Billard

Généré par Doxygen 1.8.1.2

Mercredi Mai 13 2015 18 :28 :17

Table des matières

1 Projet Billard **1**

1.1 Présentation . . . 1

1.2 Installation . . . 1

1.3 Dépendances . . . 1

1.4 Notice d'utilisation . . . 1

1.5 Développement . . . 2

1.5.1 Premier jet graphique . . . 2

1.5.2 Séparation physique et graphique . . . 2

1.5.3 Ajout des chocs entre boules et contre les murs . . . 2

1.5.4 Implémentation des calculs physiques . . . 2

1.5.5 Gestion de la précision et synchronisaton avec la boucle principale . . . 3

1.6 Physique et Mathématiques . . . 3

1.6.1 Temps de choc entre deux boules . . . 3

1.6.2 Calcul des vitesses et angles après choc entre deux boules . . . 3

1.7 Difficultés rencontrées . . . 4

1.7.1 Conversion entre système polaire et cartésien . . . 4

2 Index des classes **7**

2.1 Liste des classes . . . 7

3 Index des fichiers **9**

3.1 Liste des fichiers . . . 9

4 Documentation des classes **11**

4.1 Référence de la classe Ball . . . 11

4.1.1 Description détaillée . . . 11

4.1.2 Documentation des fonctions membres . . . 11

4.1.2.1 chocBalls . . . 11

4.1.2.2 chocBox . . . 12

4.1.2.3 dtChocBalls . . . 12

4.1.2.4 dtChocBox . . . 12

4.1.3 Documentation des données membres . . . 12

4.1.3.1 id . . . 12
4.1.3.2 m . . . 12
4.1.3.3 p . . . 12
4.1.3.4 r . . . 12
4.1.3.5 v . . . 13
4.2 Référence de la classe Billard . . . 13
4.2.1 Description détaillée . . . 13
4.2.2 Documentation des fonctions membres . . . 13
4.2.2.1 main . . . 13
4.3 Référence de la classe Board . . . 13
4.3.1 Description détaillée . . . 14
4.3.2 Documentation des fonctions membres . . . 14
4.3.2.1 make . . . 14
4.3.2.2 render . . . 14
4.3.2.3 update . . . 14
4.3.3 Documentation des données membres . . . 14
4.3.3.1 graphs . . . 14
4.3.3.2 indicators . . . 14
4.3.3.3 nbGraphs . . . 14
4.3.3.4 nbIndicators . . . 15
4.3.3.5 size . . . 15
4.4 Référence de la classe Box . . . 15
4.4.1 Description détaillée . . . 16
4.4.2 Documentation des fonctions membres . . . 16
4.4.2.1 evolve . . . 16
4.4.2.2 make . . . 16
4.4.2.3 pollEvent . . . 16
4.4.2.4 posLine . . . 16
4.4.2.5 posTriangle . . . 16
4.4.2.6 update . . . 17
4.4.2.7 update . . . 17
4.4.3 Documentation des données membres . . . 17
4.4.3.1 ballFocus . . . 17
4.4.3.2 balls . . . 17
4.4.3.3 baseBalls . . . 17
4.4.3.4 dt . . . 17
4.4.3.5 friction . . . 17
4.4.3.6 length . . . 17
4.4.3.7 nbBalls . . . 17
4.4.3.8 nbChoc . . . 17

