

Billard

Généré par Doxygen 1.8.1.2

Dimanche Mai 10 2015 23 :19 :58

Table des matières

| | | |
|----------|---|----------|
| 1 | Projet Billard | 1 |
| 1.1 | Présentation | 1 |
| 1.2 | Installation | 1 |
| 1.3 | Dépendances | 1 |
| 1.4 | Développement | 1 |
| 1.4.1 | Premier jet graphique | 2 |
| 1.4.2 | Séparation physique et graphique | 2 |
| 1.4.3 | Ajout des chocs entre boules et contre les murs | 2 |
| 1.4.4 | Implémentation des calculs physiques | 2 |
| 1.4.5 | Gestion de la précision et synchronisaton avec la boucle principale | 2 |
| 1.5 | Physique et Mathématiques | 3 |
| 1.5.1 | Temps de choc entre deux boules | 3 |
| 1.5.2 | Calcul des vitesses et angles après choc entre deux boules | 3 |
| 1.6 | Difficultés rencontrées | 4 |
| 1.6.1 | Conversion entre système polaire et cartésien | 4 |
| 2 | Index des classes | 5 |
| 2.1 | Liste des classes | 5 |
| 3 | Index des fichiers | 7 |
| 3.1 | Liste des fichiers | 7 |
| 4 | Documentation des classes | 9 |
| 4.1 | Référence de la classe Ball | 9 |
| 4.1.1 | Description détaillée | 9 |
| 4.1.2 | Documentation des fonctions membres | 9 |
| 4.1.2.1 | chocBalls | 9 |
| 4.1.2.2 | chocBox | 10 |
| 4.1.2.3 | dtChocBalls | 10 |
| 4.1.2.4 | dtChocBox | 10 |
| 4.1.3 | Documentation des données membres | 10 |
| 4.1.3.1 | id | 10 |

| | | |
|---------|-------------------------------------|----|
| 4.1.3.2 | m | 10 |
| 4.1.3.3 | p | 10 |
| 4.1.3.4 | r | 10 |
| 4.1.3.5 | v | 11 |
| 4.2 | Référence de la classe Billard | 11 |
| 4.2.1 | Description détaillée | 11 |
| 4.2.2 | Documentation des fonctions membres | 11 |
| 4.2.2.1 | main | 11 |
| 4.3 | Référence de la classe Board | 11 |
| 4.3.1 | Description détaillée | 12 |
| 4.3.2 | Documentation des fonctions membres | 12 |
| 4.3.2.1 | make | 12 |
| 4.3.2.2 | render | 12 |
| 4.3.2.3 | update | 12 |
| 4.3.3 | Documentation des données membres | 12 |
| 4.3.3.1 | graphs | 12 |
| 4.3.3.2 | indicators | 12 |
| 4.3.3.3 | nbGraphs | 12 |
| 4.3.3.4 | nbIndicators | 13 |
| 4.3.3.5 | size | 13 |
| 4.4 | Référence de la classe Box | 13 |
| 4.4.1 | Description détaillée | 14 |
| 4.4.2 | Documentation des fonctions membres | 14 |
| 4.4.2.1 | evolve | 14 |
| 4.4.2.2 | make | 14 |
| 4.4.2.3 | pollEvent | 14 |
| 4.4.2.4 | posLine | 14 |
| 4.4.2.5 | posTriangle | 14 |
| 4.4.2.6 | update | 15 |
| 4.4.2.7 | update | 15 |
| 4.4.3 | Documentation des données membres | 15 |
| 4.4.3.1 | ballFocus | 15 |
| 4.4.3.2 | balls | 15 |
| 4.4.3.3 | baseBalls | 15 |
| 4.4.3.4 | dt | 15 |
| 4.4.3.5 | friction | 15 |
| 4.4.3.6 | length | 15 |
| 4.4.3.7 | nbBalls | 15 |
| 4.4.3.8 | nbChoc | 15 |
| 4.4.3.9 | p | 16 |

| | | |
|----------|-------------------------------------|----|
| 4.4.3.10 | r | 16 |
| 4.4.3.11 | rayon | 16 |
| 4.4.3.12 | run | 16 |
| 4.4.3.13 | s | 16 |
| 4.4.3.14 | t | 16 |
| 4.4.3.15 | vmoy | 16 |
| 4.4.3.16 | width | 16 |
| 4.5 | Référence de la classe Button | 16 |
| 4.5.1 | Description détaillée | 17 |
| 4.5.2 | Documentation des fonctions membres | 17 |
| 4.5.2.1 | make | 17 |
| 4.5.2.2 | render | 17 |
| 4.5.2.3 | update | 17 |
| 4.5.3 | Documentation des données membres | 18 |
| 4.5.3.1 | b | 18 |
| 4.5.3.2 | border | 18 |
| 4.5.3.3 | design | 18 |
| 4.5.3.4 | size | 18 |
| 4.5.3.5 | state | 18 |
| 4.5.3.6 | title | 18 |
| 4.6 | Référence de la classe Color | 18 |
| 4.6.1 | Description détaillée | 18 |
| 4.6.2 | Documentation des fonctions membres | 19 |
| 4.6.2.1 | make | 19 |
| 4.6.2.2 | set | 19 |
| 4.6.3 | Documentation des données membres | 19 |
| 4.6.3.1 | b | 19 |
| 4.6.3.2 | r | 19 |
| 4.6.3.3 | v | 19 |
| 4.7 | Référence de la classe Event | 19 |
| 4.7.1 | Description détaillée | 19 |
| 4.7.2 | Documentation des données membres | 20 |
| 4.7.2.1 | b1 | 20 |
| 4.7.2.2 | b2 | 20 |
| 4.7.2.3 | type | 20 |
| 4.8 | Référence de la classe Graph | 20 |
| 4.8.1 | Description détaillée | 20 |
| 4.8.2 | Documentation des fonctions membres | 20 |
| 4.8.2.1 | make | 20 |
| 4.8.2.2 | render | 21 |

| | | |
|----------|-------------------------------------|----|
| 4.8.3 | Documentation des données membres | 21 |
| 4.8.3.1 | graph | 21 |
| 4.8.3.2 | indicator | 21 |
| 4.8.3.3 | moy | 21 |
| 4.8.3.4 | scale | 21 |
| 4.8.3.5 | size | 21 |
| 4.9 | Référence de la classe Indicator | 21 |
| 4.9.1 | Description détaillée | 22 |
| 4.9.2 | Documentation des fonctions membres | 22 |
| 4.9.2.1 | make | 22 |
| 4.9.2.2 | render | 22 |
| 4.9.3 | Documentation des données membres | 22 |
| 4.9.3.1 | name | 22 |
| 4.9.3.2 | precision | 22 |
| 4.9.3.3 | size | 22 |
| 4.9.3.4 | unite | 22 |
| 4.9.3.5 | value | 22 |
| 4.10 | Référence de la classe Input | 22 |
| 4.10.1 | Description détaillée | 23 |
| 4.10.2 | Documentation des fonctions membres | 23 |
| 4.10.2.1 | queuing | 23 |
| 4.10.2.2 | update | 23 |
| 4.10.3 | Documentation des données membres | 23 |
| 4.10.3.1 | buffer | 23 |
| 4.10.3.2 | kb | 24 |
| 4.10.3.3 | out | 24 |
| 4.10.3.4 | queue | 24 |
| 4.11 | Référence de la classe Item | 24 |
| 4.11.1 | Description détaillée | 24 |
| 4.11.2 | Documentation des données membres | 24 |
| 4.11.2.1 | event | 24 |
| 4.11.2.2 | next | 24 |
| 4.12 | Référence de la classe Menu | 24 |
| 4.12.1 | Description détaillée | 25 |
| 4.12.2 | Documentation des fonctions membres | 25 |
| 4.12.2.1 | make | 25 |
| 4.12.2.2 | render | 25 |
| 4.12.2.3 | update | 25 |
| 4.12.3 | Documentation des données membres | 26 |
| 4.12.3.1 | buttons | 26 |

