Rapport de test groupe 3

# Les tests

Les tests que nous allons effectuer se porteront exclusivement sur certains modules, tout simplement car le temps ne nous permet pas de le faire sur tout notre projet.

Le premier module que nous avons testés est le module de génération d’une carte et donc des cases qui le compose.

## Test sur la création de la classe Case

Une case valide est une case comportant une position valide et un numéro de joueur valide si celle-ci comporte un personnage. Mais si le numéro de joueur n’est pas valide, c’est que le personnage qu’elle comporte doit être un PersonnageVide. Nous avons donc réalisés le test suivant.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| estPleine | estOccupe | numeroJoueur | Position | Création |
| False | Personnage | 0 | 2,7 | False |
| True | PersonnageVide | 0 | -2,3 | False |
| False | Personnage | 1 | 3,-1 | False |

Nous avons aussi choisi de tester sur un plus gros module, le module du personnage, cependant nous ne testerons pas la création étant donné du nombre d’erreurs possibles sur cette classe.

Nous avons donc testé plusieurs méthodes qui sont importantes car elles sont la base de notre programme.

## Test sur la méthode equals

La méthode equals de personnage doit être capable de reconnaitre si le personnage en paramètre est lui-même ou si c’est un ennemi sachant qu’un personnage peut être dans les deux équipes à la fois. Cependant il ne peut être qu’une seule fois par équipe, donc il suffit de comparer le nom et le numéro d’appartenance du personnage. Ainsi on en déduit le test suivant.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Personnage 1 | | Personnage 2 | | Equals |
| Nom | NumeroAppartenanceJoueur | Nom | NumeroAppartenanceJoueur |  |
| Riven | 1 | Riven | 1 | true |
| Riven | 1 | Riven | 2 | false |
| Riven | 1 | Thresh | 1 | false |

## Test sur la méthode retournerEtat

Cette méthode renvoi une chaîne de caractère correspondant à l’état dans lequel le personnage se trouve pour pouvoir s’en servir dans la méthode toString. Voici le test élaboré.

|  |  |
| --- | --- |
| Etat | Résultat attendu |
| NORMAL | « Normal » |
| EMPOISONNE | « Empoisonné » |

## Test sur la méthode rajouterVie

Cette méthode rajoute de la vie à un personnage mais si l’on lui rend trop de vie celui-ci doit juste se retrouver avec sa vie au maximum.

|  |  |  |  |
| --- | --- | --- | --- |
| Vie | Vie max | Rajout de vie | Vie attendu |
| 100 | 100 | 10 | 100 |
| 50 | 100 | 10 | 60 |

## Test sur la méthode enleverVie

Cette méthode enlève de la vie à un personnage mais si l’on lui enlève trop de vie celui-ci ne doit pas avoir une vie négative mais juste avoir 0 points de vie.

|  |  |  |
| --- | --- | --- |
| Vie | Vie enlevé | Vie attendu |
| 100 | 200 | 0 |
| 15 | 8 | 7 |

Pour réaliser ses tests, j’ai donc réalisé deux classes test, qui sont disponibles dans le projet et qui quand on les exécute ne rapporte aucune erreur signe que tous les tests réalisés sont correct et bien géré par l’application.