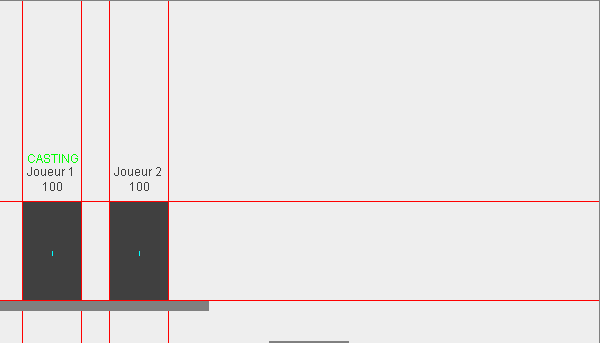
Principe et algorithmique d’éjection

Pour la mise en place de l’éjection d’un personnage lorsqu’il subit un dégât, nous allons commencer par montrer un schéma explicatif, puis un diagramme de classe, puis enfin des explications sur les changements à effectuer.

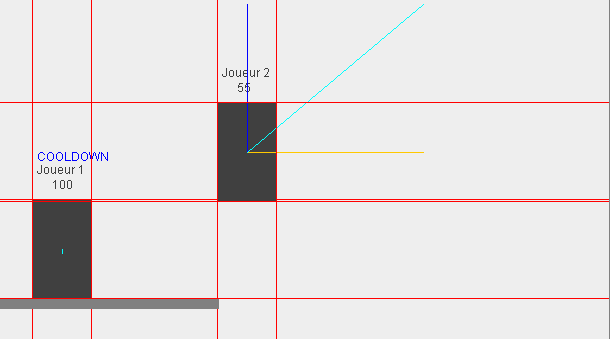
Tout d’abord, voyons différentes situations :

Le joueur 1 se prépare à éjecter le joueur 2 en le frappant :



Les 2 joueurs sont immobiles, et le joueur 1 cast son attaque.

Voici la situation quelques frames plus tard :

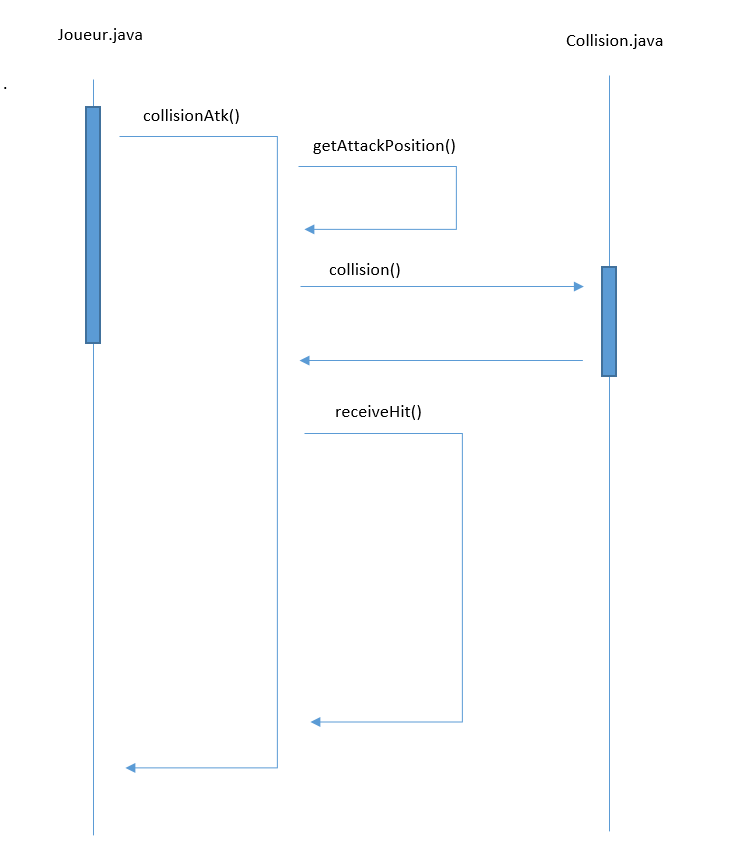


Le joueur 2 se fait éjecter par le coup du joueur 1.