| | | |
|----------|-------------------------------------|----|
| 4.12.3.2 | nbButtons | 26 |
| 4.12.3.3 | size | 26 |
| 4.12.3.4 | title | 26 |
| 4.13 | Référence de la classe Rect | 26 |
| 4.13.1 | Description détaillée | 27 |
| 4.13.2 | Documentation des fonctions membres | 27 |
| 4.13.2.1 | fill | 27 |
| 4.13.2.2 | isIn | 27 |
| 4.13.2.3 | make | 27 |
| 4.13.3 | Documentation des données membres | 27 |
| 4.13.3.1 | h | 27 |
| 4.13.3.2 | w | 27 |
| 4.13.3.3 | x | 27 |
| 4.13.3.4 | y | 27 |
| 4.14 | Référence de la classe RenderBall | 27 |
| 4.14.1 | Description détaillée | 28 |
| 4.14.2 | Documentation des fonctions membres | 28 |
| 4.14.2.1 | make | 28 |
| 4.14.2.2 | render | 28 |
| 4.14.2.3 | update | 28 |
| 4.14.3 | Documentation des données membres | 29 |
| 4.14.3.1 | c | 29 |
| 4.14.3.2 | r | 29 |
| 4.14.3.3 | x | 29 |
| 4.14.3.4 | y | 29 |
| 4.15 | Référence de la classe Stack | 29 |
| 4.15.1 | Description détaillée | 29 |
| 4.15.2 | Documentation des fonctions membres | 30 |
| 4.15.2.1 | clear | 30 |
| 4.15.2.2 | isEmpty | 30 |
| 4.15.2.3 | pull | 30 |
| 4.15.2.4 | push | 30 |
| 4.15.2.5 | size | 30 |
| 4.15.3 | Documentation des données membres | 30 |
| 4.15.3.1 | first | 30 |
| 4.16 | Référence de la classe Vector | 30 |
| 4.16.1 | Description détaillée | 31 |
| 4.16.2 | Documentation des fonctions membres | 31 |
| 4.16.2.1 | atan | 31 |
| 4.16.2.2 | dump | 31 |

| | | |
|-----------|-------------------------------------|-----------|
| 4.16.2.3 | formeCart | 31 |
| 4.16.2.4 | formePol | 32 |
| 4.16.2.5 | module | 32 |
| 4.16.2.6 | newBase | 32 |
| 4.16.2.7 | reverseBase | 32 |
| 4.16.3 | Documentation des données membres | 32 |
| 4.16.3.1 | a | 32 |
| 4.16.3.2 | m | 32 |
| 4.16.3.3 | x | 32 |
| 4.16.3.4 | y | 32 |
| 4.17 | Référence de la classe Window | 33 |
| 4.17.1 | Description détaillée | 33 |
| 4.17.2 | Documentation des fonctions membres | 33 |
| 4.17.2.1 | make | 33 |
| 4.17.2.2 | render | 34 |
| 4.17.2.3 | renderBox | 34 |
| 4.17.2.4 | renderInput | 34 |
| 4.17.2.5 | renderQueue | 34 |
| 4.17.2.6 | update | 34 |
| 4.17.3 | Documentation des données membres | 34 |
| 4.17.3.1 | board | 34 |
| 4.17.3.2 | fps | 34 |
| 4.17.3.3 | height | 34 |
| 4.17.3.4 | menu | 35 |
| 4.17.3.5 | nbBalls | 35 |
| 4.17.3.6 | renderBalls | 35 |
| 4.17.3.7 | scale | 35 |
| 4.17.3.8 | sizeBoard | 35 |
| 4.17.3.9 | sizeMenu | 35 |
| 4.17.3.10 | width | 35 |
| 5 | Documentation des fichiers | 37 |
| 5.1 | Référence du fichier Ball.java | 37 |
| 5.1.1 | Description détaillée | 37 |
| 5.2 | Référence du fichier Billard.java | 37 |
| 5.2.1 | Description détaillée | 37 |
| 5.3 | Référence du fichier Board.java | 38 |
| 5.3.1 | Description détaillée | 38 |
| 5.4 | Référence du fichier Box.java | 38 |
| 5.4.1 | Description détaillée | 38 |

| | | |
|--------|--|----|
| 5.5 | Référence du fichier Button.java | 38 |
| 5.5.1 | Description détaillée | 38 |
| 5.6 | Référence du fichier Color.java | 39 |
| 5.6.1 | Description détaillée | 39 |
| 5.7 | Référence du fichier Input.java | 39 |
| 5.7.1 | Description détaillée | 39 |
| 5.8 | Référence du fichier mainpage.dox | 39 |
| 5.8.1 | Description détaillée | 39 |
| 5.9 | Référence du fichier Menu.java | 40 |
| 5.9.1 | Description détaillée | 40 |
| 5.10 | Référence du fichier Rect.java | 40 |
| 5.10.1 | Description détaillée | 40 |
| 5.11 | Référence du fichier RenderBall.java | 40 |
| 5.11.1 | Description détaillée | 40 |
| 5.12 | Référence du fichier Stack.java | 41 |
| 5.12.1 | Description détaillée | 41 |
| 5.13 | Référence du fichier Vector.java | 41 |
| 5.13.1 | Description détaillée | 41 |
| 5.14 | Référence du fichier Window.java | 41 |
| 5.14.1 | Description détaillée | 41 |

Chapitre 1

Projet Billard

Auteurs

Baptiste Minervini baptiste.minervini@outlook.com
Romain Mekarni romain.mekarni@gmail.com

Date

Licence 1 Semestre 2 : 03/2015 - 05/2015

1.1 Présentation

Ce programme est une simulation physique d'un billard qui prend en compte les chocs entre boules et avec les murs. Il exploite un module physique indépendant du graphique. L'affichage est réglé selon un FPS, et le moteur physique selon un dt (pas de temps). Ces deux variables sont indépendantes et permettent d'observer à plusieurs vitesses l'évolution du système physique.

1.2 Installation

- Clôner le dépôt git à l'url <http://github.com/BMRM/Billard.git>
- `javac *.java` pour compiler
- `java Billard` pour exécuter
- `java -jar Billard.jar` pour exécuter l'archive déjà compilée Pour compiler la bibliothèque :
- `doxygen doxyfile`
- `cd latex && make` (pour la documentation latex)

1.3 Dépendances

Documentation

- `EcranGraphique`
- `Ecran`
- `Clavier`

1.4 Développement

1.4.1 Premier jet graphique

Afin de prendre en main l'environnement, les bibliothèques, le premier jet est basé essentiellement sur l'aspect graphique. A partir de types agrégés, on affiche le billard et les Boules qui se déplacent sans prise en compte des chocs.

1.4.2 Séparation physique et graphique

Le billard est un système physique qui possède ses propres unités de mesure de temps et de distance qui ne sont pas les mêmes que pour le moteur graphique. Par ailleurs, la fonction de l'outil graphique est d'apporter un visuel sur le système. C'est un outil que l'on peut ajouter pour mieux observer la simulation. Il devient donc évident de séparer dans le code les fonctions physiques du graphique et on obtient alors plusieurs classes aux rôles bien définies. Ainsi, la classe `Box` s'occupe du système physique avec la classe `Ball` qui représente physiquement une boule de billard. La partie graphique est gérée par les classes `Window` et `RenderBall`.

1.4.3 Ajout des chocs entre boules et contre les murs

A l'aide de conditions, on change les vitesses et les angles des boules mais très vite on observe les limites de ce modèle trop simpliste. Premièrement, les chocs ne sont pas détectés au bon moment précis (parfois trop tôt, ou trop tard) ce qui détruit progressivement le réalisme de la simulation. Ensuite, les angles ne sont pas justes car ils dépendent de plusieurs paramètres et nécessitent des calculs plus compliqués. Enfin, le modèle ne prend pas en compte plusieurs chocs simultanés (très visible au début de la simulation quand les boules sont disposées en triangle).

1.4.4 Implémentation des calculs physiques

Avec l'aide d'internet, on refait les calculs physiques et on les implémente dans le programme (voir Physique et Mathématiques). Les vitesses sont correctes mais les angles d'après choc entre boules ne sont pas toujours exacts : les directions s'en rapprochent, mais souvent les sens sont inversés. Le programme bug assez souvent car il détecte des chocs à des temps très faibles (trop faible pour faire avancer les boules) ce qui provoque une boucle infinie.

1.4.5 Gestion de la précision et synchronisation avec la boucle principale

Pour régler ce problème de précision, on a choisi d'utiliser la classe `BigDecimal` fournie par Java et qui nous permet de représenter des nombres décimaux selon un contexte mathématique (nombre de chiffres significatifs et méthode d'arrondissement) On remarque qu'avec une précision de moins de 5, certains chocs sont ignorés, alors qu'au dessus de 10 ou 15, la boucle infinie peut éventuellement faire planter le programme. A 8, on a relevé aucun bug apparent.

Pour faire évoluer le système physique, on utilise l'algorithme suivant :

- Pour chaque boules
 - Pour chaque boules
 - Calculer le temps de prochain choc entre les boules
 - S'il est positif et plus proche ou égal que le choc le plus proche
 - S'il est plus petit
 - Vider la pile des évènements
 - Mémoriser ce choc comme étant le plus proche
 - Ajouter le choc à la pile des évènements
 - Calculer le temps de prochain choc entre la boule et les murs
 - S'il est positif et plus proche que le choc le plus proche
 - S'il est plus petit
 - Vider la pile des évènements
 - Mémoriser ce choc comme étant le plus proche
 - Ajouter le choc à la pile des évènements

- Déplacer les boules au choc le plus proche (dt si inexistant)
- Décharger la pile et appliquer tous les évènements
- Renvoyer le temps de choc le plus proche (dt si inexistant)

Cet algorithme est appelé plusieurs fois tant qu'il n'a pas atteint le dt imposé par le programme (la boucle principale). Cela permet de garder notre système physique cohérent même si l'affichage graphique varie en fonction des performances de l'ordinateur (un dt à 500 ou 1000 réduit considérablement le FPS car les calculs sont très conséquents).