4.4.3.9 p . . . 18
4.4.3.10 r . . . 18
4.4.3.11 rayon . . . 18
4.4.3.12 run . . . 18
4.4.3.13 s . . . 18
4.4.3.14 t . . . 18
4.4.3.15 vmoy . . . 18
4.4.3.16 width . . . 18
4.5 Référence de la classe Button . . . 18
4.5.1 Description détaillée . . . 19
4.5.2 Documentation des fonctions membres . . . 19
4.5.2.1 make . . . 19
4.5.2.2 render . . . 19
4.5.2.3 update . . . 19
4.5.3 Documentation des données membres . . . 20
4.5.3.1 b . . . 20
4.5.3.2 border . . . 20
4.5.3.3 design . . . 20
4.5.3.4 size . . . 20
4.5.3.5 state . . . 20
4.5.3.6 title . . . 20
4.6 Référence de la classe Color . . . 20
4.6.1 Description détaillée . . . 20
4.6.2 Documentation des fonctions membres . . . 21
4.6.2.1 make . . . 21
4.6.2.2 set . . . 21
4.6.3 Documentation des données membres . . . 21
4.6.3.1 b . . . 21
4.6.3.2 r . . . 21
4.6.3.3 v . . . 21
4.7 Référence de la classe Event . . . 21
4.7.1 Description détaillée . . . 21
4.7.2 Documentation des données membres . . . 22
4.7.2.1 b1 . . . 22
4.7.2.2 b2 . . . 22
4.7.2.3 type . . . 22
4.8 Référence de la classe Graph . . . 22
4.8.1 Description détaillée . . . 22
4.8.2 Documentation des fonctions membres . . . 22
4.8.2.1 make . . . 22

4.8.2.2 render . . . 23
4.8.3 Documentation des données membres . . . 23
4.8.3.1 graph . . . 23
4.8.3.2 indicator . . . 23
4.8.3.3 moy . . . 23
4.8.3.4 scale . . . 23
4.8.3.5 size . . . 23
4.9 Référence de la classe Indicator . . . 23
4.9.1 Description détaillée . . . 24
4.9.2 Documentation des fonctions membres . . . 24
4.9.2.1 make . . . 24
4.9.2.2 render . . . 24
4.9.3 Documentation des données membres . . . 24
4.9.3.1 name . . . 24
4.9.3.2 precision . . . 24
4.9.3.3 size . . . 24
4.9.3.4 unite . . . 24
4.9.3.5 value . . . 24
4.10 Référence de la classe Input . . . 24
4.10.1 Description détaillée . . . 25
4.10.2 Documentation des fonctions membres . . . 25
4.10.2.1 queuing . . . 25
4.10.2.2 update . . . 25
4.10.3 Documentation des données membres . . . 25
4.10.3.1 buffer . . . 25
4.10.3.2 kb . . . 26
4.10.3.3 out . . . 26
4.10.3.4 queue . . . 26
4.11 Référence de la classe Item . . . 26
4.11.1 Description détaillée . . . 26
4.11.2 Documentation des données membres . . . 26
4.11.2.1 event . . . 26
4.11.2.2 next . . . 26
4.12 Référence de la classe Menu . . . 26
4.12.1 Description détaillée . . . 27
4.12.2 Documentation des fonctions membres . . . 27
4.12.2.1 make . . . 27
4.12.2.2 render . . . 27
4.12.2.3 update . . . 27
4.12.3 Documentation des données membres . . . 28

4.12.3.1 buttons . . . 28
4.12.3.2 nbButtons . . . 28
4.12.3.3 size . . . 28
4.12.3.4 title . . . 28
4.13 Référence de la classe Rect . . . 28
4.13.1 Description détaillée . . . 29
4.13.2 Documentation des fonctions membres . . . 29
4.13.2.1 fill . . . 29
4.13.2.2 isIn . . . 29
4.13.2.3 make . . . 29
4.13.3 Documentation des données membres . . . 29
4.13.3.1 h . . . 29
4.13.3.2 w . . . 29
4.13.3.3 x . . . 29
4.13.3.4 y . . . 29
4.14 Référence de la classe RenderBall . . . 29
4.14.1 Description détaillée . . . 30
4.14.2 Documentation des fonctions membres . . . 30
4.14.2.1 make . . . 30
4.14.2.2 render . . . 30
4.14.2.3 update . . . 30
4.14.3 Documentation des données membres . . . 31
4.14.3.1 c . . . 31
4.14.3.2 r . . . 31
4.14.3.3 x . . . 31
4.14.3.4 y . . . 31
4.15 Référence de la classe Stack . . . 31
4.15.1 Description détaillée . . . 31
4.15.2 Documentation des fonctions membres . . . 32
4.15.2.1 clear . . . 32
4.15.2.2 isEmpty . . . 32
4.15.2.3 pull . . . 32
4.15.2.4 push . . . 32
4.15.2.5 size . . . 32
4.15.3 Documentation des données membres . . . 32
4.15.3.1 first . . . 32
4.16 Référence de la classe Vector . . . 32
4.16.1 Description détaillée . . . 33
4.16.2 Documentation des fonctions membres . . . 33
4.16.2.1 atan . . . 33

