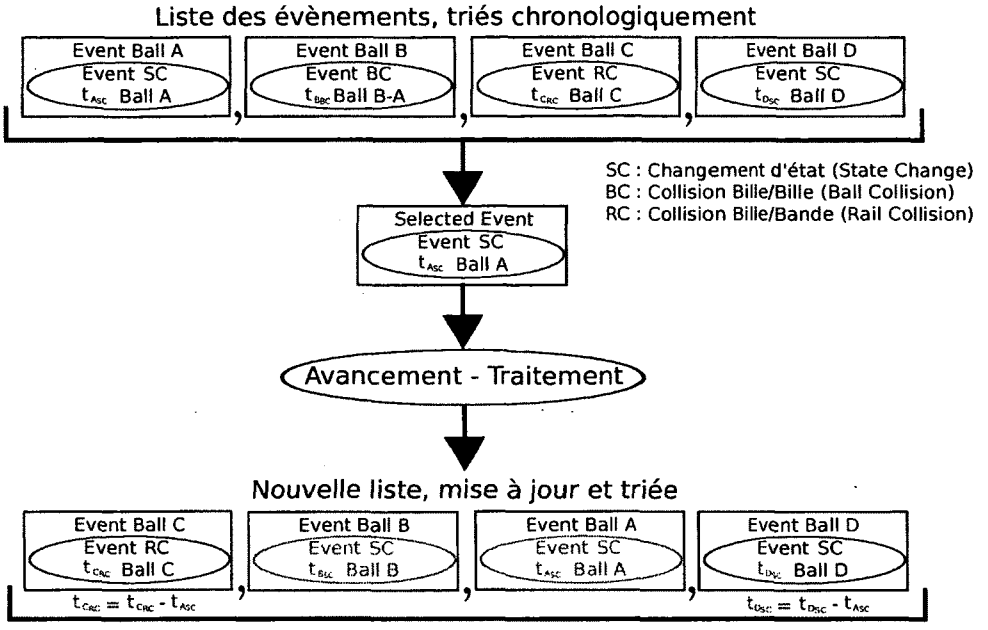

figure 1.24 – Exemple représentant la liste d'évènements et son influence. Le prochain évènement (à l'extrême gauche de la liste) est traité par le simulateur, la liste étant ensuite mise à jour en fonction ici de la bille A. Les évènements des billes A et B sont recalculés, car celui de B impliquait auparavant la bille A. Les évènements des billes C et D voient simplement leur temps respectifs mis à jour avec le temps de l'évènement traité. On évite ainsi des calculs inutiles.

tous les évènements calculés par l'étape « Détermination du prochain évènement pour chaque bille », triés du plus proche au plus lointain chronologiquement. L'évènement le plus proche sera traité par le simulateur et retiré de la liste. Tous les évènements n'étant pas en relation avec la ou les billes impliquées dans l'évènement traité ne seront pas recalculés : seul les temps seront mis à jour, en étant réduit d'une durée t, temps correspondant au dernier évènement traité. Les billes impliquées passeront quand à elles par le processus normal de détermination d'évènement (fig. 1.24).

Pour illustrer le temps de calcul économisé, dans le cas restreint du plan de la table (restriction 2D), à chaque fois qu'un évènement sera traité, les calculs nécessaires seront :

- Cas non-optimisé :
  Pour chaque bille :
  - 1-2 opérations pour la détection de changement d'état (2 si la bille est en *SPINNING*)
  - 4 quadratiques pour les collisions de bandes
  - 15 quartiques pour les collisions de billes

  Dans le pire des cas (16 billes en mouvement), on a donc :

  16-32 opérations + 64 quadratiques + 240 quartiques.

- Cas optimisé avec l'utilisation de la liste d'évènements : Le nombre de calcul pour chaque bille isolée est le même, cependant un évènement implique au maximum deux billes, ce qui veut dire qu'à chaque évènement traité, seules ces deux billes et les autres qui les avaient pour cible dans leur prochain évènement sont soumises à ces calculs. On a donc une grande majorité des cas où seules deux billes sont concernées, ce qui correspond à :

  2-4 opérations + 8 quadratiques + 30 quartiques.

Certains travaux [21] ont optimisé ce type de simulation afin qu'il devienne encore plus rapide, seulement, toutes ces approches n'étaient pas initialement prévues dans le simulateur développé. Il n'est cependant pas exclu que ces modifications soient ajoutées plus tard.

Pour conclure, si un joueur voulait se servir du simulateur, il n'aurait qu'à lui communiquer les cinq paramètres pour effectuer un *CUE STRIKE*, créant ainsi un *SHOT* qui lui reverra l'état de table final avec, s'il le souhaite, la liste d'évènements effectué par le simulateur pour arriver à ce résultat. Ceci lui permettra de savoir si les cinq paramètres sont appropriés ou si des changements doivent être fait sur ces derniers pour réussir son coup.

Chapitre 2

Outils développés et problèmes physiques

Dans ce chapitre sera détaillé la création du résolveur d'équations quartiques cité précédemment (section/paragraphe sur la détection collision billes/bandes aériennes) ainsi que les problèmes rencontrés avec ce dernier. De plus, de nouveaux problèmes physiques nécessitant un traitement spécial seront détaillés, de leur cause jusqu'à leur traitement. Les changements nécessaires sur l'algorithme général seront ensuite présentés, mettant en avant l'algorithme final du simulateur.

2.1 Résolveur d'équations quartiques

Dans le chapitre 1, dans la section sur la simulation asynchrone, il est fait référence à la résolution d'équations quartiques. Ces équations d'ordre 4 sont bien plus compliquées à résoudre que des équations quadratiques ou cubiques. Les résolveurs existants pour ce type d'équation fonctionnent souvent avec des approches itératives. Par exemple, le résolveur de polynômes supérieurs à l'ordre 3 dans la bibliothèque de calculs GSL (GNU Scientific Library) [13] utilise une méthode basée sur la décomposition QR de la matrice compagnon du polynôme. Cette technique est très robuste et trouve toujours (ou presque, dans les limites de la précision numérique) les solutions de l'équation. Il existe cependant quelques inconvénients associés à cette technique :

tout d'abord, dans le cadre du billard, les solutions imaginaires sont inutiles or la décomposition QR les calculera ce qui représente du temps de calcul inutile ; d'autres part, les méthodes itératives ne sont pas toujours rapides, se basant sur un nombre d'itérations pouvant rapidement augmenter. En vue de ces problèmes, il a été décidé de créer un résolveur qui pourrait pallier à ces défauts. Ce dernier se baserait donc sur une méthode analytique (une seule itération) et ne calculerait que les solutions réelles utiles. La méthode analytique sera dans un premier temps exposée, suivie des astuces utilisées pour compenser certains problèmes numériques et des résultats obtenus.

2.1.1 Algorithme de résolution de quartiques

L'idée de la méthode développée est de réduire l'équation quartique à deux équation quadratiques dont la résolution est simple [6]. Le point de départ est donc une équation du type : $x^4 + ax^3 + bx^2 + cx + d = 0$ (si le coefficient en x^4 est différent de 1, alors l'équation est normalisée selon ce dernier).
Cette équation peut s'écrire sous la forme : $x^4 + ax^3 = -bx^2 - cx - d$
Il est ajouté aux membres de gauche et de droite l'élément $\frac{1}{4}a^2x^2$, l'équation donne donc :

$$\left(x^2 + \tfrac{1}{2}ax\right)^2 = \left(\tfrac{1}{4}a^2 - b\right)x^2 - cx - d$$

Une nouvelle variable y est ensuite introduite via l'ajout de $y\left(x^2 + \frac{1}{2}ax\right) + \frac{1}{4}y^2$ des deux côtés de l'équation :

$$\left(x^2 + \tfrac{1}{2}ax + \tfrac{1}{2}y\right)^2 = \left(\tfrac{1}{4}a^2 - b + y\right)x^2 + \left(-c + \tfrac{1}{2}ay\right)x + \left(-d + \tfrac{1}{4}y^2\right)$$

Le membre de droite peut s'écrire comme $Ax^2 + Bx + C$.
L'objectif est que cette quadratique soit un carré parfait (les deux racines sont égales), ce qui implique $B^2 - 4\,AC$. Cela donne :

$$\left(-c+\tfrac{1}{2}ay\right)^2 = 4\left(\tfrac{1}{4}a^2-b+y\right)\left(-d+\tfrac{1}{4}y^2\right)$$
$$y^3-by^2+(ac-4d)\,y+4bd-a^2d-c^2=0$$

En résolvant la cubique ci-dessus, la quartique peut s'écrire :

$$\left(x^2+\tfrac{1}{2}ax+\tfrac{1}{2}y\right)^2=(ex+f)^2,$$

$$\text{avec } e=\sqrt{\tfrac{a^2}{4}-b+y} \quad \text{et} \quad f=\frac{-c+\frac{ay}{2}}{2\sqrt{\frac{a^2}{4}-b+y}}$$

Les deux équations quadratiques finales sont donc :

$$x^2+\tfrac{1}{2}ax+\tfrac{1}{2}y=ex+f \qquad \text{et} \qquad x^2+\tfrac{1}{2}ax+\tfrac{1}{2}y=-ex-f$$

Ces deux équations donneront les zéro à quatre racines de l'équation quartique initiale. Cette méthode, bien que simple, recèle des calculs sensibles qui peuvent vite aboutir via des erreurs numériques à l'échec total de la méthode (une racine négative par exemple). De plus, il est très important de remarquer l'importance de la résolution de la cubique dans cette démarche : si celle si n'est pas effectuée de façon précise, il est fort à parier que les résultats finaux en pâtiront sévèrement.
Le processus utilisé pour résoudre la cubique est simple et utilisé par la plupart des résolveurs. Il s'agit de la méthode suivante : avec $x^3+bx^2+cx+d=0$ (l'équation est normalisée par le coefficient en x^3), il faut obtenir deux valeurs qui serviront de discriminant pour le calcul des racines :

$$q=(b^2-3c) \qquad r=(2b^3-9bc+27d)$$

Pour des soucis de précision numérique, les variables additionnelles suivantes sont calculées :

$$Q = \frac{q}{9} \qquad R = \frac{r}{54} \qquad CR^2 = 729r^2 \qquad CQ^3 = 2916q^3$$

Si R et Q sont nuls, les trois racines sont réelles et égales : $-\frac{b}{3}$
Sinon :

- Si $CR^2 = CQ^3$ alors les trois racines sont réelles et deux d'entre elles sont égales :
  - Si $R < 0$:
    $$x_0 = -2\sqrt{Q} - \frac{b}{3} \; x_1 = \sqrt{Q} - \frac{b}{3}$$
  - Sinon :
    $$x_0 = 2\sqrt{Q} - \frac{b}{3} \; x_1 = -\sqrt{Q} - \frac{b}{3}$$
- Sinon :
  - Si $CR^2 < CQ^3$ alors les trois racines sont réelles et distinctes :

$$x_0 = norm * \cos\left(\frac{\theta}{3}\right) - \frac{b}{3}$$
$$x_1 = norm * \cos\left(\frac{\theta+2\pi}{3}\right) - \frac{b}{3}$$
$$x_2 = norm * \cos\left(\frac{\theta-2\pi}{3}\right) - \frac{b}{3}$$

$$\text{Avec } norm = -2 * \sqrt{Q} \qquad \text{et} \qquad \theta = \arccos\left(\frac{R}{(\sqrt{Q})^3}\right)$$

  - Sinon il n'y a qu'une racine réelle (les deux autres étant complexes, elles ne seront pas calculées)
    $$x_0 = A + B - \frac{b}{3}$$

$$\text{Avec } A = \frac{R}{|R|}\sqrt[3]{|R| + \sqrt{R^2 - Q^3}} \qquad \text{et} \qquad B = \frac{Q}{A}$$

Les calculs ci-dessus peuvent paraître dangereux (plusieurs racines carrés sur Q ayant une définition ne garantissant pas sa non-négativité) mais ils sont au contraire plutôt sûrs. En effet, si les conditions préalables aux calculs ayant des racines carrés sont examinées, il est clair que ces derniers ne peuvent pas rentrer dans un cas problématique où les résultats seraient complexes. Par exemple, dans le cadre du calcul de $x_0 = -2\sqrt{Q} - \frac{b}{3}$, Q ne peut pas être négatif car la condition préalable est $CR^2 = CQ^3$, avec $CR^2 \geq 0$, il n'y a donc aucun risque d'être confronté à un Q négatif (s'il l'était, alors CQ^3 le serait aussi, ce qui est impossible au vu de la condition).
Le vrai risque dans ces calculs viennent de la précision numérique. Les variables introduites sont utiles mais ont leur limite. Pour illustrer ceci, l'exemple suivant est

détaillé :

Exemple 2.1.1 :

Avec les coefficients pour l'équation quartique initiale suivants :

$a = 1.91855980781838278259e - 007$

$b = 6.292424211278557779 00e - 009$

$c = 9.592867346946150064 47e - 001$

$d = -9.595349584706126810 92e - 003$

$e = -3.893149781506508335 05e - 007$

Ceux calculés pour la cubique sont :

$a = 1.000000000000000000 00e + 000$

$b = -5.000035603713763724 04e + 006$

$c = -1.632201410990632976 07e + 003$

$d = -2.541913957764555642 85e + 009$

Ce qui entraîne les valeurs calculées suivantes :

$q = 2.500035604330186590 77e + 013$

$r = -2.500053407371746452 64e + 020$

$R = -4.629728532169900838 00e + 018$

$Q = 2.777817338144651459 22e + 012$

$R2 = 2.143438628158806453 91e + 037$

$Q3 = 2.143438626981968582 82e + 037$

$$CR2 = 4.55644467194937704293e + 043$$
$$CQ3 = 4.55644466944769897965e + 043$$

Les tests comme celui de $CR^2 = CQ^3$ se retrouvent complètement faussés par l'instabilité numérique. Il est donc capital de pallier à ce problème sous peine d'avoir un résolveur bancal, incapable de trouver certaines solutions dans certains cas (ou de trouver des solutions trop éloignées). Un solution simple existe : une approximation simplifiant l'équation quartique en une cubique ou une quadratique. En effet les problèmes cités ci-dessus n'interviennent que quand les coefficients a et/ou b de l'équation quartique sont petits. Ceci s'explique par le fait qu'il faille normaliser les équation selon ces derniers dans les processus de résolution. Ainsi, une correction simple apportant une bonne robustesse consiste à effectuer les simples changements suivant dans l'appel au résolveur :

- Si a et b sont petits (en dessous de l'ordre de 10^{-7}) : résoudre la quadratique $cx^2 + dx + e = 0$
- Sinon :
  - Si a est petit : résoudre la cubique $bx^3 + cx^2 + dx + e = 0$
  - Sinon : résoudre la quartique $ax^4 + bx^3 + cx^2 + dx + e = 0$

Si on reprend l'exemple 2.1.1 précédent, les racines calculées à partir de l'équation quartique sont (résultats Wolfram Alpha [36]) :

$$x_0 = -0.000040410043089152613284$$
$$x_1 = 0.010042997629099171339$$
$$x_2 = -0.02140011593 - 2236.07593871146i$$
$$x_3 = -0.02140011593 + 2236.07593871146i$$

Les racines obtenues avec l'approximation quadratique ($cx^2 + dx + e = 0$) sont :

$$x_0 = -0.00004041004308915257040 9$$
$$x_1 = 0.0100429976299598987365$$

Ces résultats correspondant à des temps mesurés en seconde, la précision des racines de l'approximation quadratique suffira amplement.

Un autre problème peut aussi survenir lorsque le coefficient e, représentant dans notre problème l'écartement entre deux objets (deux billes ou une bille et une bande), est très petit, les deux objets étant donc presque en contact. Il est fréquent qu'une erreur numérique se glisse pour trouver une solution proche de 0, mais négative. Une solution négative, représentant un temps, est par définition rejetée, ce qui pourrait avoir pour conséquence l'interpénétration de deux objets. Pour éviter cela, une technique simple consiste à insérer la solution $t = 0$ lorsque e devient trop petit et qu'aucune solution « petite » n'a été trouvée par le résolveur. La nouvelle solution sera testée par la vérification de la validité de la collision, ce qui permettra de la rejeter si elle n'a pas lieu d'être (dans le cas de la séparation après le choc par exemple).

Enfin, pour s'assurer que le résolveur soit le plus précis possible, une à deux itération de Newton sont ajoutées pour raffiner les solutions.