1.5 Physique et Mathématiques

1.5.1 Temps de choc entre deux boules

$$a.dt^2 + b.dt + c = 0$$

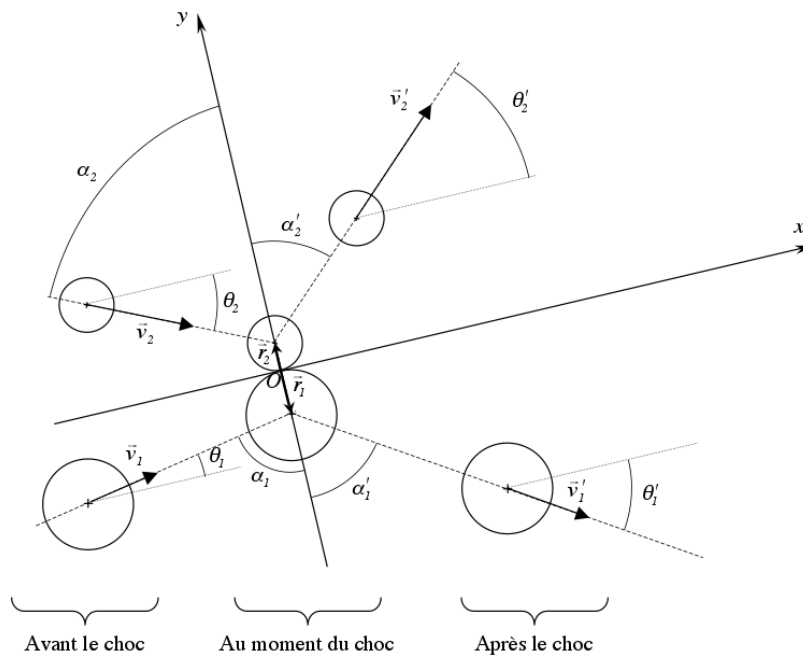
$$\begin{cases} a &= (b_{1,v,x} - b_{2,v,x})^2 + (b_{1,v,y} - b_{2,v,y})^2 \\ b &= 2(b_{1,v,x} - b_{2,v,x})(b_{1,p,x} - b_{2,p,x}) + 2(b_{1,v,y} - b_{2,v,y})(b_{1,p,y} - b_{2,p,y}) \\ c &= (b_{1,p,x} - b_{2,p,x})^2 + (b_{1,p,y} - b_{2,p,y})^2 - (b_{1,r} + b_{2,r})^2 \end{cases}$$

1.5.2 Calcul des vitesses et angles après choc entre deux boules

Etude réalisée grâce au document de [Pascal Rebetez](#)

Afin de calculer les nouvelles vitesses des boules après un choc, il est nécessaire de changer de repère pour simplifier les calculs. On utilise le repère suivant : On limite cette étude au cas où les boules ne sont pas en rotation

FIGURE 1.1 – Choc élastique en 2 dimensions.



autour de leur centre, elles ne sont animées que d'un mouvement de translation. Pour déterminer les 4 nouveaux paramètres (norme et orientation des vitesses de chaque boules), on dispose de trois principes de conservation : le principe de conservation de la quantité de mouvement, le principe de conservation de l'énergie mécanique (le choc étant supposé élastique, l'énergie cinétique est conservée) et le principe de conservation du moment cinétique. De

ces équations, on obtient les solutions suivantes :

$$\begin{aligned}\theta'_1 &= \arctan\left(\frac{m_1 - m_2}{m_1 + m_2} \tan \theta_1 + \frac{2m_2}{m_1 + m_2} \frac{v_2}{v_1} \frac{\sin \theta_2}{\cos \theta_1}\right) \\ v'_1 &= \sqrt{\left(\frac{m_1 - m_2}{m_1 + m_2} v_1 \sin \theta_1 + \frac{2m_2}{m_1 + m_2} v_2 \sin \theta_2\right)^2 + (v_1 \cos \theta_1)^2} \\ \theta'_2 &= \arctan\left(\frac{m_2 - m_1}{m_1 + m_2} \tan \theta_2 + \frac{2m_1}{m_1 + m_2} \frac{v_1}{v_2} \frac{\sin \theta_1}{\cos \theta_2}\right) \\ v'_2 &= \sqrt{\left(\frac{m_2 - m_1}{m_1 + m_2} v_1 \sin \theta_2 + \frac{2m_1}{m_1 + m_2} v_1 \sin \theta_1\right)^2 + (v_2 \cos \theta_2)^2}\end{aligned}$$

1.6 Difficultés rencontrées

1.6.1 Conversion entre système polaire et cartésien

à partir de [Wikipédia](#)

Chapitre 2

Index des classes

2.1 Liste des classes

Liste des classes, structures, unions et interfaces avec une brève description :

| | | |
|----------------------------|--|----|
| Ball | Gestion des Boules | 9 |
| Billard | Classe principale du programme | 11 |
| Board | Bandeau d'affichage des grandeurs physique du système | 11 |
| Box | Moteur physique | 13 |
| Button | Boutton du Menu | 16 |
| Color | Classe représentant une couleur | 18 |
| Event | Représente un évènement de type choc | 19 |
| Graph | Mesure graphique d'une grandeur physique du système physique | 20 |
| Indicator | Mesure numérique d'une grandeur physique du système physique | 21 |
| Input | Gestion des interactions utilisateur | 22 |
| Item | Element de Stack qui contient un Event | 24 |
| Menu | Menu interactif à disposition de l'utilisateur | 24 |
| Rect | Classe représentant un rectangle soit sa taille et sa position | 26 |
| RenderBall | Rendu des boules | 27 |
| Stack | Conteneur de Item de type pile | 29 |
| Vector | Manipulation des vecteurs mathématiques | 30 |
| Window | Moteur graphique | 33 |

Chapitre 3

Index des fichiers

3.1 Liste des fichiers

Liste de tous les fichiers avec une brève description :

| | | |
|---------------------------------|---|----|
| Ball.java | Représentation physique d'une boule et interaction avec les autres objets | 37 |
| Billard.java | Fonction main, instanciation des modules principaux | 37 |
| Board.java | Bandeau de mesures du système physique | 38 |
| Box.java | Moteur physique - Simulation du Billard | 38 |
| Button.java | Boutton du Menu | 38 |
| Color.java | Gestion des couleurs pour l'API EcranGraphique | 39 |
| Input.java | Gestion des interactions utilisateur | 39 |
| Menu.java | Menu interactif à disposition de l'utilisateur | 40 |
| Rect.java | Gestion des formes rectangulaires | 40 |
| RenderBall.java | Rendu des boules | 40 |
| Stack.java | Structure de données de type pile | 41 |
| Vector.java | Manipulation des vecteurs mathématiques pour la physique | 41 |
| Window.java | Moteur graphique | 41 |

Chapitre 4

Documentation des classes

4.1 Référence de la classe Ball

Gestion des Boules.

Fonctions statiques de paquetage

- static BigDecimal `dtChocBox` (`Box` box, `Ball` b, BigDecimal dt)
Détermine l'instant ou aura lieu le choc entre 1 boule et un mur, si il existe.
- static void `chocBox` (`Box` box, `Ball` b)
Calcul les nouveaux angles après un choc Boule/Mur.
- static BigDecimal `dtChocBalls` (`Box` box, `Ball` b1, `Ball` b2, BigDecimal dt)
Calcul l'instant de choc entre 2 boules.
- static void `chocBalls` (`Ball` b1, `Ball` b2)
Calcul les nouveaux angles et vitesses aux Boules apres un choc Boule/Boule.

Attributs de paquetage

- int `id`
Identifiant.
- double `r`
Rayon.
- `Vector` `p` = new `Vector`()
Vecteur position.
- `Vector` `v` = new `Vector`()
Vecteur vitesse.
- double `m`
Masse.

4.1.1 Description détaillée

Gestion des Boules.

Auteurs

Baptiste Minervini
Romain Mekarni

4.1.2 Documentation des fonctions membres

4.1.2.1 static void Ball.chocBalls (`Ball` b1, `Ball` b2) [static], [package]

Calcul les nouveaux angles et vitesses aux Boules apres un choc Boule/Boule.