4.16.2.2 dump . . . 33
4.16.2.3 formeCart . . . 33
4.16.2.4 formePol . . . 34
4.16.2.5 module . . . 34
4.16.2.6 newBase . . . 34
4.16.2.7 reverseBase . . . 34
4.16.3 Documentation des données membres . . . 34
4.16.3.1 a . . . 34
4.16.3.2 m . . . 34
4.16.3.3 x . . . 34
4.16.3.4 y . . . 34
4.17 Référence de la classe Window . . . 35
4.17.1 Description détaillée . . . 35
4.17.2 Documentation des fonctions membres . . . 35
4.17.2.1 make . . . 35
4.17.2.2 render . . . 36
4.17.2.3 renderBox . . . 36
4.17.2.4 renderInput . . . 36
4.17.2.5 renderQueue . . . 36
4.17.2.6 update . . . 36
4.17.3 Documentation des données membres . . . 36
4.17.3.1 board . . . 36
4.17.3.2 fps . . . 36
4.17.3.3 height . . . 36
4.17.3.4 menu . . . 37
4.17.3.5 nbBalls . . . 37
4.17.3.6 renderBalls . . . 37
4.17.3.7 scale . . . 37
4.17.3.8 sizeBoard . . . 37
4.17.3.9 sizeMenu . . . 37
4.17.3.10 width . . . 37

5 Documentation des fichiers **39**

5.1 Référence du fichier Ball.java . . . 39
5.1.1 Description détaillée . . . 39
5.2 Référence du fichier Billard.java . . . 39
5.2.1 Description détaillée . . . 39
5.3 Référence du fichier Board.java . . . 40
5.3.1 Description détaillée . . . 40
5.4 Référence du fichier Box.java . . . 40

5.4.1 Description détaillée . . . 40
5.5 Référence du fichier Button.java . . . 40
5.5.1 Description détaillée . . . 40
5.6 Référence du fichier Color.java . . . 41
5.6.1 Description détaillée . . . 41
5.7 Référence du fichier Input.java . . . 41
5.7.1 Description détaillée . . . 41
5.8 Référence du fichier mainpage.dox . . . 41
5.8.1 Description détaillée . . . 41
5.9 Référence du fichier Menu.java . . . 42
5.9.1 Description détaillée . . . 42
5.10 Référence du fichier Rect.java . . . 42
5.10.1 Description détaillée . . . 42
5.11 Référence du fichier RenderBall.java . . . 42
5.11.1 Description détaillée . . . 42
5.12 Référence du fichier Stack.java . . . 43
5.12.1 Description détaillée . . . 43
5.13 Référence du fichier Vector.java . . . 43
5.13.1 Description détaillée . . . 43
5.14 Référence du fichier Window.java . . . 43
5.14.1 Description détaillée . . . 43

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 Notice d'utilisation

Dans la fenêtre graphique, il y a 11 boutons à disposition de l'utilisateur :
- Billard classique : dispose les boules en configuration du jeu billard

– Boules en ligne : mode demandé par l'énoncé du projet
– Jeu manuel : utilisation de la queue de billard sur la boule ciblée. Cliquer sur bouton droit de la souris pour tirer
– Démarrer / Pause : met en pause ou reprend l'évolution du système physique
– dt (Box.dt) : vitesse d'évolution du moteur physique (cliquez pour modifier)
– Precision (Box.s) : nombre de chiffres significatifs pris en compte dans les calculs physiques
– FPS (Window.fps) : Nombre d'images affichées par seconde. 0 pour désactiver le blocage (affiche dès que les calculs sont finis, variable selon l'ordinateur)
– Rayon (Box.rayon) : taille en mètre des boules de billard
– Boule (Box.ballFocus) : mécanisme de ciblage d'une boule pour interagir avec le jeu manuel et afficher ses informations dans le bandeau
– Frottement (Box.friction) : valeur prise en compte dans les calculs de vitesse des boules à chaque dt
– Quitter : stop proprement le programme

Le bandeau d'informations contient 2 graphiques : le premier affiche le nombre de chocs pris en compte dans un dt, et le second affiche la somme des vitesses des boules du système. A droite s'affiche les informations (position, vitesse, angle) de la boule ciblée.