Pour tester le résolveur, deux tests ont été effectués. Le premier est un test de rapidité, le résolveur atteint-il son objectif de concurrencer les méthodes itératives existantes ? Pour y répondre, $5,000,000$ de quartiques différentes sont générées aléatoirement (avec les coefficients a, b, c, d et e aléatoirement choisis entre 10^{-6} et 10^{6}) et traitées d'une part par la méthode *gsl_complex_solve*, méthode utilisée par la bibliothèque GSL pour résoudre les équations de degré supérieur à 3, et d'autres part par le résolveur créé. Les résultats sont bons mais pas exceptionnels :

gsl_complex_solve : 23,6s
résolveur : 21,67s

L'objectif consistant à gagner un maximum de temps est partiellement atteint. Ceci peut cependant s'expliquer facilement : le fait de ne pas passer par une méthode

itérative force à utiliser beaucoup plus de vérifications sur la validité des calculs tout au long du processus. Il est aussi parfois nécessaire de faire de multiple fois un ou plusieurs calculs à cause des possibles erreurs de cancellation.
Par exemple, lors du calcul de e (et de f, ce dernier dépendant de e), un problème de cancellation peut apparaître et engendrer de gros problèmes dans la détermination des solutions :
Si la solution y de la cubique est proche du coefficient b, il y a une forte probabilité qu'une erreur de cancellation fausse les résultats finaux de la quartique.

Une erreur de cancellation entre ces deux valeurs peut entraîner des perturbations plus ou moins grande pour les coefficients des deux quadratiques. Il est donc possible de se retrouver avec un discriminant positif pour une des deux quadratiques alors que celui devrait être négatif.

Il est clair que les deux solutions associées à ce discriminant n'ont pas lieu d'être et apparaîssent à cause de l'imprécision du calcul numérique (fig. 2.1).
Pour éliminer ces fausses solutions, il existe un moyen simple : utiliser les informations de la valeur de la quartiques et de sa dérivée seconde. En effet, un point ayant une dérivée seconde positive correspond à un minimum local. Si la valeur de la quartique en ce point est négative ou nulle, l'existence des solutions additionnelles est vérifiée. Au contraire, si la valeur de la quartique en ce point est positive, alors la courbe ne franchit donc pas la valeur 0, ce qui prouve que les solutions trouvées sont invalides. De même, pour une dérivée seconde négative, le point correspond à un maximum local : si la valeur de la fonction est positive ou nulle alors les solutions sont valides, sinon inexistantes.

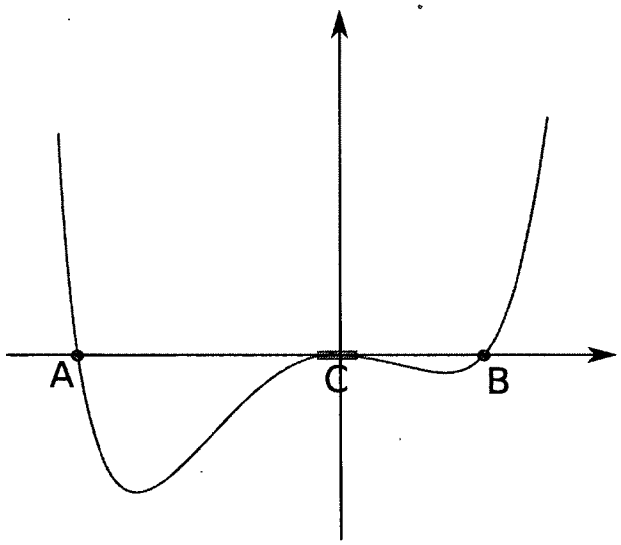

figure 2.1 – Exemple d'une quartique. Les points rouges A et B indiquent les racines réelles et la zone violette C une zone susceptible d'être perçue comme une ou deux solutions par imprécision numérique.

Cette vérification implique des calculs additionnels ralentissant le simulateur qui doit tester toutes ces solutions.
Dans le cas où l'opposé se produirait, un discriminant négatif pour le résolveur alors qu'il devrait être positif, alors il manquerait des solutions. Il est donc nécessaire de trouver un autre moyen de trouver celles-ci. Une technique possible est de choisir une solution pour la cubique qui soit plus éloignée de b. Évidemment, ceci dépend fortement du nombre de solutions trouvées. S'il n'y a qu'une seule solution à la cubique et que celle-ci pose problème, alors il faut trouver une autre astuce. Cette dernière peut être la suivante : lorsqu'il n'y a qu'une seule solution possible pour l'équation cubique (ou que toutes les solutions trouvées sont problématiques), si le discriminant d'une des équations quadratiques est négatif mais petit (généralement $< 10^{-5}$) alors passer le discriminant en valeur absolue. Le test pour les fausses solutions doivent bien sûr être effectué après chaque utilisation de cette astuce.
Toutes ces vérifications et techniques pour contrer les défauts numériques ont un coût qui s'exprime par une augmentation non négligeable du temps de calcul. Ceci explique que le résolveur ne soit pas aussi rapide que ce que laissait espérer les premières at-

tentes.

Le deuxième test se penche sur la précision des solutions trouvées. Sur les 5,000,000 d'équations testées pour la rapidité, la précision moyenne du résolveur est de $5.19141e-16$ alors que celle de *gsl_complex_solve* est de $-3.51643e-15$. Cette moyenne est calculée en injectant les solutions trouvées et en mesurant leur écart par rapport à 0. Le résolveur développé est donc plus précis que *gsl_complex_solve*, mais du même ordre de grandeur. Il serait trop hâtif de considérer le nouveau simulateur comme supérieur car il ne s'agit que d'un jeu de 5,000,000 d'instances reflétant que les deux méthodes ont le même ordre de grandeur en précision.

Il est à noter tout de même que le nouveau résolveur est beaucoup plus efficace que *gsl_complex_solve* dans certaines situations. Ces dernières sont des quartiques ayant un coefficients a initialement très petit ($< 10^{12}$). L'exemple suivant montre l'écart de précision des deux méthodes dans ces cas de figure :

Avec les coefficients de quartiques :

$a = -2.22540721297264076055e-014$
$b = -1.17865479532629251480e+001$
$c = 3.15836559492163360119e+000$
$d = 1.92974322396330535412e+001$
$e = -1.07005063910037279129e-002$

Les résultats sont :

résolveur :

$x_1 = 1.42027646282378182548e+000$
avec une précision : $ans = 0.00000000000000000000e+000$

$x_2 = 5.54453946255442235799e-004$
avec une précision : $ans = 8.47032947254300339068e-022$

gsl_complex_solve :

$z_1 = -1.15286793295829892436e + 000$
avec une précision : $ans = 2.088386018965320001 24e - 005$

$z_2 = 1.420275977679732681 84e + 000$
avec une précision : $ans = 2.088932842176510762 40e - 005$

$z_3 = -5.296355599350633125 00e + 014$
avec une précision : $ans = -2.524753905913421712 54e + 029$

$z_4 = -6.904602050781250000 00e - 003$
avec une précision : $ans = -1.437871464326127279 86e - 001$

Il est évident que *gsl_complex_solve* a un problème avec un coefficient a petit. Ceci s'explique probablement à cause d'une normalisation selon ce coefficient, rendant tous les autres beaucoup trop grand, renforçant les problèmes de précision numérique. Le résolveur développé approximant ces équations par la cubique associée n'a pas ce genre de problèmes.

En conclusion, le résolveur c'est pas très rapide à cause des différentes astuces et vérifications pour empêcher les erreurs numériques de se glisser dans les résultats, il n'est pas non plus d'une précision extrême face à une méthode comme *gsl_complex_solve*, mais il a l'avantage de ne pas avoir de point faible comme peut en avoir *gsl_complex_solve* avec l'exemple cité ci-dessus. L'outil développé rempli donc sa mission, même si les ambitions de départ étaient plus élevées.

2.2 Problèmes dus à l'extension de la simulation au 3D

L'ouverture de l'axe z est un grand pas fait dans le sens de la liberté d'action pour les joueurs, mais elle entraîne des problèmes qui n'avaient pas lieux d'être lorsque décoller du tapis était impossible. Comme dit précédemment le simulateur asynchrone

a besoin que ses évènements respectent trois hypothèses. L'ajout de la 3D à la simulation va cependant introduire des cas qui n'y répondront pas, des évènements qui ne respecteront pas ces dernières. Dans un premier temps, ces situations seront introduites afin de comprendre dans quelles mesures elles peuvent intervenir et comment les hypothèses fondamentales posées pour la simulation asynchrone sont violées. Par la suite, le modèle mathématiques pour la résolution de ces phénomènes physiques sera développé puis la création d'un nouvel évènement et l'adaptation du simulateur seront explicités. Enfin, les résultats du simulateur dans différentes situations employant le nouvel évènement seront exposées.

2.2.1 Cas problématiques

La section sur simulation asynchrone expose trois hypothèses que les évènements doivent tous respecter :

- Hypothèse 1 : Les évènements doivent tous être prédictibles
- Hypothèse 2 : Les évènements doivent tous être instantanés
- Hypothèse 3 : Deux évènements ne peuvent être simultanés

Si un évènement ne respecte pas l'une d'entre elles, alors le simulateur ne sera pas en mesure de le détecter et/ou le traiter. Avec l'arrivée de la possibilité de faire voler une bille au dessus du tapis, de nouveaux évènements font leur apparition. Parmi ceux-ci il est possible d'en distinguer une catégorie très problématique qui violent les trois hypothèses à la fois. Pour faciliter l'introduction ce type de problème, l'exemple suivant est détaillé :
Une bille aérienne vient heurter une autre bille sur le tapis. Leur vélocité linéaire et angulaire ne sont pas suffisantes pour garantir une séparation définitive. Pour simplifier, la bille située au dessus de l'autre va retomber sur cette dernière jusqu'à glisser sur un de ses côtés (fig. 2.2 (a)).

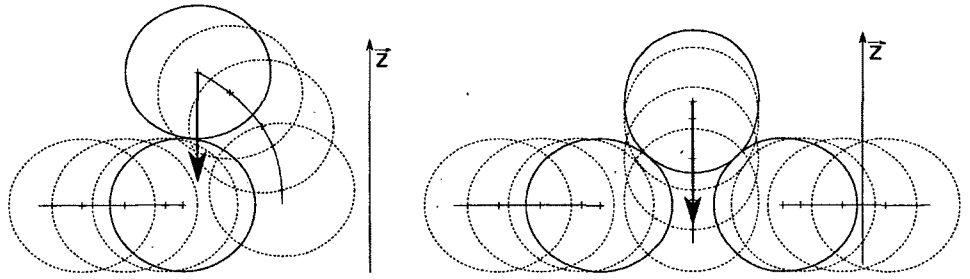

figure 2.2 – (a) Illustration d'une situation physiquement réalisable et contraire à deux hypothèses posées pour la simulation par évènement. (b) Illustration d'une situation physiquement réalisable et contraire à toutes les hypothèses posées pour la simulation par évènement.

Il s'agit donc du glissement d'une bille sur une autre. Le simulateur, à ce stade de développement, traiterait le problème comme une succession sans fin de collision entre billes. Il est en effet trivial de remarquer que ce type de cas ne rentre pas dans les hypothèses posées pour le bon fonctionnement du simulateur : il s'agit pour cet exemple d'un évènement non prévisible (analytiquement) et non instantané. Si la bille au sol était remplacée par deux, l'évènement impliquerait de plus un aspect simultané (fig. 2.2 (b)). Il est donc clair qu'un simulateur pûrement asynchrone tel que décrit jusqu'à maintenant ne peut pas être en mesure de régler ce type de problème. Ces cas problématiques ne sont malheureusement pas rares (fig. 2.3), en particulier avec une casse aérienne (comprendre que la bille blanche va arriver sur le triangle de billes par les airs) il est facile d'aboutir à ce genre de cas de figure. Il est donc indispensable de trouver un moyen de les traiter pour bénéficier de la troisième dimension sans risque de phénomènes anormaux pour le simulateur.

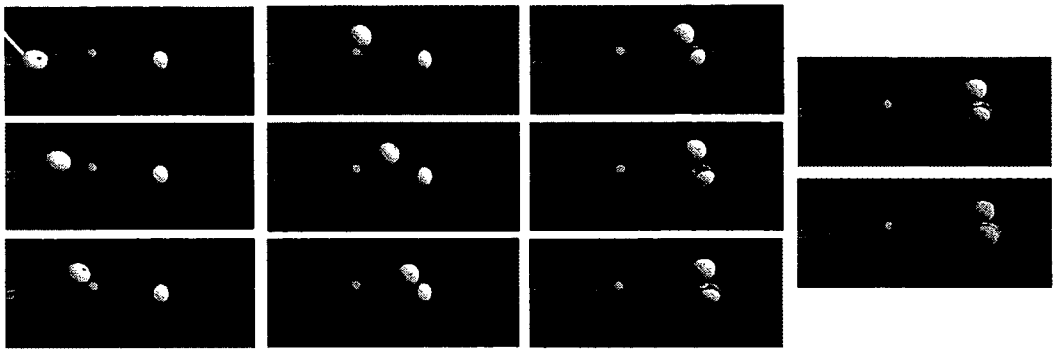

figure 2.3 – Exemple de cas problématique (une succession de collision se produit au point surligné lorsque les billes sont en positions "rouge"). En résulte deux possibilités : blocage du simulateur avec collisions infinies ou interpénétration des billes (une erreur numérique infime pour la détection de collision en étant la cause))

2.2.2 Modèle mathématique de la situation

De nombreux travaux ont été effectués pour résoudre les problèmes de simulation de la dynamique de corps rigides avec friction, dont Mihai Anitescu [4][5][3], David Baraff [7], Jeff Trinkle [35], David Stewart [32][33] or Tobias Preclik [27]. Ces problèmes étant important pour beaucoup d'autres problèmes que la simulation de billard. Plusieurs méthodes existent pour résoudre ces problèmes (deux sont présentées dans [17]). Les principaux travaux utilisés sont [4] et [27], utilisant une méthode efficace développée par la suite.

Chaque objet en contact avec un autre est soumis à un cône de friction, représentant à la fois la force normale $\vec{f_n}$ que subissent les deux objets, perpendiculaire au plan de contact évitant ainsi toute interpénétration, et la force de friction $\vec{f_t}$, tangente au plan de contact, effet du frottement entre les deux objets. Avec $\vec{n_j}$ le vecteur unitaire représentant la direction normale au plan de contact et $(\vec{t_{x_j}}, \vec{t_{y_j}})$ deux vecteurs unitaires orthogonaux appartenant au plan de contact, il est possible d'exprimer les forces de contact comme :

$$\vec{f_j} = c_{n_j}\vec{n_j} + c_{t_{x_j}}\vec{t_{x_j}} + c_{t_{y_j}}\vec{t_{y_j}},$$

où c_{n_j}, $c_{t_{x_j}}$ et $c_{t_{y_j}}$ sont les coefficients associés aux trois vecteurs pour définir la force de contact.

Le cône de friction s'exprime alors comme :

$$\mathcal{F} = \{c_{n_j}\vec{n_j} + c_{t_{x_j}}\vec{t_{x_j}} + c_{t_{y_j}}\vec{t_{y_j}} \quad | \quad c_{n_j} \geq 0,\ \sqrt{c_{t_{x_j}}^2 + c_{t_{y_j}}^2} \leq \mu_j c_{n_j}\}$$

La non-linéarité de cette définition implique par la suite un problème de complémentarité non-linéaire. Ce problème est développé ainsi :

Il est considéré par la suite que N représente le nombre d'objets impliqués et K le nombre de contacts total entre ces derniers. Pour pouvoir traiter le système d'objet en mouvement dans son ensemble, il est nécessaire de regrouper tous les vecteurs normaux et tangentiels dans deux matrices $\mathbf{N}$ et $\mathbf{D}$ telles que :

$$\mathbf{N} = \begin{pmatrix} \vec{n_1} & & \vec{0} \\ & \ddots & \\ \vec{0} & & \vec{n_K} \end{pmatrix} \in \mathbb{R}^{3K \times K}$$

$$\mathbf{D} = \begin{pmatrix} \mathbf{D_1} & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \mathbf{D_K} \end{pmatrix} \in \mathbb{R}^{3K \times 2K}$$

avec la matrice $\mathbf{D}_j$ s'exprimant comme : $\mathbf{D}_j = \left(\vec{t_{xj}}, \vec{t_{yj}}\right) \in \mathbb{R}^{3\times 2}$. Les coefficients associés à ces vecteurs seront stockés dans deux vecteurs $\vec{f_n}$ et $\vec{\beta}$ tels que :

$$\vec{f_n} = (c_{n_1}, ..., c_{n_K})^T \in \mathbb{R}^K$$
$$\vec{\beta} = \left(\vec{\beta_1}^T, ..., \vec{\beta_K}^T\right)^T \in \mathbb{R}^{2K}$$

avec $\vec{\beta_j} = (c_{txj}, c_{tyj}) \in \mathbb{R}^2$.