Auteur

Romain Mekarni

4.1.2.2 static void Ball.chocBox (Box box, Ball b) [static], [package]

Calcul les nouveaux angles après un choc Boule/Mur.

Auteur

Romain Mekarni

4.1.2.3 static BigDecimal Ball.dtChocBalls (Box box, Ball b1, Ball b2, BigDecimal dt) [static], [package]

Calcul l'instant de choc entre 2 boules.

Renvoie

L'instant du choc (-1 si pas de choc dans]0 ;dt])

Auteur

Romain Mekarni

4.1.2.4 static BigDecimal Ball.dtChocBox (Box box, Ball b, BigDecimal dt) [static], [package]

Détermine l'instant ou aura lieu le choc entre 1 boule et un mur, si il existe.

Renvoie

L'instant du choc (-1 si pas de choc dans]0 ;dt])

Auteur

Romain Mekarni

4.1.3 Documentation des données membres**4.1.3.1 int Ball.id [package]**

Identifiant.

4.1.3.2 double Ball.m [package]

Masse.

4.1.3.3 Vector Ball.p = new Vector() [package]

Vecteur position.

4.1.3.4 double Ball.r [package]

Rayon.

4.1.3.5 Vector `Ball.v = new Vector()` [package]

Vecteur vitesse.

La documentation de cette classe a été générée à partir du fichier suivant :

– [Ball.java](#)

4.2 Référence de la classe Billard

Classe principale du programme.

Fonctions membres publiques statiques

– static void [main](#) (String[] args)

4.2.1 Description détaillée

Classe principale du programme.

Auteurs

Baptiste Minervini
Romain Mekarni

4.2.2 Documentation des fonctions membres

4.2.2.1 static void `Billard.main (String[] args)` [static]

La documentation de cette classe a été générée à partir du fichier suivant :

– [Billard.java](#)

4.3 Référence de la classe Board

Bandeau d'affichage des grandeurs physique du système.

Fonctions statiques de paquetage

- static [Board](#) [make](#) ([Rect](#) size)
Constructeur.
- static void [update](#) ([Board](#) b, [Box](#) box, [Window](#) win)
Mise à jour des informations du bandeau.
- static void [render](#) ([Board](#) b)
Rendu du bandeau dans [Menu](#).

Attributs de paquetage

- int [nbIndicators](#) = 8
Nombre de [Indicator](#) dans le bandeau.
- int [nbGraphs](#) = 2
Nombre de graphiques.
- [Rect](#) size
Position et taille.
- [Indicator](#) indicators []
Tableau de [Indicator](#).
- [Graph](#) graphs []

Tableau de [Graph](#).

4.3.1 Description détaillée

Bandeau d'affichage des grandeurs physique du système.

Auteur

Romain Mekarni

4.3.2 Documentation des fonctions membres

4.3.2.1 `static Board Board.make (Rect size) [static], [package]`

Constructeur.

Paramètres

| | |
|-------------|---|
| <i>size</i> | Taille et position du Board |
|-------------|---|

Auteur

Romain Mekarni

4.3.2.2 `static void Board.render (Board b) [static], [package]`

Rendu du bandeau dans [Menu](#).

Auteur

Romain Mekarni

4.3.2.3 `static void Board.update (Board b, Box box, Window win) [static], [package]`

Mise à jour des informations du bandeau.

Auteur

Romain Mekarni

4.3.3 Documentation des données membres

4.3.3.1 `Graph Board.graphs[] [package]`

Tableau de [Graph](#).

4.3.3.2 `Indicator Board.indicators[] [package]`

Tableau de [Indicator](#).

4.3.3.3 `int Board.nbGraphs = 2 [package]`

Nombre de graphiques.

4.3.3.4 int Board.nbIndicators = 8 [package]

Nombre de [Indicator](#) dans le bandeau.

4.3.3.5 Rect Board.size [package]

Position et taille.

La documentation de cette classe a été générée à partir du fichier suivant :

– [Board.java](#)

4.4 Référence de la classe Box

Moteur physique.

Fonctions statiques de paquetage

- static [Box](#) [make](#) ()
Constructeur de [Box](#).
- static void [update](#) ([Box](#) box)
Fait évoluer le système physique de [Box](#).dt.
- static double [update](#) ([Box](#) box, double dt)
évolue jusqu'au prochain état avant dt.
- static void [pollEvent](#) ([Box](#) box, [Stack](#) stack)
Applique les événements mémorisés dans un changement d'état.
- static void [evolve](#) ([Box](#) box, [Ball](#) b, double dt)
Déplace la boule b d'un temps dt avec une friction.
- static void [posLine](#) ([Box](#) box)
Positionne les boules en ligne.
- static void [posTriangle](#) ([Box](#) box)
Positionne les boules en triangle.

Attributs de paquetage

- double dt = 0.05
Pas de temps.
- double t = 0
Temps absolu.
- double rayon = 0.03
Rayon des boules.
- double width = 1.27
Largeur du billard.
- double length = 2.54
Longueur du billard.
- int s = 8
Précision des calculs physiques.
- RoundingMode r = RoundingMode.HALF_UP
Arrondissement à 1 si 0.5.
- MathContext p = new MathContext(s, r)
Contexte mathématique d'arrondissement.
- int nbChoc = 0
Compteur nombre de choc par dt.
- double vmoy = 0
Compteur vitesse moyenne du système.
- [Ball](#) ballFocus
Boule ciblée pour interaction.
- boolean run = false
Flag d'exécution du système physique.
- double friction = 0.1
- int baseBalls = 5
Taille du triangle de boules.
- int nbBalls

- *Nombre de boules.*
`Ball balls []`
Tableau de boules.

4.4.1 Description détaillée

Moteur physique.

Auteurs

Baptiste Minervini
Romain Mekarni

4.4.2 Documentation des fonctions membres

4.4.2.1 `static void Box.evolve (Box box, Ball b, double dt) [static], [package]`

Déplace la boule b d'un temps dt avec une friction.

Auteurs

Baptiste Minervini
Romain Mekarni

4.4.2.2 `static Box Box.make () [static], [package]`

Constructeur de `Box`.

Auteurs

Baptiste Minervini
Romain Mekarni

4.4.2.3 `static void Box.pollEvent (Box box, Stack stack) [static], [package]`

Applique les événements mémorisés dans un changement d'état.

Auteur

Romain Mekarni

4.4.2.4 `static void Box.posLine (Box box) [static], [package]`

Positionne les boules en ligne.

Auteur

Baptiste Minervini

4.4.2.5 `static void Box.posTriangle (Box box) [static], [package]`

Positionne les boules en triangle.

Auteur

Baptiste Minervini

4.4.2.6 `static void Box.update (Box box) [static], [package]`

Fait évoluer le système physique de [Box.dt](#).

Auteur

Romain Mekarni

4.4.2.7 `static double Box.update (Box box, double dt) [static], [package]`

évolue jusqu'au prochain état avant dt.

Renvoie

temps t du prochain état du système

Auteur

Romain Mekarni

4.4.3 Documentation des données membres

4.4.3.1 `Ball Box.ballFocus [package]`

Boule ciblée pour interaction.

4.4.3.2 `Ball Box.balls[] [package]`

Tableau de boules.

4.4.3.3 `int Box.baseBalls = 5 [package]`

Taille du triangle de boules.

4.4.3.4 `double Box.dt = 0.05 [package]`

Pas de temps.

4.4.3.5 `double Box.friction = 0.1 [package]`

4.4.3.6 `double Box.length = 2.54 [package]`

Longueur du billard.

4.4.3.7 `int Box.nbBalls [package]`

Nombre de boules.

4.4.3.8 `int Box.nbChoc = 0 [package]`

Compteur nombre de choc par dt.

4.4.3.9 `MathContext Box.p = new MathContext(s, r)` [package]

Contexte mathématique d'arrondissement.

4.4.3.10 `RoundingMode Box.r = RoundingMode.HALF_UP` [package]

Arrondissement à 1 si 0.5.

4.4.3.11 `double Box.rayon = 0.03` [package]

Rayon des boules.

4.4.3.12 `boolean Box.run = false` [package]

Flag d'exécution du système physique.

4.4.3.13 `int Box.s = 8` [package]

Précision des calculs physiques.

4.4.3.14 `double Box.t = 0` [package]

Temps absolu.

4.4.3.15 `double Box.vmoy = 0` [package]

Compteur vitesse moyenne du système.

4.4.3.16 `double Box.width = 1.27` [package]

Largeur du billard.

La documentation de cette classe a été générée à partir du fichier suivant :

– [Box.java](#)

4.5 Référence de la classe Button

Boutton du [Menu](#).

Fonctions statiques de paquetage

- static [Button make](#) (String[] [title](#), [Rect size](#))
Constructeur.
- static boolean [update](#) ([Button b](#))
Détermine l'état du bouton.
- static void [render](#) ([Button b](#))
Rendu dans [Window](#).