Lors de la saisie de données, appuyer sur Entrée pour valider la saisie. Il est possible d'annuler en appuyant sur la touche échap.

1.5 Développement

1.5.1 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.

1.5.2 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.

1.5.3 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).

1.5.4 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 approchent, 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.

1.5.5 Gestion de la précision et synchronisaton avec la boucle principale

Pour régler ce problème de précision, on a choisit 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 suivent :
- 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).

1.6 Physique et Mathématiques

Déterminer les chocs entre boules est assez complexe en physique et exploite plusieurs équations physiques de description du mouvement et des énergies.

1.6.1 Temps de choc entre deux boules

Pour détecter si deux boules se rencontrent, on cherche un point de rencontre entre les deux équations de trajectoires. Après reformulation, on trouve une équation du second degré :

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

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

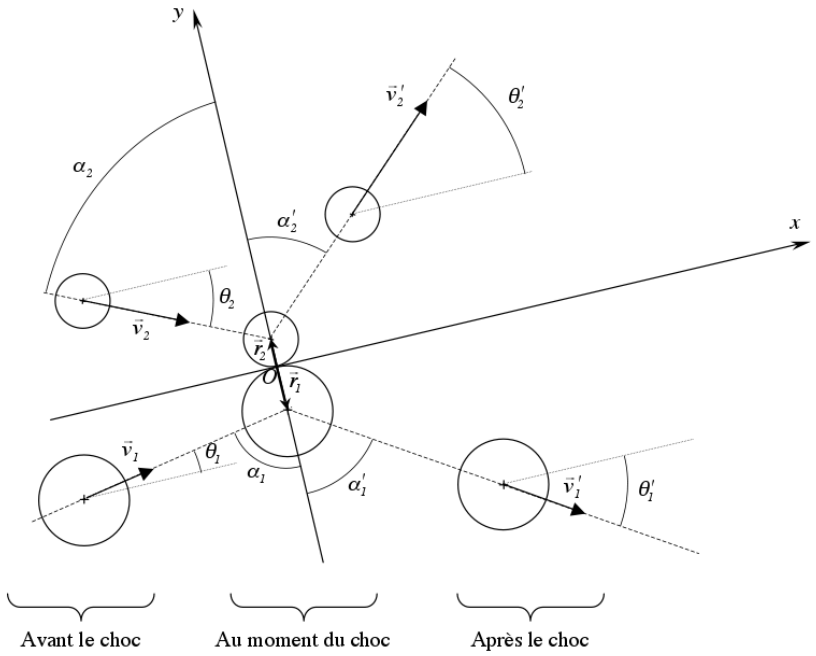
Avec le déterminant, on récupère les solutions, si elles existent, en veillant bien à prendre la plus petite positive.

1.6.2 Calcul des vitesses et angles après choc entre deux boules

Etude réalisée grace 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 autour de leur centre, elles ne sont animées que d'un mouvement de translation. Pour déterminer les 4 nouveaux

FIGURE 1.1 – Choc élastique en 2 dimensions.



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 :

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

1.7 Difficultés rencontrées

1.7.1 Conversion entre système polaire et cartésien

Problèmes lors du calcul de l'angle de rotation pour le changement de repère car il faisait intervenir la fonction arctan qui possède de nombreux cas particuliers à gérer.

De plus, il a fallut de nombreux essais pour prendre conscience que pour changer un repère il ne suffit pas de modifier l'angle des vecteurs en question, il faut utiliser une matrice de passage pour changer de base, et y revenir une fois les calculs faits. A partir de Wikipédia on obtient enfin des calculs correctes qui se vérifient graphiquement.

Une autre difficulté a été de concevoir le programme sans utiliser la programmation orientée objet. Il faut sans arrêt transmettre en argument les types agrégés considérés aux fonctions statiques.