Maintenant, les forces de contact peuvent toutes être exprimées avec :

$$\vec{f} = \begin{pmatrix} c_{n_1}\vec{n_1} + \mathbf{D_1}\vec{\beta_1} \\ \vdots \\ c_{n_K}\vec{n_K} + \mathbf{D_K}\vec{\beta_K} \end{pmatrix}$$

Pour agir sur les vélocités linéaire et angulaire des billes, il est nécessaire d'exprimer ces forces au centre de gravité des objets. Pour cela, l'opérateur $\mathbf{J}$ est constitué tel que :

$$\mathbf{J} = \begin{pmatrix} \mathbf{J_{11}} & \dots & \mathbf{J_{1K}} \\ \vdots & \ddots & \vdots \\ \mathbf{J_{N1}} & \dots & \mathbf{J_{NK}} \end{pmatrix}$$

$$\text{avec : } \mathbf{J_{ij}} = \begin{cases} \begin{pmatrix} \mathbf{E} \\ \mathbf{r_{ij}^{\times}} \end{pmatrix} & si \quad i = obj_1 \; du \; contact \; j \\ -\begin{pmatrix} \mathbf{E} \\ \mathbf{r_{ij}^{\times}} \end{pmatrix} & si \quad i = obj_2 \; du \; contact \; j \\ \mathbf{0} & sinon \end{cases} \in \mathbb{R}^{6\times 3},$$

avec $\mathbf{E}$ la matrice identité de taille 3.

La matrice $\mathbf{r_{ij}^{\times}}$ représente la «forme opérationnelle» de $\vec{r_{ij}}\times$. Le vecteur $\vec{r_{ij}}$ relie le centre de l'objet i au point de contact j. L'opération $\vec{r_{ij}}\times$ ($\times$ ici étant le produit vectoriel) est indispensable pour obtenir les variations de la vélocité angulaire des objets. Avec $\vec{r_{ij}} = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \end{pmatrix}$, on peut déduire $\mathbf{r_{ij}^{\times}}$:

$$\begin{pmatrix} r_1 \\ r_2 \\ r_3 \end{pmatrix} \times \vec{a} = \begin{pmatrix} 0 & -r_3 & r_2 \\ r_3 & 0 & -r_1 \\ -r_2 & r_1 & 0 \end{pmatrix} \vec{a} = \mathbf{r_{ij}^{\times}}\vec{a}$$

Il est maintenant possible d'établir l'équation de mouvement :

$$\mathbf{M}\vec{v} = \mathbf{J}\vec{f} + \vec{f}_{ext},$$

$$\text{avec :} \begin{cases} \mathbf{M} = \begin{pmatrix} m_1\mathbf{E} & & & \mathbf{0} \\ & \mathbf{I_1} & & \\ & & \ddots & \\ & & & m_N\mathbf{E} \\ \mathbf{0} & & & \mathbf{I_N} \end{pmatrix} \in \mathbb{R}^{6N\times 6N} \\ \vec{f_{ext}} = \begin{pmatrix} \vec{g} \\ \mathbf{I_1}\vec{\omega_1} \times \vec{\omega_1} \\ \vdots \\ \vec{g} \\ \mathbf{I_N}\vec{\omega_N} \times \vec{\omega_N} \end{pmatrix} \in \mathbb{R}^{6N} \end{cases}$$

Il ne reste qu'à formuler le problème de complémentarité pour pouvoir déterminer les forces de contact. Pour simplifier la méthode, le cas sans friction sera d'abord détaillé (de la même façon que [27]).

Le vecteur $\vec{f_n}$ est composé des coefficients c_n représentants les forces normales appliqués aux objets. Il est clair que pour éviter toute interpénétration, tous les c_n seront non-négatifs. Ceci implique $\vec{f_n} \geq \vec{0}$. Le rapprochement avec la vélocité relative au point de contact entre deux objets est aussi possible. Celle-ci se calcule ainsi : avec deux objets A et B :

$$\vec{v_{rel}} = \vec{v_A} + \vec{\omega_A} \times \vec{r_{Aj}} - \vec{v_B} - \vec{\omega_B} \times \vec{r_{Bj}}$$

Sa composante normale sera l'élément important ici. Elle se détermine comme :

$$\vec{v_{rel}}.n_j = \vec{n_j}.(\vec{v_A} + \vec{\omega_A} \times \vec{r_{Aj}} - \vec{v_B} - \vec{\omega_B} \times \vec{r_{Bj}})$$

ou :

$$\vec{v_{rel}}.n_j = \vec{n_j}.\left(\mathbf{E}\vec{v_A} - \mathbf{r_{Aj}^{\times}}\vec{\omega_A} - \mathbf{E}\vec{v_B} + \mathbf{r_{Bj}^{\times}}\vec{\omega_B}\right)$$

Il est possible de reconnaître que cette composante peut être facilement déterminée grâce à l'opérateur $\mathbf{J}^T$:

$$\mathbf{J}^T = \begin{pmatrix} \mathbf{0} & \mathbf{E}^T & \mathbf{r}_{\mathbf{Aj}}^{\times\,T} & \mathbf{0} & \cdots & \mathbf{0} & -\mathbf{E}^T & -\mathbf{r}_{\mathbf{Bj}}^{\times\,T} & \mathbf{0} \end{pmatrix}$$
$$= \begin{pmatrix} \mathbf{0} & \mathbf{E} & -\mathbf{r}_{\mathbf{Aj}}^{\times} & \mathbf{0} & \cdots & \mathbf{0} & -\mathbf{E} & \mathbf{r}_{\mathbf{Bj}}^{\times} & \mathbf{0} \end{pmatrix}$$

Si la composante de la vélocité relative sur le vecteur normal du contact est positive, alors les objets vont se séparer et la force de contact doit être nulle. Cependant, la présence d'une force de contact ne vas pas forcément briser un contact. Ceci amène la première condition de complémentarité :

$$\vec{0} \leq \mathbf{N}^T\mathbf{J}^T\vec{v}^{t+\Delta t} \quad \perp \quad \vec{f_n} \geq 0$$

En utilisant une discrétisation de $\vec{v}$: $\vec{v}^{t+\Delta t} = \vec{v}^t + \Delta t\mathbf{M}^{-1}\left(\mathbf{J}\mathbf{N}\vec{f_n} + \vec{f_{ext}}\right)$, il est possible de trouver de ramener le problème sans friction à :

$$\vec{0} \leq \left(\mathbf{N}^T\mathbf{J}^T\mathbf{M}^{-1}\mathbf{J}\mathbf{N}\right)\Delta t\vec{f_n} + \mathbf{N}^T\mathbf{J}^T\left(\vec{v}^t + \Delta t\mathbf{M}^{-1}\vec{f_{ext}}\right) \quad \perp \quad \Delta t\vec{f_n} \geq 0$$

En rajoutant la friction, la discrétisation de $\vec{v}$ donnera :

$$\vec{v}^{t+\Delta t} = \vec{v}^t + \Delta t\mathbf{M}^{-1}\left(\mathbf{J}\left(\mathbf{N}\vec{f_n} + \mathbf{D}\vec{\beta}\right) + \vec{f_{ext}}\right)$$

Pour compléter le NCP, il faut deux contraintes supplémentaires. La première est reliée au fait que si la vélocité relative au point de contact est non-nulle, la force de friction s'oppose autant que possible à son mouvement. Si au contraire la vélocité relative est nulle, cela n'implique pas qu'il n'y ai pas de friction. Finalement, pour se situer dans le cône de friction, si la vélocité relative est non-nulle, la force de friction se situera sur la limite haute du cône, sinon elle sera comprise à l'intérieur. La contrainte peut s'exprimer mathématiquement comme :

$$0 \leq \mu_j c_{nj} - \|\vec{\beta_j}\| \qquad \perp \qquad \lambda_j \geq 0 \qquad \forall j \in \{1,...,K\},$$

avec $\vec{\lambda}$ un vecteur auxiliaire. [27] ne spécifie pas sa nature mais il s'agit en fait de la magnitude de la vélocité relative au point de contact [4].

La deuxième contrainte concerne la différence de type de friction. Il y a en effet la friction dynamique et la friction statique. La friction statique intervient lorsque la vélocité relative d'un objet est nulle. On est donc en situation de frottement statique lorsqu'une bille est à l'arrêt ou est en roulement. A l'inverse, la friction dynamique entre en action si la vélocité relative est non-nulle. Il y a donc deux possibilités pour le prochain pas de temps : soit la vélocité relative est nulle (friction statique) soit elle ne l'est pas (friction dynamique). Soit $\vec{v_j} \in \mathbb{R}^2$ les vélocités projetées (celles obtenues au prochain pas de temps) au contact j.
Il est donc possible d'exprimer cette deuxième contrainte comme :

$$\vec{0} \leq \lambda_j \vec{e} + \vec{v_j} \qquad \perp \qquad s_j \geq 0 \qquad \forall j \in \{1,...,K\}$$

avec $\vec{e}$ le vecteur de 1 de taille appropriée (le nombre de contacts) et s_j tel que : $\mu_j c_{nj} - \|\vec{\beta_j}\| + s_j = 0$.

Il ne reste plus qu'à spécifier deux autres matrices représentant les coefficients de frottement, un opérateur et les vecteurs de vélocités projetées pour pouvoir exprimer pleinement le NCP :

$$\mu = \begin{pmatrix} \mu_1 & & 0 \\ & \ddots & \\ 0 & & \mu_K \end{pmatrix} \in \mathbb{R}^{K \times K}$$

$$\mathbf{E_2} = \begin{pmatrix} \vec{e} & & \vec{0} \\ & \ddots & \\ \vec{0} & & \vec{e} \end{pmatrix} \in \mathbb{R}^{2K \times K}$$

$$\begin{pmatrix} \vec{v_1} \\ \vdots \\ \vec{v_K} \end{pmatrix} = \mathbf{D}^T \mathbf{J}^T \vec{v}^{t+\Delta t} \quad \in \mathbb{R}^{2K}$$

Le problème est donc :

$$\vec{v}^{t+\Delta t} = \vec{v}^{t} + \Delta t \mathbf{M}^{-1}\left(\mathbf{J}\left(\mathbf{N}\vec{f_n} + \mathbf{D}\vec{\beta}\right) + \vec{f_{ext}}\right)$$

avec les conditions de complémentarité :

$$\begin{array}{rcl} \vec{0} \leq \mathbf{N}^T\mathbf{J}^T\vec{v}^{t+\Delta t} & \perp & \vec{f_n} \geq \vec{0} \\ \vec{0} \leq \mu\vec{f_n} - \vec{\mathbf{B}} & \perp & \vec{\lambda} \geq \vec{0} \\ \vec{0} \leq \mathbf{E}_2\vec{\lambda} + \mathbf{D}^T\mathbf{J}^T\vec{v}^{t+\Delta t} & \perp & \vec{s} \geq \vec{0} \end{array}$$

avec $\vec{\mathbf{B}} = \left(\|\vec{\beta_1}\|, \|\vec{\beta_2}\|, \ldots, \|\vec{\beta_K}\|\right)$.

Ces problèmes sont assez complexes à résoudre mais surtout : peu peuvent garantir l'existence d'une solution en tout point. Dans le cadre d'une simulation rapide et sûre, il n'est pas possible d'emprunter cette voie, il faut donc trouver un moyen d'arriver à un problème moins complexe.
Il existe une astuce pour arriver à un problème de complémentarité linéaire : il faut considérer le cône de friction avec une approximation polyédrique. Celle-ci consiste à ne plus considérer le cône ayant une coupe représentant un cercle parfait mais comme un polygône plus ou moins dôté de côtés pour une approximation plus ou moins fidèle (voir figure 1 [27]).

Les deux vecteurs $(\vec{t_{x_j}}, \vec{t_{y_j}})$ définissant la coupe du cône seront remplacés par un ensemble de vecteurs $\vec{d_{k_j}}$ ($k \in \{1, ..., \eta\}$, η représentant le nombre de vecteur et donc la complexité de l'approximation). Ces derniers s'exprimeront comme suit :

$$\vec{d_{k_j}} = \cos\left(\frac{k-1}{\eta}2\pi\right)\vec{t_{x_j}} + \sin\left(\frac{k-1}{\eta}2\pi\right)\vec{t_{y_j}}$$

Les vecteurs $\vec{d_{k_j}}$ avec $k \in \{1, ..., \eta\}$ seront stockés dans une matrice :

$$\mathbf{D_j} = \left(\vec{d_{1_j}}, ..., \vec{d_{\eta_j}}\right) \in \mathbb{R}^{3\times\eta}$$

Les coefficients $c_{t_{x_j}}$ et $c_{t_{y_j}}$ seront remplacés par β_{k_j} compris dans un vecteur :

$$\vec{\beta_j} = \left(\beta_{1_j}, ..., \beta_{\eta_j}\right)^T \in \mathbb{R}^{\eta}$$

Le cône de friction est maintenant de la forme :

$$\hat{\mathcal{F}}_j = \{c_{n_j}\vec{n_j} + \mathbf{D_j}\vec{\beta_j} \quad | \quad c_{n_j} \geq 0,\ \vec{\beta_j} \geq \vec{0},\ \vec{e}^T\vec{\beta_j} \leq \mu_j c_{n_j}\},$$

avec $\vec{e}$ le vecteur colonne $(1, ..., 1)^T \in \mathbb{R}^\eta$.

Les vecteurs normaux et tangentiels seront de même stockés dans deux matrices $\mathbf{N}$ et $\mathbf{D}$ telles que :

$$\mathbf{N} = \begin{pmatrix} \vec{n_1} & & \vec{0} \\ & \ddots & \\ \vec{0} & & \vec{n_K} \end{pmatrix} \in \mathbb{R}^{3K \times K}$$

$$\mathbf{D} = \begin{pmatrix} \mathbf{D_1} & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \mathbf{D_K} \end{pmatrix} \in \mathbb{R}^{3K \times \eta K}$$

Il est à noter que si la matrice $\mathbf{D_j}$ contient un vecteur $\vec{d}$, elle contient de même son opposé, le vecteur $-\vec{d}$. Les coefficients associés à ces vecteurs sont aussi stockés dans deux vecteurs $\vec{f_n}$ et $\vec{\beta}$ tels que :

$$\vec{f_n} = (c_{n_1}, ..., c_{n_K})^T \in \mathbb{R}^K$$
$$\vec{\beta} = \left(\vec{\beta_1}^T, ..., \vec{\beta_K}^T\right)^T \in \mathbb{R}^{\eta K}$$

Les forces de contact, la matrice $\mathbf{J}$ et l'équation de mouvement sont inchangées.

Il ne reste qu'à changer la formulation du problème de complémentarité pour qu'il devienne linéaire. Un problème de complémentarité linéaire correspond à déterminer un vecteur solution $\vec{x}$ en respectant des conditions précises :

$$\vec{x} \geq \vec{0} \qquad \mathbf{A}\vec{x} + \vec{b} \geq \vec{0} \qquad \vec{x}^T\left(\mathbf{A}\vec{x} + \vec{b}\right) = 0$$

Il faut donc trouver les expressions de $\mathbf{A}$, $\vec{b}$ et $\vec{x}$ pour résoudre le problème.