Attributs de paquetage

- int **b** = 3
Taille bordure.
- String **title** []
Libellés bouton.
- Color **design** []
Couleurs du boutons.
- Rect **size**
Taille et position.
- Rect **border**
Format de la bordure.
- int **state** = 0
Etats du bouton.

4.5.1 Description détaillée

Boutton du [Menu](#).

Auteurs

Baptiste Minervini
Romain Mekarni

4.5.2 Documentation des fonctions membres

4.5.2.1 static Button Button.make (String[] **title**, Rect **size**) [static],[package]

Constructeur.

Paramètres

| | |
|--------------|---|
| <i>title</i> | Tableau de chaines contenant le libellé du bouton |
| <i>size</i> | Taille et position du bouton |

Auteurs

Baptiste Minervini
Romain Mekarni

4.5.2.2 static void Button.render (Button **b**) [static],[package]

Rendu dans [Window](#).

Auteurs

Baptiste Minervini
Romain Mekarni

4.5.2.3 static boolean Button.update (Button **b**) [static],[package]

Détermine l'état du bouton.

Renvoie

Clique enfoncé sur bouton = true

Auteurs

Baptiste Minervini
Romain Mekarni

4.5.3 Documentation des données membres

4.5.3.1 `int Button.b = 3` [package]

Taille bordure.

4.5.3.2 `Rect Button.border` [package]

Format de la bordure.

4.5.3.3 `Color Button.design[]` [package]

Couleurs du boutons.

4.5.3.4 `Rect Button.size` [package]

Taille et position.

4.5.3.5 `int Button.state = 0` [package]

Etats du bouton.

4.5.3.6 `String Button.title[]` [package]

Libellés bouton.

La documentation de cette classe a été générée à partir du fichier suivant :

– [Button.java](#)

4.6 Référence de la classe Color

Classe représentant une couleur.

Fonctions statiques de paquetage

- static `Color make` (int `r`, int `v`, int `b`)
Constructeur.
- static void `set` (`Color` `c`)
Paramètre la couleur de EcranGraphique.

Attributs de paquetage

- int `r`
Rouge.
- int `v`
Vert.
- int `b`
Bleu.

4.6.1 Description détaillée

Classe représentant une couleur.

Auteur

Baptiste Minervini

4.6.2 Documentation des fonctions membres

4.6.2.1 `static Color Color.make (int r, int v, int b)` `[static]`, `[package]`

Constructeur.

Auteur

Baptiste Minervini

4.6.2.2 `static void Color.set (Color c)` `[static]`, `[package]`

Paramètre la couleur de EcranGraphique.

Auteur

Romain Mekarni

4.6.3 Documentation des données membres

4.6.3.1 `int Color.b` `[package]`

Bleu.

4.6.3.2 `int Color.r` `[package]`

Rouge.

4.6.3.3 `int Color.v` `[package]`

Vert.

La documentation de cette classe a été générée à partir du fichier suivant :

– [Color.java](#)

4.7 Référence de la classe Event

Représente un évènement de type choc.

Attributs de paquetage

- `int type`
Type de choc ; 0 boule/boule ; 1 boule/mur.
- `Ball b1`
- `Ball b2`

4.7.1 Description détaillée

Représente un évènement de type choc.

Auteur

Romain Mekarni

4.7.2 Documentation des données membres

4.7.2.1 **Ball Event.b1** [package]

4.7.2.2 **Ball Event.b2** [package]

4.7.2.3 **int Event.type** [package]

Type de choc ; 0 boule/boule ; 1 boule/mur.

La documentation de cette classe a été générée à partir du fichier suivant :

– [Stack.java](#)

4.8 Référence de la classe Graph

Mesure graphique d'une grandeur physique du système physique.

Fonctions statiques de paquetage

- static [Graph make](#) (String name, String unite, int precision, int [scale](#), [Rect size](#))
Constructeur.
- static void [render](#) ([Graph g](#))
Rendu graphique dans [Board](#).

Attributs de paquetage

- [Indicator indicator](#)
Grandeur physique mesurée.
- int [scale](#)
Echelle d'affichage.
- double [moy](#) = 0
Valeur moyenne.
- [Rect size](#)
Taille et zone d'affichage.
- int[][] [graph](#)
Tableau d'entier représentant le graphique.

4.8.1 Description détaillée

Mesure graphique d'une grandeur physique du système physique.

Auteur

Romain Mekarni

4.8.2 Documentation des fonctions membres

4.8.2.1 **static Graph Graph.make (String name, String unite, int precision, int scale, Rect size)** [static],
[package]

Constructeur.

Paramètres

| | |
|--------------|----------------------------|
| <i>scale</i> | Echelle |
| <i>size</i> | Taille et zone d'affichage |

Auteur

Romain Mekarni

4.8.2.2 `static void Graph.render (Graph g)` [static], [package]

Rendu graphique dans [Board](#).

Auteur

Romain Mekarni

4.8.3 Documentation des données membres

4.8.3.1 `int [][] Graph.graph` [package]

Tableau d'entier représentant le graphique.

4.8.3.2 `Indicator Graph.indicator` [package]

Grandeur physique mesurée.

4.8.3.3 `double Graph.moy = 0` [package]

Valeur moyenne.

4.8.3.4 `int Graph.scale` [package]

Echelle d'affichage.

4.8.3.5 `Rect Graph.size` [package]

Taille et zone d'affichage.

La documentation de cette classe a été générée à partir du fichier suivant :

– [Board.java](#)

4.9 Référence de la classe Indicator

Mesure numérique d'une grandeur physique du système physique.

Fonctions statiques de paquetage

- static [Indicator make](#) (String [name](#), String [unite](#), int [precision](#), [Rect size](#))
Constructor.
- static void [render](#) ([Indicator i](#))
Rendu de l'indicateur dans [Board](#).

Attributs de paquetage

- String [name](#)
- String [unite](#)
- double [value](#) = 0
- int [precision](#) = 2
- Rect [size](#)

Position dans le bandeau.

4.9.1 Description détaillée

Mesure numérique d'une grandeur physique du système physique.

Auteur

Romain Mekarni

4.9.2 Documentation des fonctions membres

4.9.2.1 `static Indicator Indicator.make (String name, String unite, int precision, Rect size)` [static], [package]

Constructor.

Auteur

Romain Mekarni

4.9.2.2 `static void Indicator.render (Indicator i)` [static], [package]

Rendu de l'indicateur dans [Board](#).

Auteur

Romain Mekarni

4.9.3 Documentation des données membres

4.9.3.1 `String Indicator.name` [package]

4.9.3.2 `int Indicator.precision = 2` [package]

4.9.3.3 `Rect Indicator.size` [package]

Position dans le bandeau.

4.9.3.4 `String Indicator.unite` [package]

4.9.3.5 `double Indicator.value = 0` [package]

La documentation de cette classe a été générée à partir du fichier suivant :

- [Board.java](#)

4.10 Référence de la classe Input

Gestion des interactions utilisateur.

Fonctions statiques de paquetage

- static void `update` (`Input` input, `Box` box, `Window` win)
Récupère l'interaction utilisateur si active.
- static void `queuing` (`Input` input, `Box` box, `Window` win)
Gestion de la queue de billard.

Attributs de paquetage

- boolean `kb` = false
Flag d'utilisation du clavier.
- boolean `queue` = false
Flag d'utilisation de la queue de billard.
- String `buffer` = ""
- int `out`

4.10.1 Description détaillée

Gestion des interactions utilisateur.

Auteurs

Baptiste Minervini
Romain Mekarni

4.10.2 Documentation des fonctions membres

4.10.2.1 static void Input.queuing (Input input, Box box, Window win) [static], [package]

Gestion de la queue de billard.

Paramètres

| | |
|------------|---|
| <i>box</i> | La queue agit sur la boule ciblée et relance le moteur physique |
| <i>win</i> | Format d'affichage pour l'échelle de la queue de billard |

Auteurs

Baptiste Minervini
Romain Mekarni

4.10.2.2 static void Input.update (Input input, Box box, Window win) [static], [package]

Récupère l'interaction utilisateur si active.

Paramètres

| | |
|------------|--|
| <i>box</i> | La queue de billard agit sur la boule ciblée |
| <i>win</i> | format d'affichage et libellé des boutons |

Auteurs

Baptiste Minervini
Romain Mekarni

4.10.3 Documentation des données membres

4.10.3.1 String Input.buffer = "" [package]

4.10.3.2 `boolean Input.kb = false` [package]

Flag d'utilisation du clavier.

4.10.3.3 `int Input.out` [package]

4.10.3.4 `boolean Input.queue = false` [package]

Flag d'utilisation de la queue de billard.