Faire du graphique avec une bibliothèque bas niveau qui ne permet que de dessiner des formes géométriques est assez casse-tête (surtout pour faire des menus, boutons etc..) Cette partie du programme impose plus de types

agrégés et de nombreuses fonctions supplémentaires pour un traitement optimal (sans répétition de code).

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 . . . 11

Billard
: Classe principale du programme . . . 13

Board
: Bandeau d'affichage des grandeurs physique du système . . . 13

Box
: Moteur physique . . . 15

Button
: Boutton du Menu . . . 18

Color
: Classe représentant une couleur . . . 20

Event
: Représente un évènement de type choc . . . 21

Graph
: Mesure graphique d'une grandeur physique du système physique . . . 22

Indicator
: Mesure numérique d'une grandeur physique du système physique . . . 23

Input
: Gestion des interactions utilisateur . . . 24

Item
: Element de Stack qui contient un Event . . . 26

Menu
: Menu interactif à disposition de l'utilisateur . . . 26

Rect
: Classe représentant un rectangle soit sa taille et sa position . . . 28

RenderBall
: Rendu des boules . . . 29

Stack
: Conteneur de Item de type pile . . . 31

Vector
: Manipulation des vecteurs mathématiques . . . 32

Window
: Moteur graphique . . . 35

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 . . . 39

Billard.java
: Fonction main, instanciation des modules principaux . . . 39

Board.java
: Bandeau de mesures du système physique . . . 40

Box.java
: Moteur physique - Simulation du Billard . . . 40

Button.java
: Boutton du Menu . . . 40

Color.java
: Gestion des couleurs pour l'API EcranGraphique . . . 41

Input.java
: Gestion des interactions utilisateur . . . 41

Menu.java
: Menu interactif à disposition de l'utilisateur . . . 42

Rect.java
: Gestion des formes rectangulaires . . . 42

RenderBall.java
: Rendu des boules . . . 42

Stack.java
: Structure de données de type pile . . . 43

Vector.java
: Manipulation des vecteurs mathématiques pour la physique . . . 43

Window.java
: Moteur graphique . . . 43

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

size	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 evoluer le systeme 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 evoluer le systeme 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

title	Tableau de chaines contenant le libellé du bouton
size	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

scale	Echelle
size	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

box	La queue agit sur la boule ciblée et relance le moteur physique
win	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

box	La queue de billard agit sur la boule ciblée
win	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

size	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

type	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, 34
atan
    Vector, 33

b
    Button, 20
    Color, 21
b1
    Event, 22
b2
    Event, 22
Ball, 11
    chocBalls, 11
    chocBox, 12
    dtChocBalls, 12
    dtChocBox, 12
    id, 12
    m, 12
    p, 12
    r, 12
    v, 12
Ball.java, 39
ballFocus
    Box, 17
balls
    Box, 17
baseBalls
    Box, 17
Billard, 13
    main, 13
Billard.java, 39
Board, 13
    graphs, 14
    indicators, 14
    make, 14
    nbGraphs, 14
    nbIndicators, 14
    render, 14
    size, 15
    update, 14
board
    Window, 36
Board.java, 40
border
    Button, 20
Box, 15
    ballFocus, 17
    balls, 17
    baseBalls, 17
    dt, 17
    evolve, 16
    friction, 17
    length, 17
    make, 16
    nbBalls, 17
    nbChoc, 17
    p, 17
    pollEvent, 16
    posLine, 16
    posTriangle, 16
    r, 18
    rayon, 18
    run, 18
    s, 18
    t, 18
    update, 16, 17
    vmoy, 18
    width, 18
Box.java, 40
buffer
    Input, 25
Button, 18
    b, 20
    border, 20
    design, 20
    make, 19
    render, 19
    size, 20
    state, 20
    title, 20
    update, 19
Button.java, 40
buttons
    Menu, 28

c
    RenderBall, 31
chocBalls
    Ball, 11
chocBox
    Ball, 12
clear
    Stack, 32
Color, 20
    b, 21
    make, 21
    r, 21
    set, 21
    v, 21

Color.java, 41

design
    Button, 20
dt
    Box, 17
dtChocBalls
    Ball, 12
dtChocBox
    Ball, 12
dump
    Vector, 33