La première condition de complémentarité reste la même, elle ne concerne pas la friction mais seulement les composantes normales. La deuxième est adaptée à la nouvelle formulation du cône de friction avec l'approximation polyédrique :

$$0 \leq \mu_j c_{n_j} - \vec{e}^T \vec{\beta}_j \quad \perp \quad \lambda_j \geq 0 \qquad \forall j \in \{1, ..., K\}.$$

La troisième condition de complémentarité reprend la théorie développée pour le NCP précédent en rajoutant un aspect : le fait que la matrice $\mathbf{D_j}$ soit composée de paires de vecteurs antipodaux, la moitié des composantes de $\underline{\vec{v}_j}$ sera non-négative, l'autre non-positive. Il est possible d'avancer le fait que $\vec{\beta}_j$ aura une composante non-négative dans la direction opposée la plus proche de la vélocité relative (pour maximiser la dissipation de puissance) alors que ses autres composantes devraient disparaître [27]. C'est dans cette direction que le minium de vélocité projetée sera observé [27]. La troisième condition s'exprime donc comme :

$$\vec{0} \leq \lambda_j \vec{e} + \underline{\vec{v}_j} \quad \perp \quad \vec{\beta}_j \geq \vec{0} \qquad \forall j \in \{1, ..., K\}$$

Il ne reste plus qu'à exprimer les matrices pour pouvoir exprimer pleinement le LCP :

$$\mathbf{E}_\eta = \begin{pmatrix} \vec{e} & & \vec{0} \\ & \ddots & \\ \vec{0} & & \vec{e} \end{pmatrix} \in \mathbb{R}^{\eta K \times K}$$

$$\begin{pmatrix} \underline{\vec{v}_1} \\ \vdots \\ \underline{\vec{v}_K} \end{pmatrix} = \mathbf{D}^T \mathbf{J}^T \vec{v}^{t+\Delta t} \quad \in \mathbb{R}^{\eta K}$$

Les conditions de complémentarité sont :

$$\begin{aligned} \vec{0} \leq \mathbf{N}^T \mathbf{J}^T \vec{v}^{t+\Delta t} \quad &\perp \quad \vec{f}_n \geq \vec{0} \\ \vec{0} \leq \mu \vec{f}_n - \mathbf{E}_\eta^T \vec{\beta} \quad &\perp \quad \vec{\lambda} \geq \vec{0} \\ \vec{0} \leq \mathbf{E}_\eta \vec{\lambda} + \mathbf{D}^T \mathbf{J}^T \vec{v}^{t+\Delta t} \quad &\perp \quad \vec{\beta} \geq \vec{0} \end{aligned}$$

Le LCP final dans sa forme standard est donc :

$$\begin{pmatrix} \mathbf{D}^T \mathbf{J}^T \mathbf{M}^{-1} \mathbf{J} \mathbf{D} & \mathbf{D}^T \mathbf{J}^T \mathbf{M}^{-1} \mathbf{J} \mathbf{N} & \mathbf{E}_\eta \\ \mathbf{N}^T \mathbf{J}^T \mathbf{M}^{-1} \mathbf{J} \mathbf{D} & \mathbf{N}^T \mathbf{J}^T \mathbf{M}^{-1} \mathbf{J} \mathbf{N} & \mathbf{0} \\ -\mathbf{E}_\eta^T & \mu & \mathbf{0} \end{pmatrix} \cdot \begin{pmatrix} \Delta t \vec{\beta} \\ \Delta t \vec{f}_n \\ \vec{\lambda} \end{pmatrix} + \begin{pmatrix} \mathbf{D}^T \mathbf{J}^T \left(\vec{v}^t + \Delta t \mathbf{M}^{-1} \vec{f_{ext}} \right) \\ \mathbf{N}^T \mathbf{J}^T \left(\vec{v}^t + \Delta t \mathbf{M}^{-1} \vec{f_{ext}} \right) \\ \vec{0} \end{pmatrix} \geq \vec{0}$$

$$\perp$$

$$\begin{pmatrix} \Delta t\vec{\beta} \\ \Delta t\vec{f_n} \\ \vec{\lambda} \end{pmatrix} \geq \vec{0}$$

La validité du LCP est justifiée par [27] et [4] dans [27], section 2.3.4, elle ne sera pas étudiée ici. Il sera considéré comme acquis le fait que le problème ait toujours une solution.

2.2.3 Résolution

Un LCP se présente sous la forme $\mathbf{A}\vec{x} + \vec{b} = \vec{w}$, avec :

$$\vec{w} \geq \vec{0}$$
$$\vec{x} \geq \vec{0}$$
$$\vec{x}^T\vec{w} = \vec{0}$$

Ces problèmes font partie de cas particuliers de la famille des problèmes généraux de complémentarité :

$$F(z) \geq 0$$
$$z \geq 0$$
$$z^T F(z) \geq 0$$

Pour expliquer la résolution d'un LCP, il est commode de faire un rapprochement avec la programmation quadratique. Les problèmes de cette dernière sont de la forme :

$$\text{Min } \tfrac{1}{2}x^T Dx + c^T x$$
$$\text{Sujet à } Ax \geq b$$
$$x \geq 0$$

Avec D une matrice symétrique $m \times m$, A de taille $n \times n$, c et x des vecteurs $\in \mathbb{R}^m$ et b un vecteur de taille n. La première condition de ce problème peut très bien s'exprimer comme : $b - Ax \leq 0$

Les conditions de Karush-Kuhn-Tucker (KKT) pour ce problème s'expriment comme :

$$\begin{gathered} c + Dx - A^T\lambda \geq 0, \\ -b + Ax \geq 0, \\ \lambda^T(-b + Ax) = 0, \\ x^T\left(c + Dx - A^T\lambda\right) = 0, \\ x \geq 0,\ \lambda \geq 0, \end{gathered}$$

où $\lambda = [\lambda_1, ..., \lambda_m]^T$ est le vecteur de multiplicateur de Lagrange approprié.
En prenant :

$$\begin{aligned} \mu &= c + Dx - A^T\lambda \\ \nu &= -b + Ax \end{aligned}$$

Il est possible de réécrire les conditions de KKT :

$$\begin{gathered} \begin{bmatrix} \mu \\ \nu \end{bmatrix} = \begin{bmatrix} c \\ -b \end{bmatrix} + \begin{bmatrix} D & -A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} x \\ \lambda \end{bmatrix} \\ \lambda^T\nu = 0 \\ x^T\mu = 0 \\ x, \lambda, \mu, \nu \geq 0 \end{gathered}$$

En utilisant $w = \begin{bmatrix} \mu \\ \nu \end{bmatrix}$, $z = \begin{bmatrix} x \\ \lambda \end{bmatrix}$, $q = \begin{bmatrix} c \\ -b \end{bmatrix}$, $M = \begin{bmatrix} D & -A^T \\ A & 0 \end{bmatrix}$, alors les conditions KKT pour le LCP sont de la forme :

$$\begin{gathered} w = q + Mz \\ z^T w = 0 \\ w, z \geq 0 \end{gathered}$$

Il est possible de retrouver un lien direct avec le problème de complémentarité linéaire initial :

$$\begin{array}{ll} Mz+q \geq 0 & [I-M]\begin{bmatrix} w \\ z \end{bmatrix} \\ z \geq 0 & w,z \geq 0 \\ z^T(Mz+q)=0 & z^T w = 0 \end{array}$$

Pour résoudre ce genre de problème, plusieurs méthodes sont disponibles. Parmi elles, la méthode de Lemke est une technique, utilisant un pivôt, qui, après un nombre plus ou moins grand d'itérations, trouvera une solution acceptable pour le problème. Cette technique ne sera pas explicitée ici, de nombreux ouvrages et articles y faisant référence [11].

Le problème traité sera de la forme : $\mathbf{A}\vec{x} + \vec{b} = \vec{w}$, avec :

$$\begin{array}{c} \vec{w} \geq \vec{0} \\ \vec{x} \geq \vec{0} \\ \vec{x}^T\vec{w} = \vec{0}, \end{array}$$

et :

$$A = \begin{pmatrix} \mathbf{D}^T\mathbf{J}^T\mathbf{M}^{-1}\mathbf{J}\mathbf{D} & \mathbf{D}^T\mathbf{J}^T\mathbf{M}^{-1}\mathbf{J}\mathbf{N} & \mathbf{E}_\eta \\ \mathbf{N}^T\mathbf{J}^T\mathbf{M}^{-1}\mathbf{J}\mathbf{D} & \mathbf{N}^T\mathbf{J}^T\mathbf{M}^{-1}\mathbf{J}\mathbf{N} & \mathbf{0} \\ -\mathbf{E}_\eta^T & \mu & \mathbf{0} \end{pmatrix}$$

$$x = \begin{pmatrix} \Delta t\vec{\beta} \\ \Delta t\vec{f_n} \\ \vec{\lambda} \end{pmatrix}$$

$$b = \begin{pmatrix} \mathbf{D}^T\mathbf{J}^T\left(\vec{v}^t + \Delta t\mathbf{M}^{-1}\vec{f_{ext}}\right) \\ \mathbf{N}^T\mathbf{J}^T\left(\vec{v}^t + \Delta t\mathbf{M}^{-1}\vec{f_{ext}}\right) \\ \vec{0} \end{pmatrix}$$

La méthode de Lemke sera utilisée par l'intermédiaire du résolveur PATH [12]. L'utilisation de ce dernier permet d'avoir accès à une méthode robuste et optimisée ce qui sera indispensable pour assurer l'efficacité et la précision du simulateur.

La prochaine étape consiste à déterminer le nombre de vecteurs à utiliser dans l'approximation polyédrique pour obtenir des trajectoires de billes similaires à celles

qu'il est possible d'obtenir avec les équations analytique du mouvement. Pour ce faire, le test suivant est utilisé : une bille dessinant une trajectoire courbe avec les équations du régime de mouvement *SLIDING* est confrontée à une bille dirigée par le LCP vu précédemment (durant sa phase de glissement seulement, par la suite la bille reprend un régime normal de *ROLLING*). Plus le nombre de vecteur est augmenté, plus les trajectoires sont semblables, mais la résolution est aussi de plus en plus lourde (fig. 2.4).

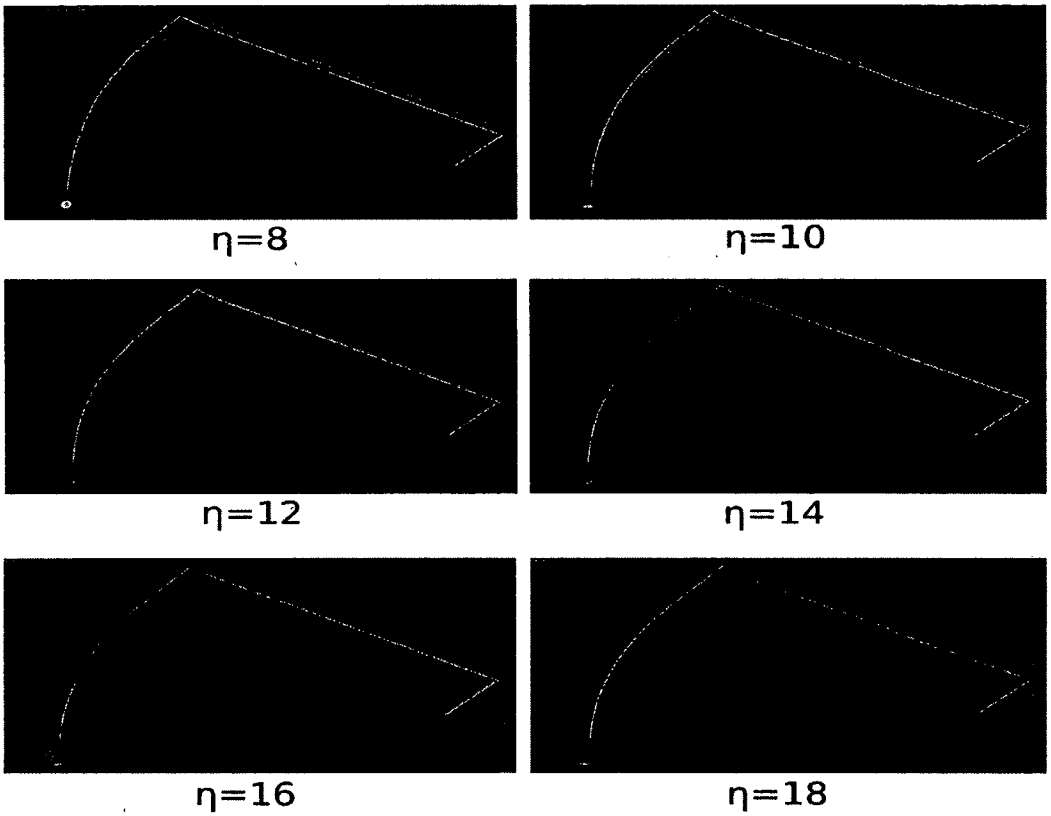

figure 2.4 – Comparaison de trajectoires de billes avec variation de η, le nombre de vecteur pour l'approximation polyédrique. La courbe blanche correspond à la trajectoire d'une bille en *SLIDING*, la bleue utilise le modèle développé.

Il est possible de remarquer une certaine stagnation du résultat à partir de $\eta = 14$, c'est donc le nombre de vecteur qui sera utilisé par la suite à chaque utilisation du

LCP.

Maintenant qu'un outil est à disposition pour régler ces problèmes, il faut adapter le simulateur pour qu'il ait un moyen de savoir quand ce type d'évènement a lieu et quand en sortir (les billes ne se séparent pas instantanément).

2.2.4 Nouvel évènement et changements dans l'algorithme général

Comme dit précédemment, les nouveaux phénomènes physiques à introduire dans le simulateur violent les règles de fonctionnement de la simulation synchrone. Il est donc capital de trouver un moyen soit d'exprimer ces évènements en plusieurs "étapes évènementielles" qui elles répondent aux hypothèses, soit de créer une simulation hybride entre synchrone et asynchrone. La solution choisie est en fait un mélange des deux.

Dans un premier temps, un nouvel évènement caractérisant le glissement d'une bille sur une autre est défini : le *LINK*. Le nom fait référence au fait qu'il s'agit de billes liés entre elles par un point de contact.

Ce nouvel évènement ne sera pas détecté d'une façon analytique comme les autres évènements, il se déclenchera après observation du système présent et passé. En effet, le phénomène est traité jusqu'à maintenant comme une succession de collision par la simulateur, cette information va donc être mise à contribution : l'évènement de *LINK* ne se déclenchera qu'après un nombre défini de collisions entre deux même billes. Le fait que les deux billes doivent être les mêmes vient du fait qu'il peut se passer beaucoup de collisions en très peu de temps sans pour autant être en présence du cas problématique traité par l'évènement. La casse est un exemple flagrant de succession de collisions n'engendrant pas forcément de problèmes.
L'évènement *LINK* ne sera donc pas déterminé par un temps comme les autres mais par l'observation d'évènements passés !

Une fois détecté, comment le simulateur doit traiter ce dernier alors qu'il n'est pas censé être instantané ? En effet il va falloir trouver un moyen d'effectuer d'autres évènements alors que le *LINK* est en cours. Il est parfaitement possible que deux

billes rentrent en collision ou changent de régime de mouvement pendant le glissement d'une bille sur une autre !
Pour pallier à ce problème, il est confortable de décomposer le *LINK* en plusieurs sous-évènements appelés *LINKING*. Ces derniers auront le rôle de mise à jour de la table selon un pas de temps défini. Pour faire simple, les vélocités linéaire et angulaire des billes impliquées dans le *LINK* seront calculées avec un pas de temps fixé, grâce au LCP vu précédemment, et par la suite toutes les billes avanceront de ce pas de temps. Afin d'éviter de sauter un évènement, le pas de temps sera variable en fonction des évènements calculés à venir. Il est aussi indispensable d'utiliser un pas de temps de base très court afin de ne pas tomber dans le piège de précision qu'est la simulation synchrone.

Il est à noter que les billes impliquées par le *LINK* ne seront plus dans un régime de mouvement "normal". Ces nouveaux états de mouvement se décomposent en deux nouveaux état de bille :

- *LINKER* : Il s'agit de l'état de la bille déclenchant l'évènement *LINK*. Il ne peut y avoir initialement qu'une seule bille en *LINKER*, mais par la suite, d'autres peuvent le devenir si elles déclenchent un évènement similaire avec d'autres billes (à noter que si cela se produisait, tous les sous-systèmes impliquant un *LINKER* différent se traiteront dans les mêmes évènement de *LINKING*). Les billes définies comme *LINKER* sont les pièces centrales de l'évènement *LINK* : celles-ci définiront quelles sont les billes qui sont impliquées dans cet évènement. Les billes en régime *LINKER* ne reprennent un régime normal que lorsque plus aucune billes ne sont en contact avec celles-ci.
- *LINKED* : Les billes sont *LINKED* si elles sont détectées comme au contact d'une bille en état *LINKER*. Celles-ci subissent les actions des *LINKER* et retrouvent un régime de mouvement normal dès qu'elles quittent ces dernières.

Le point commun entre les billes *LINKER* et *LINKED* est qu'elles n'évoluent plus selon les règles du mouvement vues précédemment. En effet, elles entrent dans une phase de simulation synchrone où leurs vélocités seront recalculées à chaque pas de temps. Leurs positions ne prendront donc en compte que leurs vélocités calculées à chaque pas de temps, en ignorant les effets de l'accélération. Il est fortement conseillé

d'avoir un pas de temps petit afin que les imprécisions ne s'accumulent pas. Les billes en régime *LINKER* ou *LINKED* n'auront qu'un seul évènement possible : le *LINKING*. Les autres billes pourront par contre détecter les collisions avec celles-ci de la même manière que les autres.

L'exemple simple (seules deux billes sont impliquées) suivant illustre le cheminement du *LINK* :

figure 2.5 – Exemple : Etat initial

La bille 11 (rouge rayée) frappée par la queue avec des caractéristiques qui vont la faire voler vers la bille 10 (bleue rayée)(fig. 2.5). Le prochain évènement est donc sa collision avec le tapis.

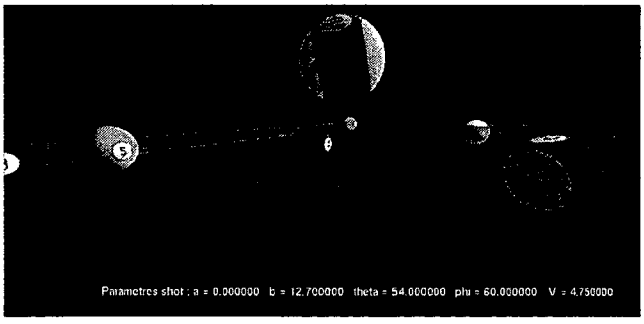

figure 2.6 – Bille 11 en l'air, détection de collision avec la bille 10

La bille 11 va ensuite entrer en régime *FLYING* et détecter la collision avec la

bille 10 (fig 2.6).