La documentation de cette classe a été générée à partir du fichier suivant :

– [Input.java](#)

4.11 Référence de la classe Item

Element de [Stack](#) qui contient un [Event](#).

Attributs de paquetage

- [Event event](#)
Evènement mémorisé
- [Item next](#) = null
Référence sur l'élément suivant de la pile.

4.11.1 Description détaillée

Element de [Stack](#) qui contient un [Event](#).

Auteur

Romain Mekarni

4.11.2 Documentation des données membres

4.11.2.1 `Event Item.event` [package]

Evènement mémorisé

4.11.2.2 `Item Item.next = null` [package]

Référence sur l'élément suivant de la pile.

La documentation de cette classe a été générée à partir du fichier suivant :

– [Stack.java](#)

4.12 Référence de la classe Menu

menu interactif à disposition de l'utilisateur

Fonctions statiques de paquetage

- static [Menu make](#) ([Window win](#), [Box box](#), [Rect size](#))
Constructeur.

- static int `update` (Menu menu, Box box, Window win)
Actualise les libellés des boutons et prise en compte des clics.
- static void `render` (Menu menu)
Rendu du menu dans Window.

Attributs de paquetage

- int `nbButtons` = 11
Nombre de Button du Menu.
- String `title` [][]
Libellés des boutons.
- Button `buttons` []
Tableau de boutons.
- Rect `size`
Taille et position du menu dans Window.

4.12.1 Description détaillée

menu interactif à disposition de l'utilisateur

Auteurs

Baptiste Minervini
Romain Mekarni

4.12.2 Documentation des fonctions membres

4.12.2.1 static Menu Menu.make (Window win, Box box, Rect size) [static], [package]

Constructeur.

Paramètres

| | |
|-------------|----------------------------|
| <i>size</i> | Taille et position du Menu |
|-------------|----------------------------|

Auteurs

Baptiste Minervini
Romain Mekarni

4.12.2.2 static void Menu.render (Menu menu) [static], [package]

Rendu du menu dans Window.

Auteurs

Baptiste Minervini
Romain Mekarni

4.12.2.3 static int Menu.update (Menu menu, Box box, Window win) [static], [package]

Actualise les libellés des boutons et prise en compte des clics.

Renvoie

id du bouton cliqué, -1 si aucun

Auteurs

Baptiste Minervini
Romain Mekarni

4.12.3 Documentation des données membres

4.12.3.1 `Button Menu.buttons[]` [package]

Tableau de boutons.

4.12.3.2 `int Menu.nbButtons = 11` [package]

Nombre de `Button` du `Menu`.

4.12.3.3 `Rect Menu.size` [package]

Taille et position du menu dans `Window`.

4.12.3.4 `String Menu.title[][]` [package]

Valeur initiale :

```
{
    {"Billard classique", ""},
    {"Boules en ligne", ""},
    {"Jeu Manuel", ""},
    {"Demarrer/Pause", ""},
    {"dt : ", ""},
    {"Precision : ", ""},
    {"FPS : ", ""},
    {"Rayon : ", ""},
    {"Boule : ", ""},
    {"Frottement : ", ""},
    {"Quitter", ""}}
```

Libellés des boutons.

La documentation de cette classe a été générée à partir du fichier suivant :

– `Menu.java`

4.13 Référence de la classe Rect

Classe représentant un rectangle soit sa taille et sa position.

Fonctions statiques de paquetage

- static `Rect make` (int `x`, int `y`, int `w`, int `h`)
Constructeur.
- static void `fill` (`Rect` rect)
Rempli un rectangle avec `EcranGraphique`.
- static boolean `isIn` (`Rect` rect, int `x`, int `y`)
Vérifie si le point (x,y) est dans le rectangle rect.

Attributs de paquetage

- int `x`
Position x.
- int `y`
Position y.
- int `w`
Longueur.
- int `h`
Largeur.

4.13.1 Description détaillée

Classe représentant un rectangle soit sa taille et sa position.

Auteur

Romain Mekarni

4.13.2 Documentation des fonctions membres

4.13.2.1 `static void Rect.fill (Rect rect) [static], [package]`

Rempli un rectangle avec EcranGraphique.

4.13.2.2 `static boolean Rect.isIn (Rect rect, int x, int y) [static], [package]`

Vérifie si le point (x,y) est dans le rectangle rect.

Auteur

Romain Mekarni

4.13.2.3 `static Rect Rect.make (int x, int y, int w, int h) [static], [package]`

Constructeur.

Auteur

Romain Mekarni

4.13.3 Documentation des données membres

4.13.3.1 `int Rect.h [package]`

Largeur.

4.13.3.2 `int Rect.w [package]`

Longueur.

4.13.3.3 `int Rect.x [package]`

Position x.

4.13.3.4 `int Rect.y [package]`

Position y.

La documentation de cette classe a été générée à partir du fichier suivant :

– [Rect.java](#)

4.14 Référence de la classe RenderBall

Rendu des boules.

Fonctions statiques de paquetage

- static `RenderBall[] make (Box box)`
Constructeur.
- static void `update (Window win, RenderBall rB, Ball b)`
Synchronisation graphique à partir du moteur physique.
- static void `render (RenderBall rB)`
Rendu d'une boule.

Attributs de paquetage

- int `r`
Rayon en pixels.
- int `x`
Position x.
- int `y`
Position y.
- `Color c = new Color()`
Couleur d'affichage.

4.14.1 Description détaillée

Rendu des boules.

Auteurs

Baptiste Minervini
Romain Mekarni

4.14.2 Documentation des fonctions membres

4.14.2.1 static `RenderBall[] RenderBall.make (Box box)` [static], [package]

Constructeur.

Auteurs

Baptiste Minervini
Romain Mekarni

4.14.2.2 static void `RenderBall.render (RenderBall rB)` [static], [package]

Rendu d'une boule.

Auteurs

Baptiste Minervini
Romain Mekarni

4.14.2.3 static void `RenderBall.update (Window win, RenderBall rB, Ball b)` [static], [package]

Synchronisation graphique à partir du moteur physique.

Auteurs

Baptiste Minervini
Romain Mekarni

4.14.3 Documentation des données membres

4.14.3.1 Color RenderBall.c = new Color() [package]

Couleur d'affichage.

4.14.3.2 int RenderBall.r [package]

Rayon en pixels.

4.14.3.3 int RenderBall.x [package]

Position x.

4.14.3.4 int RenderBall.y [package]

Position y.

La documentation de cette classe a été générée à partir du fichier suivant :

– [RenderBall.java](#)

4.15 Référence de la classe Stack

Conteneur de [Item](#) de type pile.

Fonctions statiques de paquetage

- static [Stack push](#) ([Stack](#) stack, int type, [Ball](#) b1, [Ball](#) b2)
Ajoute un élément à la pile.
- static [Stack pull](#) ([Stack](#) stack)
Retire un élément.
- static void [clear](#) ([Stack](#) stack)
Vide la pile.
- static boolean [isEmpty](#) ([Stack](#) stack)
Vérifie si la pile est pleine.
- static int [size](#) ([Stack](#) stack)
Renvoie la taille de la pile.

Attributs de paquetage

- [Item first](#) = null
Premier élément de la pile.

4.15.1 Description détaillée

Conteneur de [Item](#) de type pile.

Auteur

Romain Mekarni

4.15.2 Documentation des fonctions membres

4.15.2.1 `static void Stack.clear (Stack stack) [static], [package]`

Vide la pile.

Auteur

Romain Mekarni

4.15.2.2 `static boolean Stack.isEmpty (Stack stack) [static], [package]`

Vérifie si la pile est pleine.

4.15.2.3 `static Stack Stack.pull (Stack stack) [static], [package]`

Retire un élément.

Auteur

Romain Mekarni

4.15.2.4 `static Stack Stack.push (Stack stack, int type, Ball b1, Ball b2) [static], [package]`

Ajoute un élément à la pile.

Paramètres

| | |
|-------------|--|
| <i>type</i> | 0 pour un choc boule/boule et 1 pour boule/mur |
|-------------|--|

Auteur

Romain Mekarni

4.15.2.5 `static int Stack.size (Stack stack) [static], [package]`

Renvoie la taille de la pile.

Auteur

Romain Mekarni

4.15.3 Documentation des données membres

4.15.3.1 `Item Stack.first = null [package]`

Premier élément de la pile.

La documentation de cette classe a été générée à partir du fichier suivant :

– [Stack.java](#)

4.16 Référence de la classe Vector

Manipulation des vecteurs mathématiques.

Fonctions statiques de paquetage

- static double `module (Vector v)`
Calcule le module d'un vecteur.
- static double `atan (double y, double x)`
Calcule atan.
- static `Vector formePol (Vector v)`
Conversion coordonnées cartésiennes -> polaires.
- static `Vector formeCart (Vector v)`
Conversion coordonnées polaires -> cartésiennes.
- static `Vector newBase (Vector v, double alpha)`
Change le vecteur de base.
- static `Vector reverseBase (Vector v, double alpha)`
Change le vecteur de base (retour vers initial)
- static void `dump (Vector v)`
Dump vecteur.

