RAPPORT: Projet POO 2

Simulateur de gravité en 2D

LUTTRINGER

MULLER

Jean – Romain

Cyrille

Présentation rapide:

Le programme a été codé en Lisaac et se lance, après compilation via la commande « ./main ». Une fois lancé, il est possible de rajouter des cercles en faisant soit un clic gauche (pour un rayon de 15) soit un clic droit (pour un rayon de 35) à l'endroit voulu dans la fenêtre.

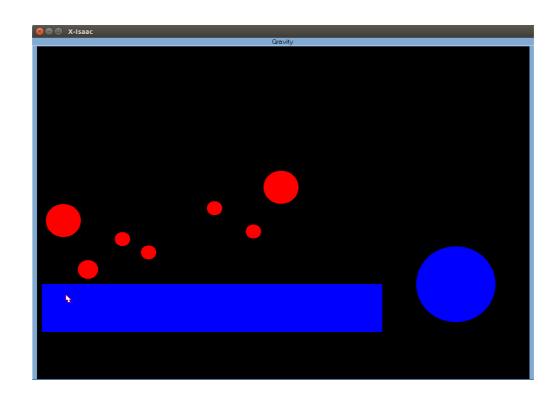
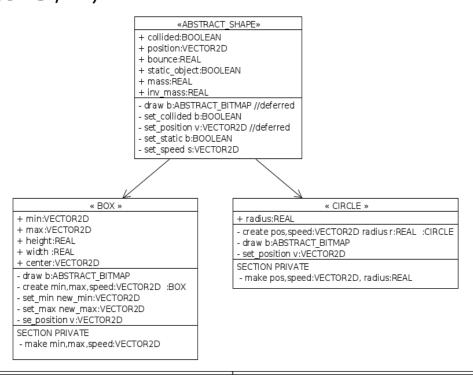


DIAGRAMME DES CLASSES

(en omettant l'héritage du prototype **OBJECT** présent dans tous les prototypes)

Le diagramme d'héritage est plutôt simple. Notre projet est constitué d'un prototype ABSTRACT SHAPE, hérité en Expanded dans les prototypes **Box** et **Circle**. Les autres prototypes n'ont pas de lien d'héritage, et sont utilisés pour le calcul et la gestion des collision (i.e **Vecteur2D**, etc).



« MATH »

- print_real r:REAL
- min a,b:REAL
- max a,b:REAL
- clamp value,min,max:REAL

« GRAVITY »

- next_frame shapes:FAST_ARRAY(ABSTRACT_SHAPE)
- handle_collision r:REPORT
- correction r:REPORT

« CONSTANTS »

- framerate:REAL
- dt:REAL // 1/framerate

« VECTOR2D »

- percent:REAL - slop:REAL
- gravity:VECTOR2D

« REPORT »

- create c,d:ABSTRACT_SHAPE, pen,dist:REAL, n:VECTOR2D, coll:BOOLEAN :REPORT
- create_no_collision c,d:ABSTRACT_SHAPE, pen,dist:REAL

, n:VECTOR2D, coll:BOOLEAN :REPORT

- gen_report a,b:ABSTRACT_SHAPE :REPORT
- gen_report_circle_circle a,b:ABSTRACT_SHAPE :REPORT
- gen_report_box_box a,b:ABSTRACT_SHAPE REPORT
- gen_report_circle_box a,b:ABSTRACT_SHAPE :REPORT
- gen_report_box_circle a,b:ABTRACT_SHAPE :REPORT
- SECTION PRIVATE
- make c,d:ABSTRACT_SHAPE, pen,dist:REAL, n:VECTOR2D, coll:BOOLEAN

+ x,y:REAL

- create a.b:REAL :VECTOR2D
- copy :VECTOR2D
- zeró :VECTOR2D
- set_x v:REAL set y v:REAL
- set_xy a,b:REAL
- length :REAL
- length_squared :REAL //pour éviter un sqrt si non nécessaire
- normalized :VECTOR2D
- dot_product other:VECTOR2D :REAL

REDEFITION OPERATEURS

'#' // Produit scalaire

SECTION PRIVATE

- make a.b:RFAL

FONCTIONNEMENT GLOBAL

De manière périodique (selon la variable *framerate*) définie dans **CONSTANTS**, la fonction *next_frame* se lance avec comme argument l'ensemble des *ABSTRACT SHAPE* présentes sur l'écran.

Cette fonction va calculer les nouvelles position des formes concernées en :

- Appliquant le vecteur gravity à leur vitesse
- Calculant les éventuelles collisions
- Appliquant le vecteur speed de chaque formes à leur vecteur position

Le calcul des collisions se fait comme suit : Chaque couple de *ABSTRACT_SHAPE* est envoyé dans la fonction *gen_report*, qui va, en fonction du type dynamique des arguments reçus, appeler une des quatre fonctions *gen_report X X*.

La fonction *gen_report_X_X* appelée va calculer les informations relatives à la collision nécessaires pour traiter cette dernière (*distance, pénétration, vecteur*, *etc*). Ces informations sont ensuite mises dans un **REPORT** et renvoyées.

Si le couple de **ABSTRACT_SHAPE** n'entre pas en collision, le rapport renvoyé est généré via la fonction *create_no_collision*, mettant le booléen *collided* du rapport à false, et aucun calcul relatif à la collision n'est effectué.

Les informations obtenues sont alors utilisées par la fonction handle_collision, qui va, via l'équation de conservation de l'énergie de Newton, calculer les vecteurs d'impulsion devant être appliqués aux deux formes.

Dans le cadre de la collision entre deux **CIRCLE**, la génération du rapport est simple. Dans le cas d'une collision **CIRCLE vs BOX**, le problème est résolu comme suit :

- Un point X est calculé. Ce point est le point sur la **BOX** étant le plus proche du **CIRCLE**. Le vecteur de la collision est donc ramené à un vecteur entre X et le centre du cercle. La suite du traitement est à partir de ce point semblable au traitement d'une collision **CIRCLE vs CIRCLE**.

REPARTITION DU TRAVAIL

Le travail a été reparti de manière équitable. Le codage de chaque classe a été divisé entre nous deux, et les bugs / problèmes algorithmiques ont été abordés ensemble.