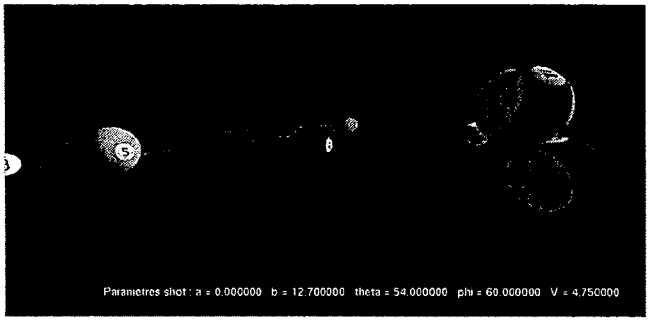

figure 2.7 – Première collision entre la bille 11 et 10

La bille 11 avance jusqu'à sa collision avec la bille 10. La collision est traitée, la bille 11 repart en *FLYING*, effectuant un léger rebond sur la bille 10 (fig 2.7).

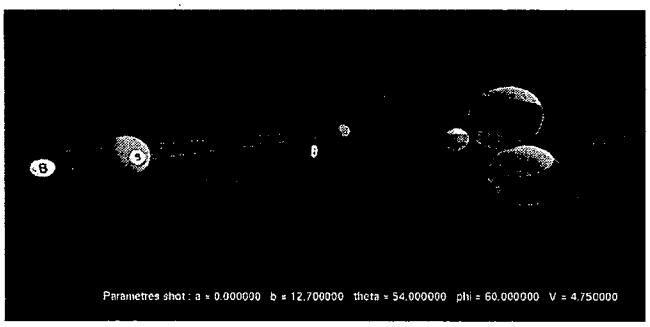

figure 2.8 – Deuxième collision entre la bille 11 et 10

Le léger rebond laisse peu de temps la bille 11 en *FLYING*, celle-ci détectant une nouvelle collision imminente avec la bille 10 (fig 2.8). Cette collision est la première des collisions successives prise en compte.

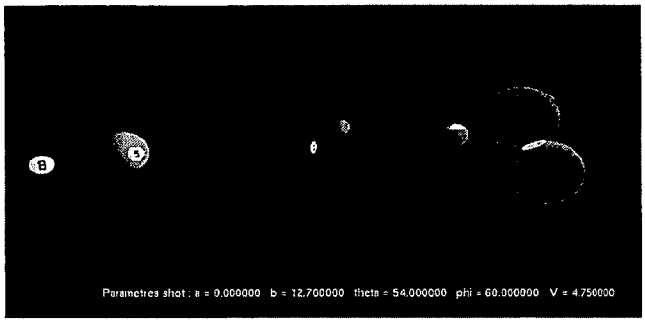

figure 2.9 – Entrée dans le phénomène de *LINK*

La bille 11 et 10 sont successivement soumises à leur collision, la force de gravité poussant la bille 11 à redescendre sur la table. Au bout d'un nombre défini de collision, l'évènement *LINK* est déclenché, la bille 11 devient *LINKER* et la bille 10 *LINKED*. A sa création, il met à jour les billes avec le LCP vu dans le modèle mathématique, avec un pas de temps égal à 0 (il s'agit juste d'ajuster les directions des vélocités afin que les billes ne rentrent plus l'une dans l'autre dès la prochaine avancée, il n'y a aucune perte d'énergie étant donné que le temps n'avance pas).

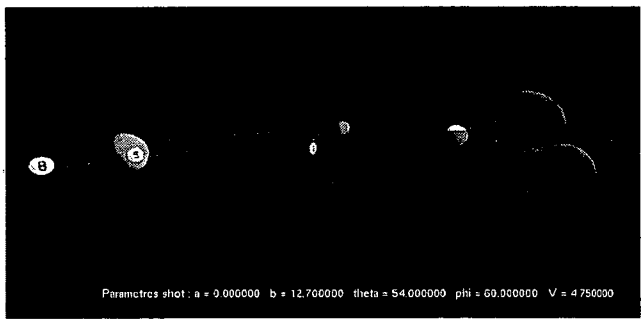

figure 2.10 – Mise à jour des vélocités linéaire et angulaire selon le modèle établi. Avancement de la table d'un pas de temps

Le *LINK* provoque la création d'un sous-évènement *LINKING*. Ce dernier se passera dans un pas de temps défini par le *LINK*. Étant le seul évènement possible des deux billes, le simulateur avance jusqu'au *LINKING*.

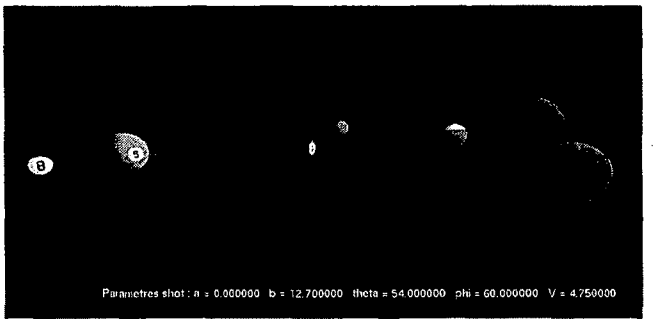

figure 2.11 – Deuxième mise à jour, avancement au prochain pas de temps.

Le simulateur a fini d'avancer. Les billes sont toujours en contact et gardent donc leurs états de mouvement spéciaux. Création d'un nouvel évènement *LINKING*. De même que précédemment, c'est le seul évènement possible, le simulateur avance donc d'un pas de temps.

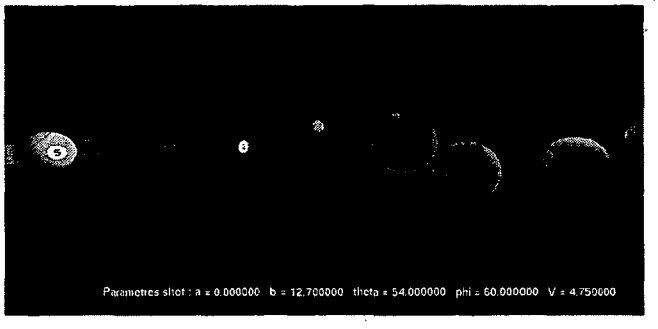

figure 2.12 – Dernier évènement *LINKING*, les billes se séparent.

Le simulateur a de nouveau fini d'avancer. Les billes arrivent dans une dernière configuration de *LINKING* car elles sont toujours en contact, mais il est clair que celui-ci sera le dernier.

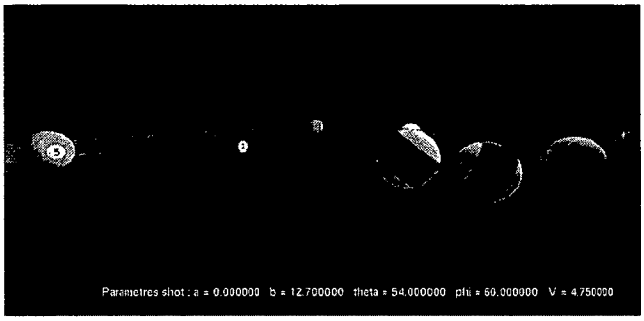

figure 2.13 – Les billes ne sont plus en contact, le *LINK* est terminé.

Après le dernier pas de temps, les billes ne sont plus en contact direct, elles peuvent donc reprendre un régime de mouvement normal. Ici la bille 11 redevient donc *FLYING* et la bille 10 *SLIDING_SPINNING* (par défaut, toutes les billes étant *LINKER* ou *LINKED* deviennent *SLIDING_SPINNING*). Il ne reste aucune bille sur la table en état de *LINKER*, l'évènement *LINK* est donc terminé. Le simulateur peut reprendre un cheminement d'exécution pleinement asynchrone.

Il s'agissait d'un exemple du cas le plus simple possible. il peut cependant y avoir d'autres billes en mouvement aux alentours, ce qui implique de régler le pas de temps afin d'effectuer tous les évènements se produisant en parallèle du *LINK*. Pour ce faire, un ajustement simple est effectué dans l'algorithme de détection d'évènement (fig. 2.14).

Chaque bille déterminera son prochain évènement (comme précédemment) mais lorsque l'évènement sélectionné en cours est de type *LINKING*, alors une autre condition est ajoutée. Si l'évènement calculé a lieu avant le *LINKING* alors il n'y a pas de changement dans l'algorithme, cet évènement devient sélectionné. Cependant si le temps t avant son traitement est supérieur au temps avant le *LINKING* mais qu'il soit en revanche plus petit que ce dernier auquel est ajouté le pas de temps, alors ce pas de temps doit s'adapter. Mathématiquement, avec t_e le temps avant l'évènement calculé, $t_{LINKING}$ le temps avant l'évènement de *LINKING* et Δt le pas de temps du *LINKING*, ceci s'exprime comme :

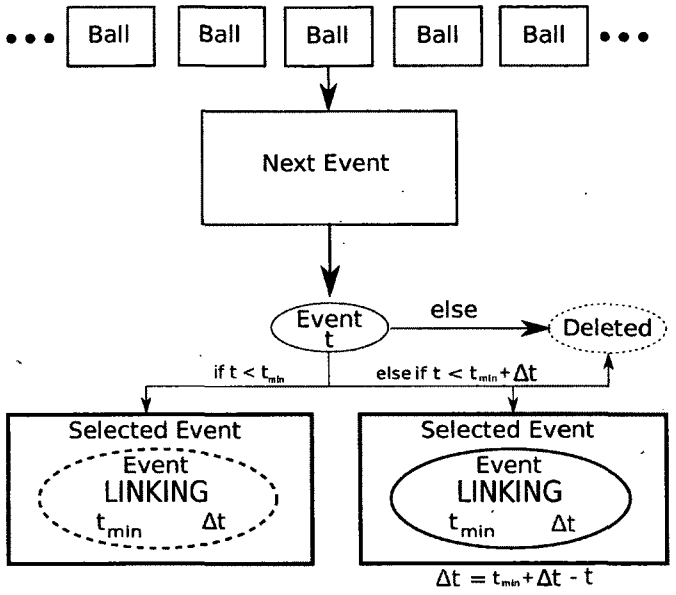

figure 2.14 – Nouveau schéma fonctionnel de NextEvent.

$$t_e > t_{LINKING} \quad \text{et} \quad t_e < t_{LINKING} + \Delta t \quad \text{alors} : \Delta t = t_e - t_{LINKING}$$

C'est le seul changement majeur dans le déroulement de la simulation. Ce changement, auquel on rajoute le traitement du *LINK* débouche sur une hybridation de la simulation synchrone et asynchrone qui résout toutes les situations problématiques auxquelles il était possible d'être confronté.

Il reste cependant un problème : comme Preclik le signale dans [27], les sphères parfaites (le cas du billard traité ici) qui n'ont qu'un contact avec friction mais sans vélocité relative ne seront plus soumises à une force s'opposant au mouvement. Une bille en état de roulement va par conséquent rouler indéfiniment sur la table. Le problème est que si une bille en pousse une autre et que la vélocité relative au point de contact entre la bille *LINKER* et le sol est nulle de même que celle au point entre cette dernière et une bille *LINKED*, alors les deux billes avanceront indéfiniment. Il est donc nécessaire d'ajouter un point d'arrêt à l'évènement *LINK*. Ce point d'arrêt est simple : lorsque l'ensemble des vélocités relatives aux contacts d'un système sont plus petites qu'une certaine limite, alors l'évènement *LINK* s'arrête.

Les situations problématiques sont maintenant traitées par ce procédé, il ne devrait plus y avoir de coups où le simulateur bloquerait. Le subjonctif est utilisé car dans la pratique, il serait nécessaire de tester des milliers de tables différentes pour affirmer la validité absolue du simulateur. Beaucoup de parties ont été simulées, les bugs rencontrés corrigés, mais il existe peut être encore des situations (rares) où le simulateur serait en difficulté.

2.2.5 Résultats

Ici il ne s'agit pas des résultats généraux du simulateur obtenu, mais plutôt de ses performances dans différents cas de figures problématiques. En effet le nouvel évènement peut apparaître dans plusieurs situations qu'il est possible de diviser en deux catégories :

- Poussée par la gravité : il s'agit des cas où une bille en vol glisse sur une bille ou plus. Ces situations impliquent rarement beaucoup de billes (le seul cas de figure serait une casse aérienne, très difficilement réalisable) et se résolvent rapidement sans problèmes. Cette catégorie représente le problème originel posé pour la création du nouvel évènement *LINK*.
- Poussée par accélération angulaire : il existe des situations où une bille en frappe une autre et lui communique une vélocité angulaire non-nulle qui peut la faire accélérer vers d'autres billes en contact (fig. 2.15). Ces cas sont assez fréquents avec les casse aérienne où la première bille frappée dans le triangle va gagner de la vélocité angulaire et linéaire dépendamment de la vélocité linéaire et angulaire de la bille blanche (se vélocité linéaire étant généralement assez grande pour la faire voler du point de départ de la blanche jusqu'au triangle de bille).

Les problèmes de la première catégorie sont généralement simples et facilement résolvable car ils impliquent peu de billes. La résolution est donc rapide et robuste. Le problème intervient surtout lorsque le simulateur est confronté à problème de poussée par accélération angulaire lors d'une casse aérienne. Une casse aérienne implique que toutes les billes (ou presque) seront impliquées, ce qui fait grandir les matrices utilisées par le LCP de façon exponentielle. De même les points de contacts seront nombreux

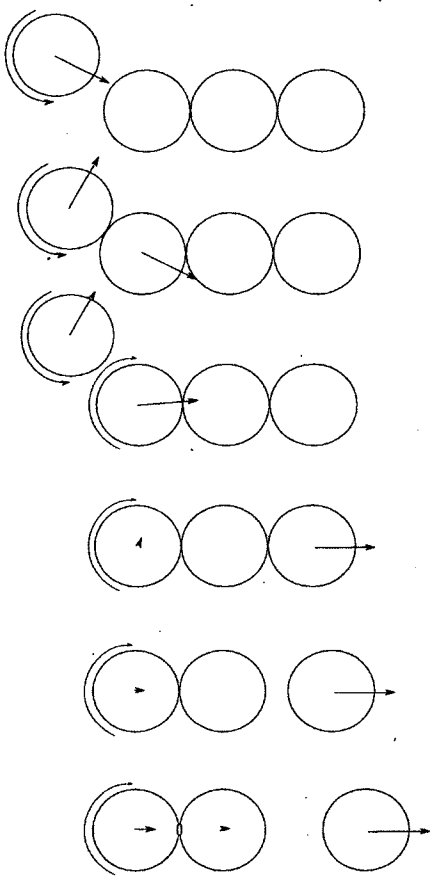

figure 2.15 – Schéma explicatif du problème de la catégorie "Poussée par accélération angulaire". La bille en vol rentre en contact avec la première des trois billes au repos, communiquant une partie de sa vélocité angulaire. La vélocité linéaire est quand à elle transmise à la dernière bille en contact. L'accélération transmise initialement est donc utilisée par la première bille pour rentrer en collision avec la deuxième bille indéfiniment.

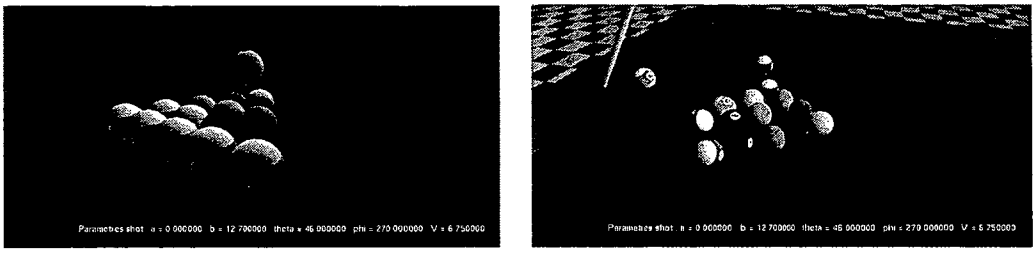

figure 2.16 – Exemple de problème de la catégorie "Poussée par accélération angulaire". La bille blanche va heurter la bille 5 ce qui lui communiquera de la vélocité angulaire alors que la vélocité linéaire sera transmise aux billes aux extrêmes du triangle. Le simulateur mettra plusieurs secondes à résoudre le LCP sur les premiers pas de *LINK*

car les billes à la casse sont en contact avec 2 à 6 billes (fig. 2.16). La résolution va donc être bien plus complexe dans ce type de situation.

Le résolveur utilisé pour trouver la solution du LCP, PATH, prends parfois plusieurs secondes pour résoudre le système lors d'un problème de cette deuxième catégorie. Il est clair que ce type de situation est **TRES** coûteuse pour un simulateur ayant pour but d'être rapide ! Ces situations sont donc traitées mais il est fortement déconseillé à un joueur d'utiliser des coups qui mèneraient à ces situations.