Attributs de paquetage

- double `x`
Position x.
- double `y`
Position y.
- double `m`
Module.
- double `a`
Angle en radian.

4.16.1 Description détaillée

Manipulation des vecteurs mathématiques.

Auteur

Romain Mekarni

4.16.2 Documentation des fonctions membres

4.16.2.1 static double Vector.atan (double y, double x) [static], [package]

Calcule atan.

Auteur

Romain Mekarni

4.16.2.2 static void Vector.dump (Vector v) [static], [package]

Dump vecteur.

4.16.2.3 static Vector Vector.formeCart (Vector v) [static], [package]

Conversion coordonnées polaires -> cartésiennes.

Auteur

Romain Mekarni

4.16.2.4 `static Vector Vector.formePol (Vector v)` [static], [package]

Conversion coordonnées cartésiennes -> polaires.

Auteur

Romain Mekarni

4.16.2.5 `static double Vector.module (Vector v)` [static], [package]

Calcule le module d'un vecteur.

Auteur

Romain Mekarni

4.16.2.6 `static Vector Vector.newBase (Vector v, double alpha)` [static], [package]

Change le vecteur de base.

Auteur

Romain Mekarni

4.16.2.7 `static Vector Vector.reverseBase (Vector v, double alpha)` [static], [package]

Change le vecteur de base (retour vers initial)

Auteur

Romain Mekarni

4.16.3 Documentation des données membres

4.16.3.1 `double Vector.a` [package]

Angle en radian.

4.16.3.2 `double Vector.m` [package]

Module.

4.16.3.3 `double Vector.x` [package]

Position x.

4.16.3.4 `double Vector.y` [package]

Position y.

La documentation de cette classe a été générée à partir du fichier suivant :

– [Vector.java](#)

4.17 Référence de la classe Window

Moteur graphique.

Fonctions statiques de paquetage

- static [Window make](#) ([Box](#) box)
Constructeur.
- static void [renderBox](#) ([Window](#) win, [Box](#) box)
Rendu de la table de billard.
- static void [renderInput](#) ([Window](#) win, [Input](#) input)
Rendu de l'entrée clavier.
- static void [renderQueue](#) ([RenderBall](#) rB)
Rendu de la canne de jeu.
- static void [update](#) ([Window](#) win, [Box](#) box, [Input](#) input)
Synchronisation avec le système physique.
- static void [render](#) ([Window](#) win, [Box](#) box, [Input](#) input)
Rendu de la fenêtre.

Attributs de paquetage

- int [fps](#) = 60
Taux de rafraichissement.
- int [sizeMenu](#) = 200
Taille du menu en largeur.
- int [sizeBoard](#) = 200
Taille du bandeau en hauteur.
- int [scale](#) = 300
Echelle de conversion mètre/pixels.
- int [width](#)
Largeur totale de l'intérieur de la fenêtre.
- int [height](#)
Hauteur totale de l'intérieur de la fenêtre.
- int [nbBalls](#)
Nombre de boules.
- [RenderBall\[\]](#) [renderBalls](#)
Tableau de boules à dessiner.
- [Menu](#) [menu](#)
Menu de l'interface graphique.
- [Board](#) [board](#)
Bandeau moniteur.

4.17.1 Description détaillée

Moteur graphique.

Auteurs

Baptiste Minervini
Romain Mekarni

4.17.2 Documentation des fonctions membres

4.17.2.1 static Window Window.make ([Box](#) box) [static], [package]

Constructeur.

Auteurs

Baptiste Minervini
Romain Mekarni

4.17.2.2 `static void Window.render (Window win, Box box, Input input)` [static],[package]

Rendu de la fenêtre.

Auteurs

Baptiste Minervini
Romain Mekarni

4.17.2.3 `static void Window.renderBox (Window win, Box box)` [static],[package]

Rendu de la table de billard.

Auteur

Baptiste Minervini

4.17.2.4 `static void Window.renderInput (Window win, Input input)` [static],[package]

Rendu de l'entrée clavier.

Auteur

Romain Mekarni

4.17.2.5 `static void Window.renderQueue (RenderBall rB)` [static],[package]

Rendu de la canne de jeu.

Auteur

Baptiste Minervini

4.17.2.6 `static void Window.update (Window win, Box box, Input input)` [static],[package]

Synchronisation avec le système physique.

Auteur

Romain Mekarni

4.17.3 Documentation des données membres

4.17.3.1 `Board Window.board` [package]

Bandeau moniteur.

4.17.3.2 `int Window.fps = 60` [package]

Taux de rafraichissement.

4.17.3.3 `int Window.height` [package]

Hauteur totale de l'intérieur de la fenêtre.

4.17.3.4 Menu Window.menu [package]

[Menu](#) de l'interface graphique.

4.17.3.5 int Window.nbBalls [package]

Nombre de boules.

4.17.3.6 RenderBall [] Window.renderBalls [package]

Tableau de boules à dessiner.

4.17.3.7 int Window.scale = 300 [package]

Echelle de conversion mètre/pixels.

4.17.3.8 int Window.sizeBoard = 200 [package]

Taille du bandeau en hauteur.

4.17.3.9 int Window.sizeMenu = 200 [package]

Taille du menu en largeur.

4.17.3.10 int Window.width [package]

Largeur totale de l'intérieur de la fenêtre.

La documentation de cette classe a été générée à partir du fichier suivant :

– [Window.java](#)

Chapitre 5

Documentation des fichiers

5.1 Référence du fichier Ball.java

Représentation physique d'une boule et interaction avec les autres objets.

Classes

- class `Ball`
Gestion des Boules.

5.1.1 Description détaillée

Représentation physique d'une boule et interaction avec les autres objets.

Auteurs

Romain Mekarni
Baptiste Minervini

5.2 Référence du fichier Billard.java

Fonction main, instanciation des modules principaux.

Classes

- class `Billard`
Classe principale du programme.

5.2.1 Description détaillée

Fonction main, instanciation des modules principaux.

Auteurs

Baptiste Minervini
Romain Mekarni

5.3 Référence du fichier Board.java

Bandeau de mesures du système physique.

Classes

- class [Indicator](#)
Mesure numérique d'une grandeur physique du système physique.
- class [Graph](#)
Mesure graphique d'une grandeur physique du système physique.
- class [Board](#)
Bandeau d'affichage des grandeurs physique du système.

5.3.1 Description détaillée

Bandeau de mesures du système physique.

Auteur

Romain Mekarni

5.4 Référence du fichier Box.java

Moteur physique - Simulation du [Billard](#).

Classes

- class [Box](#)
Moteur physique.

5.4.1 Description détaillée

Moteur physique - Simulation du [Billard](#).

Auteurs

Baptiste Minervini
Romain Mekarni

5.5 Référence du fichier Button.java

Boutton du [Menu](#).

Classes

- class [Button](#)
Boutton du [Menu](#).

5.5.1 Description détaillée

Boutton du [Menu](#).

Auteurs

Baptiste Minervini
Romain Mekarni

5.6 Référence du fichier Color.java

Gestion des couleurs pour l'API EcranGraphique.

Classes

- class [Color](#)
Classe représentant une couleur.

5.6.1 Description détaillée

Gestion des couleurs pour l'API EcranGraphique.

Auteurs

Baptiste Minervini
Romain Mekarni

5.7 Référence du fichier Input.java

Gestion des interactions utilisateur.

Classes

- class [Input](#)
Gestion des interactions utilisateur.

5.7.1 Description détaillée

Gestion des interactions utilisateur.

Auteurs

Baptiste Minervini
Romain Mekarni

5.8 Référence du fichier mainpage.dox

Page principale de la documentation du projet [Billard](#).

5.8.1 Description détaillée

Page principale de la documentation du projet [Billard](#).

Auteur

Romain Mekarni

5.9 Référence du fichier Menu.java

[Menu](#) interactif à disposition de l'utilisateur.

Classes

- class [Menu](#)
menu interactif à disposition de l'utilisateur

5.9.1 Description détaillée

[Menu](#) interactif à disposition de l'utilisateur.

Auteurs

Baptiste Minervini
Romain Mekarni

5.10 Référence du fichier Rect.java

Gestion des formes rectangulaires.

Classes

- class [Rect](#)
Classe représentant un rectangle soit sa taille et sa position.

5.10.1 Description détaillée

Gestion des formes rectangulaires.

Auteur

Romain Mekarni

5.11 Référence du fichier RenderBall.java

Rendu des boules.

Classes

- class [RenderBall](#)
Rendu des boules.

