Projet Spicy Invaders

# Tests unitaires

Afin d’assurer le bon fonctionnement du programme, nous avons élaborés une série de tests unitaires qui doivent être valide lors de chaque changement.

## CommonTests

1. GetEnemyExtremityTest()

Teste la méthode *GetEnemyExtremity()* qui a pour but d’identifier le/les ennemis aux quatre extrémités d’un essaim. Ici, nous créons un essaim de quatre ennemis puis nous regardons si la méthode nous ressort les bons ennemis pour chaque direction (top, bottom, left & right).

1. GetMaxLengthTest()

Teste la méthode *GetMaxLength()* qui a pour but de ressortir la longueur maximale d’une liste de strings. Nous le testons avec une liste de strings test contenant une chaîne *s4* clairement plus longue. Après exécution de la méthode, nous comparons *s4.Length* avec le résultat de *GetMaxLength()*

1. GetFrontLineEnemiesTest()

Teste la méthode *GetFrontLineEnemies()* qui a pour but de ressortir une liste de tous les ennemis se trouvant en tête d’un essaim. Ceci permet de savoir quels ennemis peuvent tirer en jeu. Nous créons alors un essaim de 5 x 3 ennemis. Nous comparons ensuite la dernière ligne de l’essaim avec le résultat de la méthode.

## ElementTests

1. ElementCtorTest()

Teste le constructeur de la classe mère *Element* en vérifiant que la variable assignée à un *new Element* ne soit pas nulle et soit de type *Element*.

1. LoadTest()

Teste la méthode *Load()* qui va inscrire dans un tableau en cascade *char* chaque caractère de l’élément aux bons emplacements en fonction de la position x et y qui sont attributs de la classe où le point (x, y) est le coin supérieur gauche de l’élément. Le test s’effectue avec un élément d’un caractère où nous comparons le caractère avec l’index correspondant dans le tableau de char.

## SpaceShipTests

1. SpaceShipCtorTest()

Teste le constructeur de la classe *SpaceShip* qui hérite de *Element.* Même procédure qu’au test 4.

1. MoveShipTest()

Teste la méthode qui déplace le vaisseau du joueur en changeant son attribut *\_x*. Vérification de la propriété *X* de *SpaceShip.* On compare la position du vaisseau au calcul engendré par les multiples déplacements effectués.

## EnemyTests

1. EnemyCtorTest()

Teste du constructeur de *Enemy.* Même procédé qu’au point 6.

1. ShootTest()

Teste la méthode *Shoot()* de *Enemy*. Nous allons comparer le nombre d’éléments dans une liste *projectiles* et le nombre de fois que la méthode *Shoot()* est appelée.

1. MoveTest()

Teste la méthode *Move()* de *Enemy*. Après quelques appels de la méthode nous comparons les positions x et y avec les résultats attendus lorsqu’on exécute cette méthode, obtenus par calcul explicite.

## BulletTests()

1. BulletCtorTest()

Teste du constructeur de *Bullet.* Même procédé qu’au point 8.

1. MoveTest()

Teste la méthode *Move()* des projectiles. Même procédé qu’au point 10.

1. CollidesWithTest()

Teste la méthode *CollidesWith()* qui vérifie si le projectile coïncide avec un *Element.* Pour cela nous créons deux ennemis, un qui se trouve au même emplacement que le projectile et un autre plus loin. Nous comparons ensuite les deux résultats. Le premier doit être *true* et le deuxième *false.*