Chapitre 3

Résultats et remarques

Tous les éléments permettant le bon fonctionnement du simulateur ont maintenant été abordés, il est temps de confronter la simulation à la réalité. Cette partie se concentrera donc sur la validité du travail par rapport aux objectifs fixés : le simulateur est-il à la hauteur de nos espérances ? Quelles sont ses limites ? Toutes ces questions seront abordées dans ce dernier chapitre. Tout d'abord, des tests confronteront la réalité observée aux coups simulés, permettant de tester jusqu'où le simulateur représente fidèlement la physique réelle ce qui permettra par la suite de mettre en avant ses limites actuelles. Pour finir, une comparaison avec un simulateur existant sera effectuée dans le but d'expliciter les effets d'une plus grande liberté d'actions sur un joueur évoluant dans un environnement moins restreint qu'un autre.

3.1 Résultats face à la réalité physique

Pour tester la validité physique du simulateur, il était initialement prévu d'utiliser un bras robotique monté sur une table de billard. Ce dernier aurait pu effectuer des coups suivant un état de table prédéfini et les cinq paramètres spécifiant un coup. Celui-ci, combiné à une caméra aurait pu lancer une quantité impressionnante de tests qui auraient pu ensuite être confrontés facilement aux résultats du simulateur

qui utilise le même format pour effectuer un coup. Cependant ce dernier n'est plus entretenu et est devenu un projet abandonné pour l'université qui l'avait développé. N'ayant plus accès à cet outil et ne disposant pas des ressources nécessaires pour effectuer une calibration aussi poussée qu'elle aurait pu l'être avec le robot, une étude qualitative sera proposée. Celle-ci consiste en une série de vidéos prises avec un appareil Casio Exilim disposant d'une capacité de 1000 images par seconde (1000 FPS). Les coups filmés étant effectués par des joueurs humains et la résolution des vidéos se rapprochant de 224x64, il est difficile de faire une étude précise. Le but est donc de retrouver les coups filmés avec le simulateur. Ceci implique bien sûr de calibrer ce dernier et de trouver cinq paramètres initiaux se rapprochant de ceux utilisés pour les coups filmés.

La première vidéo concerne une collision entre deux billes. Il s'agit ici de montrer les conséquences d'une vélocité angulaire en z non-nulle lorsque les deux billes sont en contact.

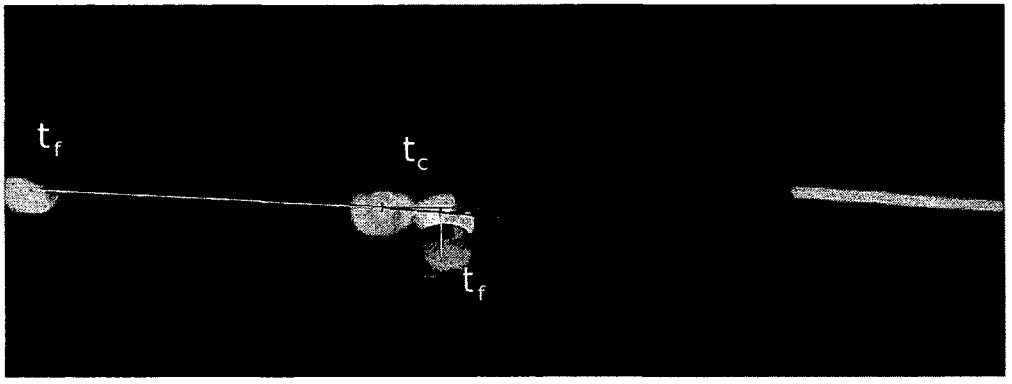

figure 3.1 – Collision de billes étudiée. La bille de droite (numérotée 10) frappe la bille de gauche (11). La droite rouge relie les centres des billes à l'instant initial, le droite verte passe par le centre de la 10 à l'instant initial et son centre au moment de la collision, la droite violette représente la direction que devrait prendre la bille 11 si l'effet de côté n'était pas pris en compte (elle relie simplement les centres des billes lorsqu'elles sont en contact), la droite jaune est la direction prise par la 11 après la collision et le segment turquoise le déplacement de la 10 jusqu'à son arrêt complet. Ici, t_c note l'emplacement des billes au moment de la collision et t_f leur position finale.

Il est ici évident que l'effet communiqué à la bille 11 influence de quelques degrés la direction de la bille 10 après la collision.

Des paramètres sont entrés de façon à ce que la situation simulée soit qualitativement similaire à celle observée.

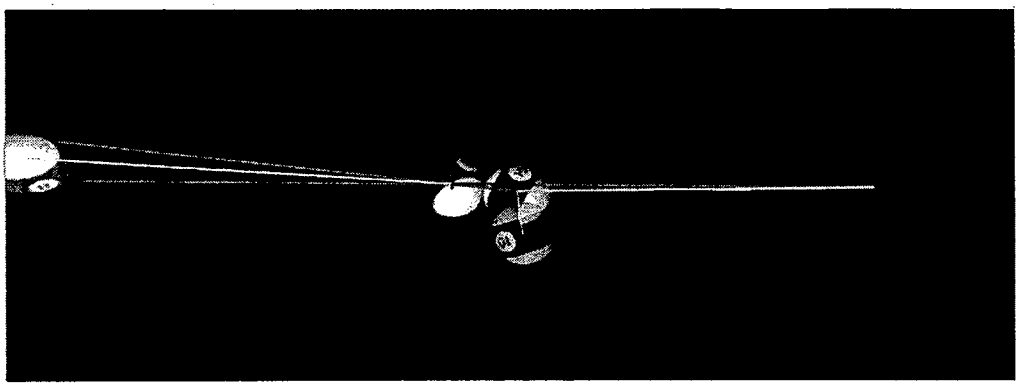

figure 3.2 – Collision de billes simulée. Les droites de couleurs ont toujours les mêmes rôles attribués. On observe que le comportement de la bille 11 est très similaire à celui exposé dans la figure précédente.

Les figures 3.1 et 3.2 montrent clairement que les comportements, observé et simulé, sont très similaires. Les seules différences que l'on peut noter sont pour la majeure partie dûes au fait que la simulation est basée sur une observation qualitative. Si les cinq paramètres initiaux de la vidéo étaient connus, les différences auraient pu être corrigées avec une calibration efficace. Il est cependant confortable d'observer que les collisions de billes soient très proches de la réalité, confirmant la validité du modèle, au moins dans les situations semblables au cas étudié. Il est clair que la vélocité angulaire en z influence le déplacement post-collision de la bille 11.

La série de tests suivante se penche sur les collisions entre une bille et une bande. Le premier test consiste en une collision simple avec une vélocité angulaire en z de moyenne amplitude avec une vélocité linéaire faible. La vidéo de la situation observée donne :

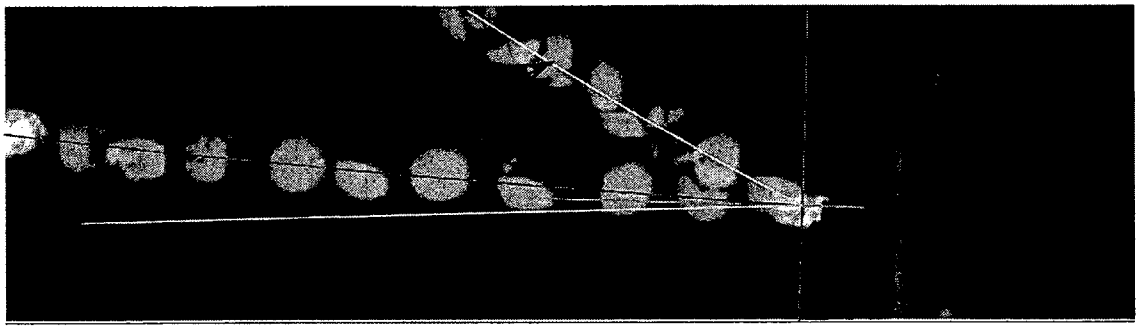

figure 3.3 – Collision entre une bille et une bande. La ligne bleue représente la trajectoire de la bille avant sa collision, la droite rouge est parallèle à la bande et sa perpendiculaire en orange permet de déterminer la trajectoire post-collision, la droite turquoise étant cette dernière si la bille n'avait pas de vélocité angulaire en z. La droite jaune est la trajectoire réelle prise par la bille après son contact avec la bande.

La vélocité angulaire en z communiquée à la bille décale de plusieurs degrés la trajectoire que prendrait la bille si elle était entrée en collision avec la bande sans cette vitesse. On peut observer que le frottement entre la bille et la bande est très fort car avec une magnitude moyenne, le décalage est conséquent.

En simulant un coup similaire, on obtient :

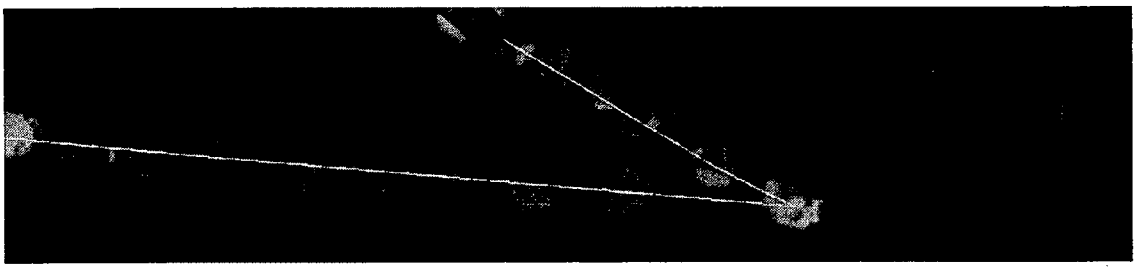

figure 3.4 – Collision entre une bille et une bande simulée.

Il est évident que le coup simulé se rapproche beaucoup de la réalité. Le fait que l'étude soit qualitative ne permet pas d'affirmer pleinement que le simulateur est en accord parfait avec la physique réelle des collisions de bande, mais il n'est pas imprudent d'avancer le fait que le modèle pour ce genre de coup est plutôt robuste.

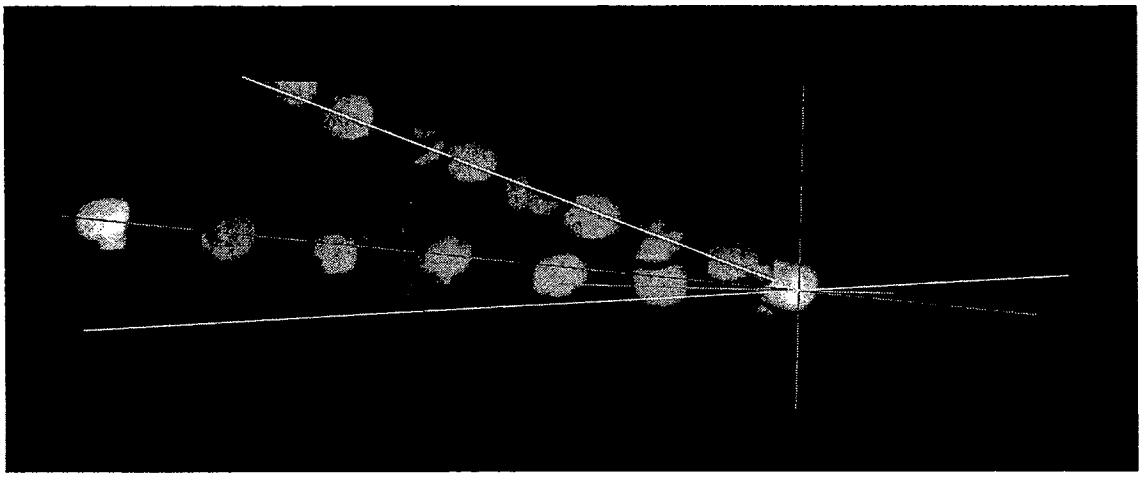

figure 3.5 – Autre collision entre une bille et une bande. Les droites de couleurs ont toujours les mêmes rôles attribués. La trace noire traversant la bande à droite est un défaut dû à l'acquisition de la vidéo, elle n'intervient en aucun cas dans le comportement de la bille ou de la bande.

Pour confirmer cette idée, une deuxième vidéo a été réalisée, cette fois-ci avec plus de vélocité linéaire mais un peu moins de vélocité angulaire en z.

La simulation, pour un coup adapté, est toujours très proche de la réalité (fig 3.6).

Le prochain test se centre toujours sur les collisions bille-bande, mais cette fois-ci avec une forte vélocité linéaire. Cette fois un problème survient pour recréer la situation dans le simulateur : comme l'expose la figure 3.7, la bande se déforme au contact de la bille. Cette déformation n'est pour l'instant pas comprise dans la physique étudiée et développée pour le simulateur, il est donc impossible de retranscrire un coup de ce type.

Pour confronter la réalité impliquant la déformation et le simulateur, le coup initial est approximé et testé (fig 3.8). Les résultats sont évidemment éloignés étant donné l'amplitude de la déformation. Ceci met donc en évidence un point faible du simulateur, l'incapacité à reproduire la déformation des bandes limite le réalisme des coups impliquant des collisions entre bille et bande si la bille a une vélocité trop élevée. Il est cependant difficile de fixer un seuil où les bandes sont considérées comme trop déformées dans la réalité pour pouvoir reproduire le coup dans le simulateur. En effet, ce paramètre dépend de chaque table, celle-ci ayant des bandes plus ou moins rigides

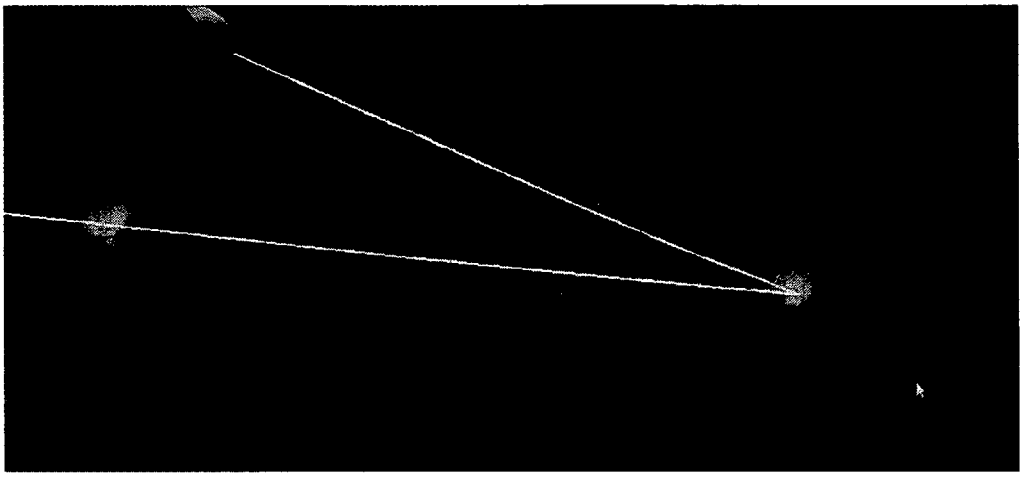

figure 3.6 – Deuxième collision entre une bille et une bande simulée.

et/ou usées. Il s'agit là d'un défaut majeur, la déformation de la bande pouvant agir grandement sur la trajectoire d'une bille [1].

3.2 Évaluation de la sensibilité d'un coup

Dans cette section, deux simulateurs seront confrontés pour évaluer la différence de sensibilité qu'apporte les nouvelles libertés du simulateur. Pour bien les mettre en avant, FastFiz [19] sera pris comme « simulateur asynchrone concurrent » car il pêche dans des domaines où le simulateur développé est à l'aise : l'effet de la vélocité angulaire en z dans les différentes collisions (entre billes ou entre une bille et une bande) n'est pas pris en compte. Les coups calculés utilisent le joueur PoolMaster [18], développé par Jean-François Landry. Les points blancs dans les prochaines images représentent la position finale de la blanche. Ceux-ci sont au nombre de 500 et sont générés avec un bruit gaussien venant perturber les cinq paramètres du coup. Pour les prochaines figures, l'image du haut est tirée de FastFiz, la deuxième du simulateur développé. On comparera ici les zones dessinées par l'ensemble des points dans chaque simulateur, le bruit gaussien étant généré avec des valeurs différentes pour l'un et l'autre.