5.11.1 Description détaillée

Rendu des boules.

Auteurs

Baptiste Minervini
Romain Mekarni

5.12 Référence du fichier Stack.java

Structure de données de type pile.

Classes

- class [Event](#)
Représente un évènement de type choc.
- class [Item](#)
Element de [Stack](#) qui contient un [Event](#).
- class [Stack](#)
Conteneur de [Item](#) de type pile.

5.12.1 Description détaillée

Structure de données de type pile.

Auteur

Romain Mekarni

5.13 Référence du fichier Vector.java

Manipulation des vecteurs mathématiques pour la physique.

Classes

- class [Vector](#)
Manipulation des vecteurs mathématiques.

5.13.1 Description détaillée

Manipulation des vecteurs mathématiques pour la physique.

Auteur

Romain Mekarni

5.14 Référence du fichier Window.java

Moteur graphique.

Classes

- class [Window](#)
Moteur graphique.

5.14.1 Description détaillée

Moteur graphique.

Auteurs

Baptiste Minervini
Romain Mekarni

Index

a
 Vector, [32](#)

atan
 Vector, [31](#)

b
 Button, [18](#)
 Color, [19](#)

b1
 Event, [20](#)

b2
 Event, [20](#)

Ball, [9](#)
 chocBalls, [9](#)
 chocBox, [10](#)
 dtChocBalls, [10](#)
 dtChocBox, [10](#)
 id, [10](#)
 m, [10](#)
 p, [10](#)
 r, [10](#)
 v, [10](#)

Ball.java, [37](#)

ballFocus
 Box, [15](#)

balls
 Box, [15](#)

baseBalls
 Box, [15](#)

Billard, [11](#)
 main, [11](#)

Billard.java, [37](#)

Board, [11](#)
 graphs, [12](#)
 indicators, [12](#)
 make, [12](#)
 nbGraphs, [12](#)
 nbIndicators, [12](#)
 render, [12](#)
 size, [13](#)
 update, [12](#)

board
 Window, [34](#)

Board.java, [38](#)

border
 Button, [18](#)

Box, [13](#)
 ballFocus, [15](#)
 balls, [15](#)
 baseBalls, [15](#)
 dt, [15](#)
 evolve, [14](#)
 friction, [15](#)
 length, [15](#)
 make, [14](#)
 nbBalls, [15](#)
 nbChoc, [15](#)
 p, [15](#)
 pollEvent, [14](#)
 posLine, [14](#)
 posTriangle, [14](#)
 r, [16](#)
 rayon, [16](#)
 run, [16](#)
 s, [16](#)
 t, [16](#)
 update, [14](#), [15](#)
 vmoy, [16](#)
 width, [16](#)

Box.java, [38](#)

buffer
 Input, [23](#)

Button, [16](#)
 b, [18](#)
 border, [18](#)
 design, [18](#)
 make, [17](#)
 render, [17](#)
 size, [18](#)
 state, [18](#)
 title, [18](#)
 update, [17](#)

Button.java, [38](#)

buttons
 Menu, [26](#)

c
 RenderBall, [29](#)

chocBalls
 Ball, [9](#)

chocBox
 Ball, [10](#)

clear
 Stack, [30](#)

Color, [18](#)
 b, [19](#)
 make, [19](#)
 r, [19](#)
 set, [19](#)
 v, [19](#)

- Color.java, 39
- design
 - Button, 18
- dt
 - Box, 15
- dtChocBalls
 - Ball, 10
- dtChocBox
 - Ball, 10
- dump
 - Vector, 31
- Event, 19
 - b1, 20
 - b2, 20
 - type, 20
- event
 - Item, 24
- evolve
 - Box, 14
- fill
 - Rect, 27
- first
 - Stack, 30
- formeCart
 - Vector, 31
- formePol
 - Vector, 31
- fps
 - Window, 34
- friction
 - Box, 15
- Graph, 20
 - graph, 21
 - indicator, 21
 - make, 20
 - moy, 21
 - render, 21
 - scale, 21
 - size, 21
- graph
 - Graph, 21
- graphs
 - Board, 12
- h
 - Rect, 27
- height
 - Window, 34
- id
 - Ball, 10
- Indicator, 21
 - make, 22
 - name, 22
 - precision, 22
 - render, 22
 - size, 22
 - unite, 22
 - value, 22
- indicator
 - Graph, 21
- indicators
 - Board, 12
- Input, 22
 - buffer, 23
 - kb, 23
 - out, 24
 - queue, 24
 - queuing, 23
 - update, 23
- Input.java, 39
- isEmpty
 - Stack, 30
- isIn
 - Rect, 27
- Item, 24
 - event, 24
 - next, 24
- kb
 - Input, 23
- length
 - Box, 15
- m
 - Ball, 10
 - Vector, 32
- main
 - Billard, 11
- mainpage.dox, 39
- make
 - Board, 12
 - Box, 14
 - Button, 17
 - Color, 19
 - Graph, 20
 - Indicator, 22
 - Menu, 25
 - Rect, 27
 - RenderBall, 28
 - Window, 33
- Menu, 24
 - buttons, 26
 - make, 25
 - nbButtons, 26
 - render, 25
 - size, 26
 - title, 26
 - update, 25
- menu
 - Window, 34
- Menu.java, 40
- module
 - Vector, 32

moy
 Graph, 21

name
 Indicator, 22

nbBalls
 Box, 15
 Window, 35

nbButtons
 Menu, 26

nbChoc
 Box, 15

nbGraphs
 Board, 12

nbIndicators
 Board, 12

newBase
 Vector, 32

next
 Item, 24

out
 Input, 24

p
 Ball, 10
 Box, 15

pollEvent
 Box, 14

posLine
 Box, 14

posTriangle
 Box, 14

precision
 Indicator, 22

pull
 Stack, 30

push
 Stack, 30

queue
 Input, 24

queuing
 Input, 23

r
 Ball, 10
 Box, 16
 Color, 19
 RenderBall, 29

rayon
 Box, 16

Rect, 26
 fill, 27
 h, 27
 isIn, 27
 make, 27
 w, 27
 x, 27
 y, 27

Rect.java, 40

render
 Board, 12
 Button, 17
 Graph, 21
 Indicator, 22
 Menu, 25
 RenderBall, 28
 Window, 33

RenderBall, 27
 c, 29
 make, 28
 r, 29
 render, 28
 update, 28
 x, 29
 y, 29

RenderBall.java, 40

renderBalls
 Window, 35

renderBox
 Window, 34

renderInput
 Window, 34

renderQueue
 Window, 34

reverseBase
 Vector, 32

run
 Box, 16

s
 Box, 16

scale
 Graph, 21
 Window, 35

set
 Color, 19

size
 Board, 13
 Button, 18
 Graph, 21
 Indicator, 22
 Menu, 26
 Stack, 30

sizeBoard
 Window, 35

sizeMenu
 Window, 35

Stack, 29
 clear, 30
 first, 30
 isEmpty, 30
 pull, 30
 push, 30
 size, 30

Stack.java, 41

state

- Button, [18](#)
- t
 - Box, [16](#)
- title
 - Button, [18](#)
 - Menu, [26](#)
- type
 - Event, [20](#)
- unite
 - Indicator, [22](#)
- update
 - Board, [12](#)
 - Box, [14](#), [15](#)
 - Button, [17](#)
 - Input, [23](#)
 - Menu, [25](#)
 - RenderBall, [28](#)
 - Window, [34](#)
- v
 - Ball, [10](#)
 - Color, [19](#)
- value
 - Indicator, [22](#)
- Vector, [30](#)
 - a, [32](#)
 - atan, [31](#)
 - dump, [31](#)
 - formeCart, [31](#)
 - formePol, [31](#)
 - m, [32](#)
 - module, [32](#)
 - newBase, [32](#)
 - reverseBase, [32](#)
 - x, [32](#)
 - y, [32](#)
- Vector.java, [41](#)
- vmoy
 - Box, [16](#)
- w
 - Rect, [27](#)
- width
 - Box, [16](#)
 - Window, [35](#)
- Window, [33](#)
 - board, [34](#)
 - fps, [34](#)
 - height, [34](#)
 - make, [33](#)
 - menu, [34](#)
 - nbBalls, [35](#)
 - render, [33](#)
 - renderBalls, [35](#)
 - renderBox, [34](#)
 - renderInput, [34](#)
 - renderQueue, [34](#)
 - scale, [35](#)
 - sizeBoard, [35](#)
 - sizeMenu, [35](#)
 - update, [34](#)
 - width, [35](#)
- Window.java, [41](#)
- x
 - Rect, [27](#)
 - RenderBall, [29](#)
 - Vector, [32](#)
- y
 - Rect, [27](#)
 - RenderBall, [29](#)
 - Vector, [32](#)