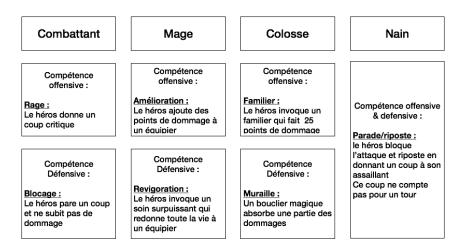


PROIET N° 3 : CRÉEZ VOTRE PREMIER IEU EN SWIFT // BONUS

Lien vers le projet

DESCRIPTION

Chaque personnage peut invoquer de manière aléatoire une compétence offensive de classe pendant qu'il attaque et une compétence défensive de classe pendant qu'il est attaqué.



Nota : la classe nain à une seule compétence de type défensive mais qui inflige des dégâts à son ennemi (elle a donc une propriété offensive aussi).

Ces compétences reposent sur des statistiques de classe : **agilité / force / intelligence / sortilège**. Chaque classe s'appuie sur 2 caractéristiques, celles-ci serviront à calculer la probabilité d'invoquer la compétence ainsi que les dégâts qu'elle infligera ou empechera.

Répartition des caractéristiques par classe:



Mécanique d'attribution des points entre les 2 caractéristiqueS

Au moment de la création d'un personnage, le joueur devra répartir 10 points entre les 2 caractéristiques propres à la classe du personnage.

IMPLEMENTATION

Dans la classe Character (en charge du concept de personnage et de ses actions), nous avons donc implémenter :

- 4 variables nommées agility/force/intelligence/wizardry.
- 2 méthodes : *offensiveClassAbility()* et *DefensiveClassAbility()* en charge de l'invocation de la compétence sur la base d'un nombre aléatoire basé sur la caractéristique principale de classe
- Des méthodes propres à à chaque compétence permettant d'appliquer les nouveaux dégâts du personnage. Ces dégâts sont calculés par algorithme basé sur les 2 caractéristiques de classe.
- Une méthode *CharacterAttackReset()* permettant de **faire retourner les dégâts du héros** à ses dégâts de classe initiaux à la fin du tour.

Dans la Class Player, en charge des actions du joueur pendant un tour nous avons fait appelle:

- À la méthode *CaracteristicAttribution()*: permettant de **répartir** les 10 points pendant la création du personnage et de bien **vérifier que la somme des 2 caractéristiques est égale à 10.**
- Aux méthodes *offensiveClassAbility()* et *DefensiveClassAbility()* dans la méthode *Turn()* pour implémenter cette fonctionnalité.