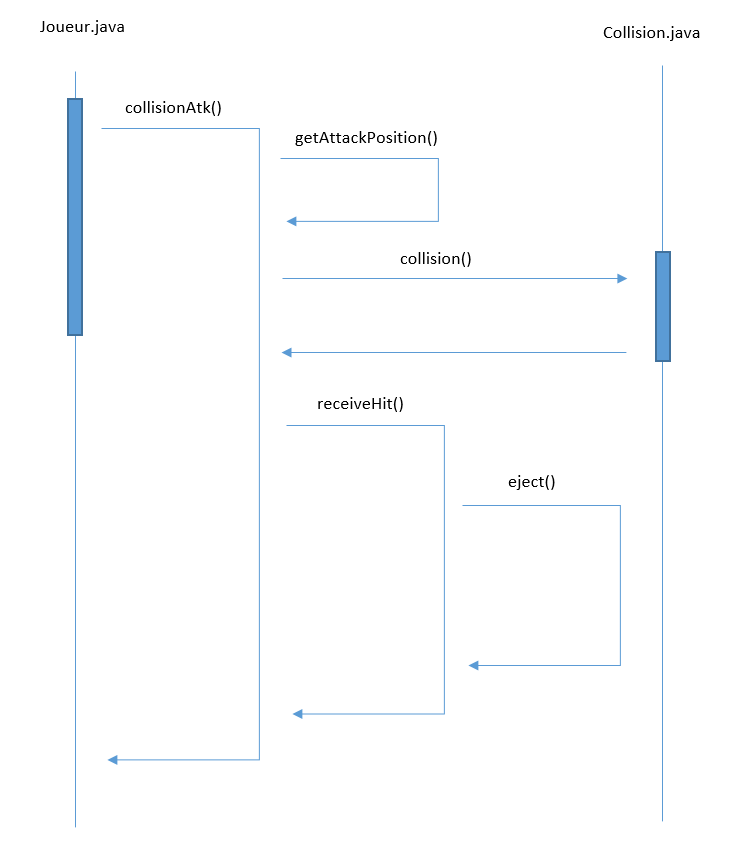
On peut voir que les forces semblent s’appliquer correctement au joueur touché.

Dans le cas où le joueur 2 bouge, la force appliquée est sensiblement la même.

Voici maintenant le diagramme de classe anciennement utilisé :



Puis le nouveau :



Explications :

Nous avons dû modifier les attaques des joueurs pour qu’elles appliquent une force au joueur qui sera touché par cette dernière.  
Nous avons donc seulement ajouté 2 attributs à la classe Attaque : powerX et powerY.

Il est important que prendre en compte le sens du joueur pour que l’éjection reste cohérente :

En effet, si une attaque possède un powerX de 200, et un powerY de 100, le joueur touché verra donc sa vitesse actuelle remplacée par ces valeurs, et ce, quelle que soit la direction de l’attaquant, donc il faut aussi penser à inverser le powerX lorsque l’attaquant attaque vers la gauche. S’il attaque vers la droite, il n’y a rien besoin de changer.

Après implémentation, nous avons pu constater que le moteur physique (collision, gravité…) fonctionnait toujours aussi bien, en dehors d’un cas :

Dans l’algorithme de gestion de la vitesse du joueur, si ce dernier désire aller dans une direction, et que son personnage à une vitesse supérieure à sa vitesse cap, sa vitesse sera remise à la limite cap.

(Si le joueur ne touche pas les commandes de direction, alors il perdra tout seul de la vitesse, comme prévu dans le moteur de déplacement.)

Cela a pour effet concret :

Si le personnage est éjecté vers la droite, et que le joueur souhaite aller vers la gauche, alors il perdra de la vitesse normalement, jusqu’à atteindre la vitesse nulle puis repartir dans le sens inverse. Pas de problème ici.   
Le souci survient lorsque le joueur souhaite aller vers la droite, lorsqu’il est éjecté vers la droite, (respectivement pour la gauche bien sûr), l’algorithme va donc appliquer la vitesse cap au joueur, ce qui aura pour effet de ralentir brusquement le joueur, cet effet étant indésirable.  
Nous avons donc imaginé résoudre ce problème en appliquant 1 état au joueur : NORMAL ou EJECTED.  
Dès que la fonction ejected() est appelée sur le joueur, alors ce dernier passe en EJECTED, puis lorsque sa vitesse passe en dessous de la vitesse cap, alors il redevient en état NORMAL.  
Ainsi, dans l’algorithme de la gestion des mouvements, le joueur souhaitant aller vers la droite, s’il est en état EJECTED, alors la limite ne s’appliquera pas, et seule l’inertie s’appliquera au joueur, pour le ralentir, jusqu’à ce que sa vitesse soit inférieure à celle de la vitesse cap, condition à laquelle le joueur pourra continuer à aller vers la droite.