Event, 21
    b1, 22
    b2, 22
    type, 22
event
    Item, 26
evolve
    Box, 16

fill
    Rect, 29
first
    Stack, 32
formeCart
    Vector, 33
formePol
    Vector, 33
fps
    Window, 36
friction
    Box, 17

Graph, 22
    graph, 23
    indicator, 23
    make, 22
    moy, 23
    render, 23
    scale, 23
    size, 23
graph
    Graph, 23
graphs
    Board, 14

h
    Rect, 29
height
    Window, 36

id
    Ball, 12
Indicator, 23
    make, 24
    name, 24
    precision, 24
    render, 24
    size, 24
    unite, 24
    value, 24
indicator
    Graph, 23
indicators
    Board, 14
Input, 24
    buffer, 25
    kb, 25
    out, 26
    queue, 26
    queuing, 25
    update, 25
Input.java, 41
isEmpty
    Stack, 32
isIn
    Rect, 29
Item, 26
    event, 26
    next, 26

kb
    Input, 25

length
    Box, 17

m
    Ball, 12
    Vector, 34
main
    Billard, 13
mainpage.dox, 41
make
    Board, 14
    Box, 16
    Button, 19
    Color, 21
    Graph, 22
    Indicator, 24
    Menu, 27
    Rect, 29
    RenderBall, 30
    Window, 35
Menu, 26
    buttons, 28
    make, 27
    nbButtons, 28
    render, 27
    size, 28
    title, 28
    update, 27
menu
    Window, 36
Menu.java, 42
module
    Vector, 34

moy
Graph, 23

name
Indicator, 24
nbBalls
Box, 17
Window, 37
nbButtons
Menu, 28
nbChoc
Box, 17
nbGraphs
Board, 14
nbIndicators
Board, 14
newBase
Vector, 34
next
Item, 26

out
Input, 26

p
Ball, 12
Box, 17
pollEvent
Box, 16
posLine
Box, 16
posTriangle
Box, 16
precision
Indicator, 24
pull
Stack, 32
push
Stack, 32

queue
Input, 26
queuing
Input, 25

r
Ball, 12
Box, 18
Color, 21
RenderBall, 31
rayon
Box, 18
Rect, 28
fill, 29
h, 29
isIn, 29
make, 29
w, 29
x, 29
y, 29
Rect.java, 42
render
Board, 14
Button, 19
Graph, 23
Indicator, 24
Menu, 27
RenderBall, 30
Window, 35
RenderBall, 29
c, 31
make, 30
r, 31
render, 30
update, 30
x, 31
y, 31
RenderBall.java, 42
renderBalls
Window, 37
renderBox
Window, 36
renderInput
Window, 36
renderQueue
Window, 36
reverseBase
Vector, 34
run
Box, 18

s
Box, 18
scale
Graph, 23
Window, 37
set
Color, 21
size
Board, 15
Button, 20
Graph, 23
Indicator, 24
Menu, 28
Stack, 32
sizeBoard
Window, 37
sizeMenu
Window, 37
Stack, 31
clear, 32
first, 32
isEmpty, 32
pull, 32
push, 32
size, 32
Stack.java, 43
state

Button, 20

t
Box, 18
title
Button, 20
Menu, 28
type
Event, 22

unite
Indicator, 24
update
Board, 14
Box, 16, 17
Button, 19
Input, 25
Menu, 27
RenderBall, 30
Window, 36

v
Ball, 12
Color, 21
value
Indicator, 24
Vector, 32
a, 34
atan, 33
dump, 33
formeCart, 33
formePol, 33
m, 34
module, 34
newBase, 34
reverseBase, 34
x, 34
y, 34
Vector.java, 43
vmoy
Box, 18

w
Rect, 29
width
Box, 18
Window, 37
Window, 35
board, 36
fps, 36
height, 36
make, 35
menu, 36
nbBalls, 37
render, 35
renderBalls, 37
renderBox, 36
renderInput, 36
renderQueue, 36
scale, 37
sizeBoard, 37
sizeMenu, 37
update, 36
width, 37
Window.java, 43

x
Rect, 29
RenderBall, 31
Vector, 34

y
Rect, 29
RenderBall, 31
Vector, 34