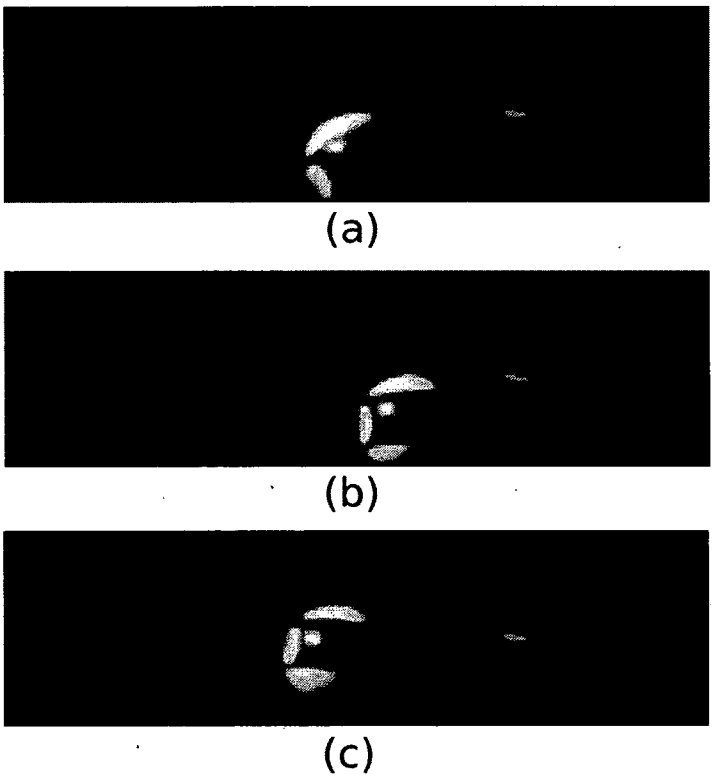

figure 3.7 – Collision entre une bille ayant une forte vélocité linéaire et une bande. En (a) la bille se prépare à rentrer en contact avec la bande, ce qui arrive en (b). La bille rentre à l'intérieur de la zone qui appartenait auparavant à la bande, ce qui représente une déformation. En (c) la bille est expulsée par la reformation de la bande.

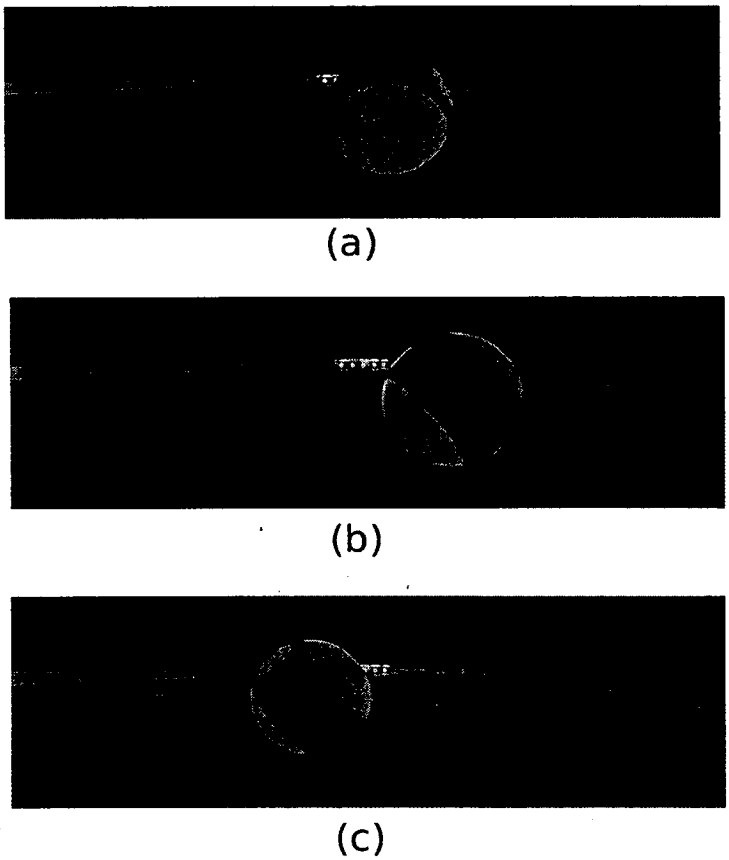

figure 3.8 – Collision entre une bille ayant une forte vélocité linéaire et une bande. En (a) la bille arrive vers la bande avec une grand vélocité et rentre en contact avec elle en (b). La bande ne pouvant se déformer, la bille subit une faible vélocité négative en z, la faisant un peu décoller du tapis en (c).

Le premier coup est un coup très simple sans vélocité angulaire en z. Les résultats montré sur la figure 3.9 sont très similaires, le bruit gaussien étant trop petit pour provoquer des changements brutaux sur a.

Le prochain coup illustre les conséquences de la prise en compte de la vélocité angulaire en z lors d'une collision de bande (fig. 3.10). Pour FastFiz, on obtient une dispersion quasi-linéaire. Les coups simulés avec le simulateur développé sont bien plus séparés : une surface bien plus grande se dessine, montrant bien l'importance du paramètre a (ici 6 mm à gauche du centre) et la sensibilité qui lui est associée.

La troisième comparaison se concentre sur un coup empochant une bille et ayant pour objectif un replacement pour la blanche spécifique (fig. 3.11). Le coup est déterminé et simulé sur le nouveau simulateur, puis les mêmes paramètres sont donnés à FastFiz qui exécute donc le même coup. Il est clair que le fait que la vélocité angulaire en z des billes n'étant pas pris en compte dans ce dernier, la reposition de la bille blanche ne se situe pas dans la même zone finale que celle du nouveau simulateur. Le coup étant relativement puissant (4.25 m/s), les collisions entre billes et avec les bandes sont utilisées, ce qui explique ce décalage important.

Ces exemples démontrent clairement que le comportement des billes est bien plus libre dans le nouveau simulateur développé que ce que peut proposer FastFiz. Le fait d'utiliser la vélocité angulaire en z et les déplacements sur l'axe z permettent d'accéder à des stratégies de coups inaccessibles dans FastFiz.

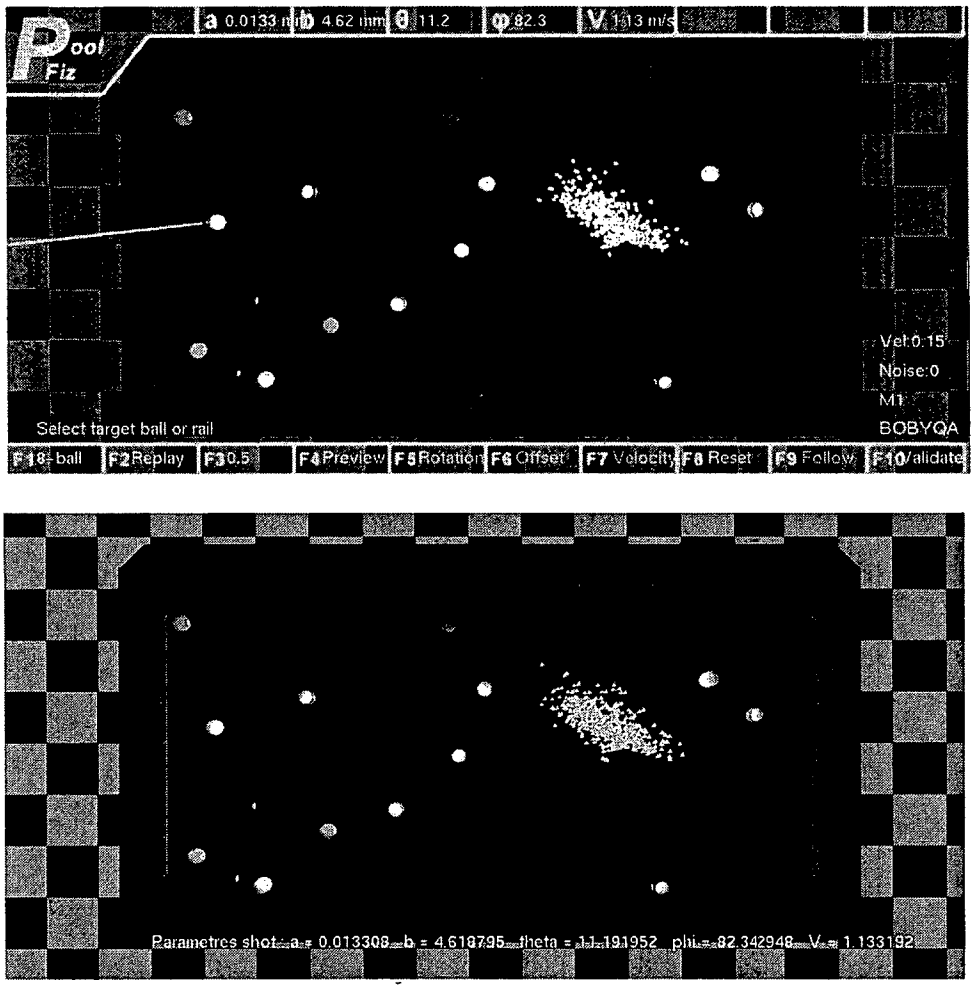

figure 3.9 – Comparaison de la répartition de la reposition de la blanche lors de son immobilisation après un coup. Les surfaces sont très similaires malgré le bruit gaussien influant aussi sur le paramètre a.

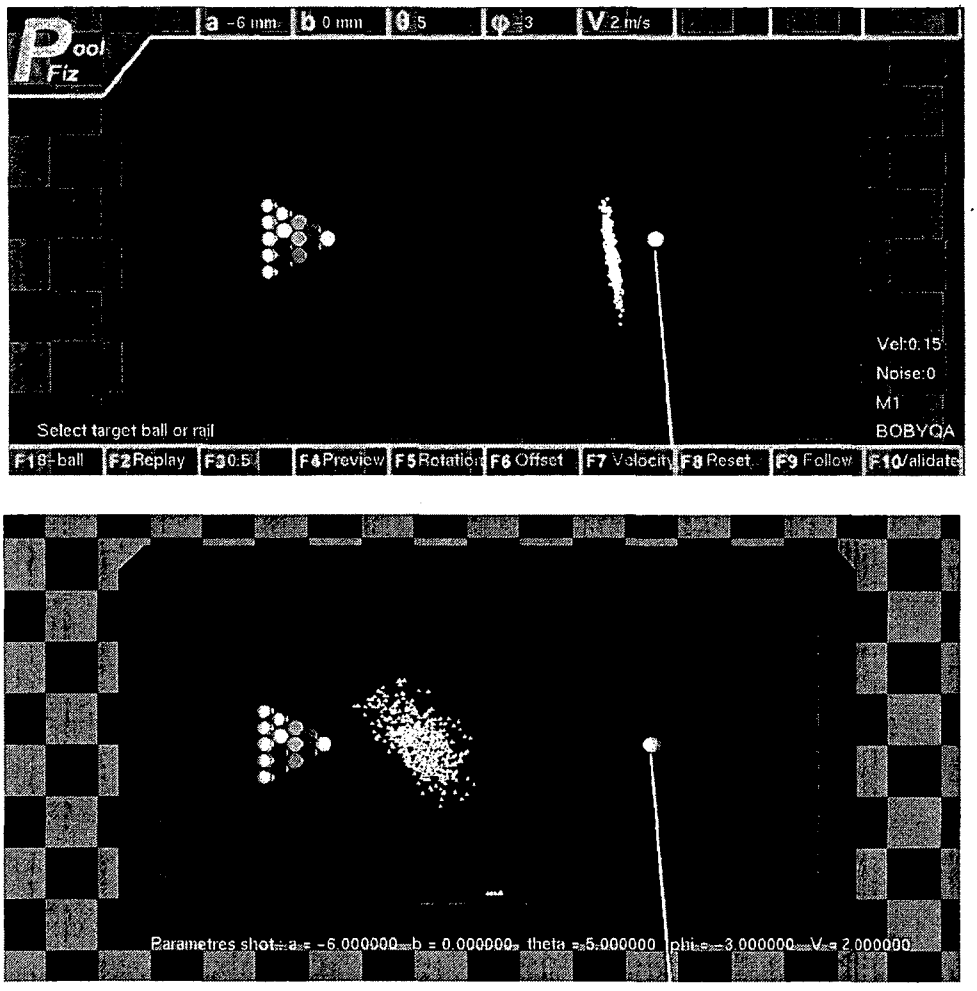

figure 3.10 – Comparaison de la répartition de la reposition de la blanche après collision avec une bande et du bruit gaussien appliqué aux paramètres initiaux.

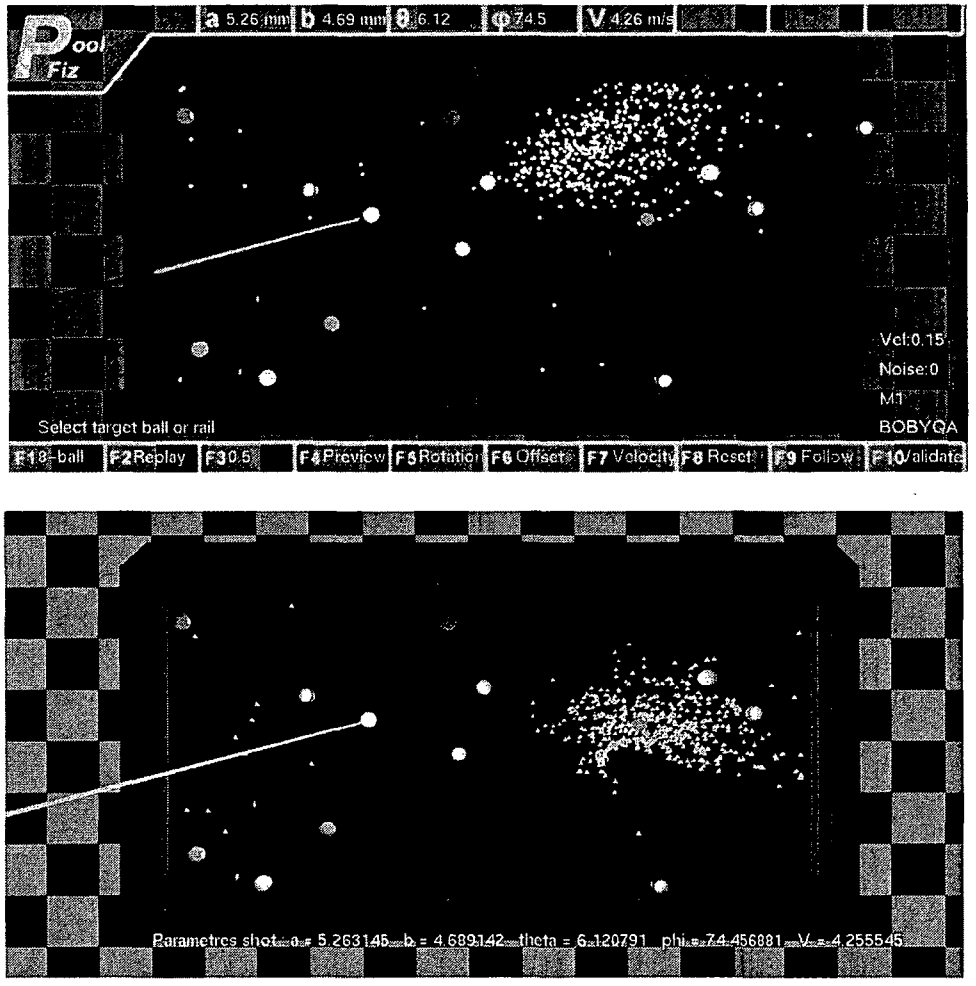

figure 3.11 – Comparaison de la répartition de la reposition de la blanche après un coup empochant une bille et ayant une vélocité angulaire en z non nulle. Le point orange représente la reposition de la blanche définie comme objectif à atteindre.

Conclusion

La simulation parfaite du jeu de billard est quasiment inaccessible car elle impliquerait beaucoup trop de phénomènes physiques pouvant interagir en même temps. Il est donc nécessaire de faire des approximations. Le simulateur développé ici utilise donc cette idée en proposant une modélisation physique qui n'est pas dénuée de simplifications, mais qui recouvre une plus grande liberté d'action que ce que la majorité des simulateurs libres proposent. De plus, cette modélisation est assez fidèle pour observer que qualitativement, les coups simulés ont un comportement similaire à ceux de la réalité. L'utilisation de la simulation par évènements permet elle aussi de contribuer à la précision générale du simulateur et à son efficacité en terme de rapidité d'exécution. L'utilisation de certaines techniques comme la liste d'évènements potentiels, permet aussi de réduire les temps de calculs afin d'avoir un simulateur étant précis mais en restant relativement léger dans son exécution.
Le simulateur n'est cependant pas encore exactement conforme à la réalité pour certains points, l'absence de la déformation des bandes arrivant en tête de ces problèmes. On pourrait l'approcher avec un système de masse-ressort suffisamment réaliste (impliquant l'effet de compression et décompression de la bande). Le problème n'a cependant pas encore été abordé pour tester la validité de cette hypothèse dans un cadre de réalisme optimal. Les autres points concernent la calibration qui aurait pu être plus respectueuse de la réalité avec un matériel avancé pour effectuer des coups précis et un matériel d'acquisition vidéo permettant une visualisation de plusieurs points de vue. De même, certaines tâches restent à entreprendre pour rendre la simulation encore plus légère, les travaux présentés dans [21] seraient par exemple très bénéfiques à la fluidité générale du simulateur.
Le simulateur s'approche donc de son objectif initial, à savoir se confronter le plus

fidèlement possible à la réalité, mais n'est pas encore capable de reproduire tous les coups à l'heure actuelle. Il n'est pas cependant impossible de créer et d'implémenter un modèle de déformation d'objets non-rigides qui permettrait par la suite de pallier à ce problème tout en restant dans une optique de simulation par évènements. Ceci, combiné à une calibration minutieuse pourrait être suffisant pour reproduire un comportement réaliste.

Avec un simulateur assez réaliste pour effectuer des coups plus ou moins simples, on peut envisager plus tard de relier celui-ci à un bras robotique et un joueur IA adapté pour défier des joueurs professionnels. De quoi créer de nouveaux challenges pour les meilleurs joueurs du monde !

Annexe A

Détermination de u

Dans le chapitre 1, il est donné la formule de u sans rentrer en profondeur dans les calculs. Ici seront détaillés les étapes pour arriver à ce résultat :

On cherche une relation entre $\vec{V}_{c\,final}$ et $\vec{V}_{c\,init}$:

$$\vec{V}_{c\,final} = (\vec{v}_{1final} - \vec{v}_{2final}) - [(\vec{v}_{1final} - \vec{v}_{2final}) \cdot \vec{n}]\,\vec{n} + \vec{n} \times (R_1\vec{\omega}_{1final} + R_2\vec{\omega}_{2final})$$

avec :

$$\vec{v}_{1final} - \vec{v}_{2final} = \vec{v}_{1init} + \frac{P}{M_1}\left(\vec{n} - \mu\frac{\vec{V}_{c\,init}}{\|\vec{V}_{c\,init}\|}\right) - \left(\vec{v}_{2init} + \frac{M_1}{M_2}\left[\vec{v}_{1init} - \vec{v}_{1final}\right]\right)$$

$$\vec{v}_{1final} - \vec{v}_{2final} = \vec{v}_{1init} + \frac{P}{M_1}\left(\vec{n} - \mu\frac{\vec{V}_{c\,init}}{\|\vec{V}_{c\,init}\|}\right) - \left(\vec{v}_{2init} + \frac{M_1}{M_2}\left[\vec{v}_{1init} - \left(\vec{v}_{1init} + \frac{P}{M_1}\left(\vec{n} - \mu\frac{\vec{V}_{c\,init}}{\|\vec{V}_{c\,init}\|}\right)\right)\right]\right)$$

$$\vec{v}_{1final} - \vec{v}_{2final} = \vec{v}_{1init} - \vec{v}_{2init} + \left(1 + \frac{M_1}{M_2}\right)\left(\frac{P}{M_1}\right)\left(\vec{n} - \mu\frac{\vec{V}_{c\,init}}{\|\vec{V}_{c\,init}\|}\right)$$

$$\vec{v}_{1final} - \vec{v}_{2final} = \vec{v}_{1init} - \vec{v}_{2init} + \left(1 + \frac{M_1}{M_2}\right)\left(\frac{P}{M_1}\right)\vec{n} - \left(1 + \frac{M_1}{M_2}\right)\left(\frac{P}{M_1}\right)\left(\mu\frac{\vec{V}_{c\,init}}{\|\vec{V}_{c\,init}\|}\right)$$

et :

$$[(\vec{v}_{1final} - \vec{v}_{2final})\,\vec{n}]\,\vec{n} = [(\vec{v}_{1init} - \vec{v}_{2init})\,\vec{n}]\,\vec{n} + \left(1 + \frac{M_1}{M_2}\right)\left(\frac{P}{M_1}\right)\vec{n}$$

et :

$$\vec{n}\times(R_1\vec{\omega}_{1final}+R_2\vec{\omega}_{2final})=\vec{n}\times\Big(R_1\left[\vec{\omega}_{1init}+\tfrac{5}{2R_1}\vec{n}\times(\vec{v}_{1init}-\vec{v}_{1final})\right]+R_2\left[\vec{\omega}_{2init}-\tfrac{M_1R_1}{M_2R_2}(\vec{\omega}_{1init}-\vec{\omega}_{1final})\right]$$

sachant que :

$$R_1\vec{\omega}_{1final}+R_2\vec{\omega}_{2final}=R_1\vec{\omega}_{1init}+\left(\tfrac{5}{2}\tfrac{P}{M_1}\mu\right)\vec{n}\times\tfrac{\vec{V}_{cinit}}{\|\vec{V}_{cinit}\|}+R_2\vec{\omega}_{2init}-\tfrac{M_1R_1}{M_2}\left(-\left(\tfrac{5}{2R_1}\tfrac{P}{M_1}\mu\right)\vec{n}\times\tfrac{\vec{V}_{cinit}}{\|\vec{V}_{cinit}\|}\right)$$

$$R_1\vec{\omega}_{1final}+R_2\vec{\omega}_{2final}=R_1\vec{\omega}_{1init}+\left(\tfrac{5}{2}\tfrac{P}{M_1}\mu\right)\vec{n}\times\tfrac{\vec{V}_{cinit}}{\|\vec{V}_{cinit}\|}+R_2\vec{\omega}_{2init}+\left(\tfrac{5}{2}\tfrac{P}{M_2}\mu\right)\vec{n}\times\tfrac{\vec{V}_{cinit}}{\|\vec{V}_{cinit}\|}$$

on a donc :

$$\vec{n}\times(R_1\vec{\omega}_{1final}+R_2\vec{\omega}_{2final})=R_1\vec{n}\times\vec{\omega}_{1init}+\left(\tfrac{5}{2}\tfrac{P}{M_1}\mu\right)\tfrac{\vec{n}\times\vec{n}\times\vec{V}_{cinit}}{\|\vec{V}_{cinit}\|}+R_2\vec{n}\times\vec{\omega}_{2init}+\left(\tfrac{5}{2}\tfrac{P}{M_2}\mu\right)\tfrac{\vec{n}\times\vec{n}\times\vec{V}_{cinit}}{\|\vec{V}_{cinit}\|}$$

$$\vec{n}\times(R_1\vec{\omega}_{1final}+R_2\vec{\omega}_{2final})=R_1\vec{n}\times\vec{\omega}_{1init}-\left(\tfrac{5}{2}\tfrac{P}{M_1}\mu\right)\tfrac{\vec{V}_{cinit}}{\|\vec{V}_{cinit}\|}+R_2\vec{n}\times\vec{\omega}_{2init}-\left(\tfrac{5}{2}\tfrac{P}{M_2}\mu\right)\tfrac{\vec{V}_{cinit}}{\|\vec{V}_{cinit}\|}$$

l'équation de $\vec{V}_{cfinal}$ devient :

$$\vec{V}_{cfinal}=\vec{v}_{1init}-\vec{v}_{2init}+\left(1+\tfrac{M_1}{M_2}\right)\left(\tfrac{P}{M_1}\right)\vec{n}-\left(1+\tfrac{M_1}{M_2}\right)\left(\tfrac{P}{M_1}\right)\left(\mu\tfrac{\vec{V}_{cinit}}{\|\vec{V}_{cinit}\|}\right)-\left[(\vec{v}_{1init}-\vec{v}_{2init})\,\vec{n}\right]\vec{n}-\left(1+\tfrac{M_1}{M_2}\right)\left(\tfrac{P}{M_1}\right)\vec{n}+R_1\vec{n}\times\vec{\omega}_{1init}-\left(\tfrac{5}{2}\tfrac{P}{M_1}\mu\right)\tfrac{\vec{V}_{cinit}}{\|\vec{V}_{cinit}\|}+R_2\vec{n}\times\vec{\omega}_{2init}-\left(\tfrac{5}{2}\tfrac{P}{M_2}\mu\right)\tfrac{\vec{V}_{cinit}}{\|\vec{V}_{cinit}\|}$$

$$\vec{V}_{cfinal}=\vec{V}_{cinit}-\left(1+\tfrac{M_1}{M_2}\right)\left(\tfrac{P}{M_1}\right)\left(\mu\tfrac{\vec{V}_{cinit}}{\|\vec{V}_{cinit}\|}\right)-\left(\tfrac{5}{2}\tfrac{P}{M_1}\mu\right)\tfrac{\vec{V}_{cinit}}{\|\vec{V}_{cinit}\|}-\left(\tfrac{5}{2}\tfrac{P}{M_2}\mu\right)\tfrac{\vec{V}_{cinit}}{\|\vec{V}_{cinit}\|}$$

$$\vec{V}_{cfinal}=\vec{V}_{cinit}-\left(\tfrac{P}{M_1}\right)\left(\mu\tfrac{\vec{V}_{cinit}}{\|\vec{V}_{cinit}\|}\right)-\left(\tfrac{P}{M_2}\right)\left(\mu\tfrac{\vec{V}_{cinit}}{\|\vec{V}_{cinit}\|}\right)-\left(\tfrac{5}{2}\tfrac{P}{M_1}\right)\left(\mu\tfrac{\vec{V}_{cinit}}{\|\vec{V}_{cinit}\|}\right)-\left(\tfrac{5}{2}\tfrac{P}{M_2}\right)\left(\mu\tfrac{\vec{V}_{cinit}}{\|\vec{V}_{cinit}\|}\right)$$

$$\vec{V}_{cfinal}=\vec{V}_{cinit}\left(1-\left(\tfrac{\mu}{\|\vec{V}_{cinit}\|}\right)\left[\left(\tfrac{P}{M_1}\right)+\left(\tfrac{PM_1}{M_2M_1}\right)+\left(\tfrac{5}{2}\tfrac{PM_2}{M_1M_2}\right)+\left(\tfrac{5}{2}\tfrac{PM_1}{M_2M_1}\right)\right]\right)$$

$$\vec{V}_{cfinal}=\vec{V}_{cinit}\left(1-\left(\tfrac{\mu}{\|\vec{V}_{cinit}\|}\left(\tfrac{P}{M_1}\right)\right)\left[1+\left(\tfrac{M_1}{M_2}\right)+\left(\tfrac{5}{2}\tfrac{M_2}{M_2}\right)+\left(\tfrac{5}{2}\tfrac{M_1}{M_2}\right)\right]\right)$$

$$\vec{V}_{cfinal}=\vec{V}_{cinit}\left(1-\left(\tfrac{\mu}{\|\vec{V}_{cinit}\|}\left(\tfrac{P}{M_1}\right)\left(\tfrac{7}{2}\right)\right)\left[1+\left(\tfrac{M_1}{M_2}\right)\right]\right)$$

u s'exprime donc comme :

$$u = 1 - \left(\frac{\mu}{\|\vec{V}_{c\,init}\|} \left(\frac{P}{M_1} \right) \left(\frac{7}{2} \right) \right) \left(1 + \left(\frac{M_1}{M_2} \right) \right)$$

Annexe A. Détermination de u

Bibliographie

[1] D. ALCIATORE.
The Illustrated Principles of Pool and Billiards.
Sterling Publishing, 2004.

[2] M. Ebne ALIAN, S. Bagheri SHOURAKI, M.T. Manzuri SHALMANI, P. KARIMIAN et P. SABZMEYDANI.
« Robotshark : a gantry pool player robot ».
Dans *ISR 2004 : 35th Intl. Sym. Rob.*, March 2004.

[3] M. ANITESCU.
« Optimization-Based Simulation of Nonsmooth Rigid Multibody Dynamics ».
Math. Program., 105:113–143, 2006.

[4] M. ANITESCU et G. HART.
« A Constraint-Stabilized Time-Stepping Approach for Rigid Multibody Dynamics with Joints, Contact and Friction ».
International Journal For Numerical Methods in Engineering, 60:2335–2371, 2004.

[5] M. ANITESCU et F.A. POTRA.
« Formulating Dynamic Multi-Rigid-Body Contact Problems with Friction as Solvable Linear Complementarity Problems ».
Nonlinear Dynamics, 14:231–247, 1997.

[6] N.P. BALI.
Golden Algebra.
Firewal Media, 2010.

[7] D. BARAFF.
« Issues in Computing Contact Forces for Non-Penetrating Rigid Bodies ».
Algorithmica, 10:292–352, 1993.

[8] F. BERGER.
« FooBillard ».
`http://foobillard.sunsite.dk/`.

[9] P. BRATLEY, B. FOX et L. SCHRAGE.
A Guide to Simulation.
Springer-Verlag, 1983.

[10] G.G. CORIOLIS.
Théorie Mathématique des Effets du Jeu de Billard.
Jacques Gabay, 1835 (republié en 1990).

[11] R. COTTLE, J. PANG et R. STONE.
The Linear Complementarity Problem.
Academic Press, 1992.

[12] S. DIRKSE et M. FERRIS.
« The PATH Solver : A Non-Monotone Stabilization Scheme For Mixed Complementarity Problems ».
Optimization Methods and Software, 5:123–156, 1995.

[13] M. Galassi et AL.
GNU Scientific Library Reference Manual, 3e éd., 2009.
Manuel distribué avec l'Atelier B.

[14] G.S. FISHMAN.
Concepts and Methods in Discrete Event Digital Simulation.
John Wiley & Sons, 1973.

[15] M. GREENSPAN, J. LAM, M. GODARD, I. ZAIDI, S. JORDAN, W. LECKIE, K. ANDERSON et D. DUPUIS.
« Toward a Competitive Pool-Playing Robot ».
Computer, 41(1):46–53, 2008.

[16] HICKOK.
« Billiard—history », April 2008.
http ://www.hickoksports.com/history/billiard.shtml.

[17] A. KLARBRING, P. CHRISTENSEN, J.S. PANG et N. STRÖMBERG.
« Formulation and Comparison of Algorithms for Frictional Contact Problems ».
International Journal for Numerical Methods in Engineering, 42:145–173, 1999.

[18] J-F. LANDRY, J-P. DUSSAULT et P. MAHEY.
« A robust controller for a two-layered approach applied to the game of billiards ».
Entertainment Computing, 2012.

[19] W. LECKIE et M. GREENSPAN.
« An Event-Based Pool Physics Simulator ».
Dans *ACG*, pages 247–262, 2006.

[20] F. LONG, J. HERLAND, M.-Ch. TESSIER, D. NAULLS, A. ROTH, G. ROTH et M. GREENSPAN.
« An Experiment in Automatic Potting ».
IROS 2004 :IEEE/RSJ Intl. Conf. Intell. Rob. Sys, pages 361–366, 2004.

[21] B.D. LUBACHEVSKY.
« How to Simulate Billiards and Similar Systems ».
Journal of Computational Physics, 94:255, juin 1991.

[22] W.C. MARLOW.
The Physics of Pocket Billiards.
Marlow Advanced Systems Technologies, 1995.

[23] S. MATHAVAN, M.R. JACKSON et R.M. PARKIN.
« Application of High-Speed Imaging to Determining the Dynamics of Billiards ».
American Journal of Physics, 77:788–794, 2009.

[24] T. NIERHOFF, O. KOURAKOS et S. HIRCHE.
« Playing pool with a dual-armed robot ».
Dans *ICRA*, pages 3445–3446, 2011.

[25] D. PAPAVASILIOU.
« Billards ».
`http://www.nongnu.org/billiards/`.

[26] R. PETIT.
Billard Théorie du jeu, 2e éd.
Chiron Editeur, 2004.

[27] T. PRECLIK.
« Frictional Rigid Body Dynamics ».
Mémoire de maîtrise, Friedrich-Alexander University Erlangen-Nuremberg, Allemagne, 2007.

[28] M. SHAMOS.
« A Brief History of the Noble Game of Billiards », 1995.
http ://www.bca-pool.com/aboutus/history/start.shtml.

[29] R. SHEPARD.
Amateur Physics for the Amateur Pool Player, 3e éd.
auto-publié, 1997.

[30] M. SMITH.
« PickPocket : A computer billiards shark ».
Artif. Intell., 171(16-17):1069–1091, 2007.

[31] R. SMITH.
« Open Dynamics Engine ».
`http://www.ode.org/`.

[32] D.E. STEWART.
« Rigid-Body Dynamics with Friction and Impact ».
SIAM Review, 42:3–39, 2000.

[33] D.E. STEWART et J.C. TRINKLE.
« An Implicit Time-Stepping Scheme for Rigid Body Dynamics with Inelastic Collisions and Coulomb Friction ».
International Journal For Numerical Methods in Engineering, 39:2673–2691, 1996.

[34] SUPREME.
« Histoire du billard ».
`http://www.supreme.fr/histoire_billard/`.

[35] J. TRINKLE, J-S. PANG, S. SUDARSKY et G. LO.
« On Dynamic Multi-Rigid-Body Contact Problems with Coulomb Friction ».
ZAMM - Journal of Applied Mathematics and Mecanics, 77:267–279, 1997.

[36] WOLFRAM|ALPHA.
« Wolfram|Alpha ».
`http://www.wolframalpha.com/`.