Doc API finale

# But du jeu

todo

# Fonctionnement de la map

## Passabilité

La map contient une matrice de taille variable (ex : 200x100) de cases. Chaque case peut être passable (du sol) ou non passable (un mur). Les zones passables et non passables sont définies dans la variable Passability de la classe MapView, qui est une matrice de booléens.

Ex : savoir si on peut marcher sur la map à la case (x=8, y=9) :

Views.MapView mapView = state.GetMapView();

bool canWalk = mapView.Passability[8][9];

Bien que la passabilité (le fait de pouvoir marcher) sur une case soit définie pour des positions « entières », les héros peuvent se déplacer sur des distances non entières. La passabilité de la map à la case (8.6, 7.5) est donc celle dans le tableau de passabilité en (8, 7) (on prend la troncature).

Les unités de distances sont aussi exprimées en cases, et correspondent à la distance euclidienne entre 2 points de la carte.

Ex : distance entre (1, 2) et (1.5, 3.5) = 1.58 cases.

## Vision

Chaque unité accorde la vision sur un certain nombre de cases. La vision est importante car sans elle, impossible de connaître la position des IAs ennemies sur la carte.

## Evènements présents sur la map

### Les creeps et les spawners

todo

### Les camps

todo

### Les mini-boss

todo

### Le big boss

todo

# Base de données du jeu

Au début de la partie, chaque IA récupère une copie d’éléments de base de donnée du jeu. Cette base de données comprends les modèles de sorts (Spells), ainsi que d’équipements.

Views.GameStaticDataView data = state.GetStaticData();

Cette base de données comprend les modèles de sorts (Spells), ainsi que d’équipements (Weapons, Armors, Boots, Enchants). Les modèles sont des classes qui décrivent le comportement des sorts / armes. Chacun de ces modèles possède un **identifiant** unique, auquel il faudra se référer lorsqu’on voudra dialoguer avec l’API. Exemple :

public List<int> ShopGetArmors()

Cette fonction retourne la liste des identifiants des modèles d’armures disponibles dans l’échoppe de votre équipe. Les modèles correspondants sont ceux qui ont le bon ID parmi les modèles contenus dans GameStaticDataView.

# Le héros : présentation

Le héros possède, outre ses statistiques, un rôle, un passif unique, plusieurs sorts actifs, ainsi que de l’équipement : une arme, une armure, des bottes. Et enfin un enchantement pour l’arme.

Tous ces équipements / sorts peuvent s’acheter auprès des marchands (Shops), en échange de Points d’Amélioration (PA) récupérés lors de certaines actions (détaillées après).

## Les statistiques

* Les points de vie (HP / MaxHP) : s’ils tombent à 0, le héros meurt et revient au combat quelque temps après.
* L’armure et la résistance magique : plus ces statistiques sont hautes, plus les dégâts reçus sont faibles. L’armure réduit les dégâts physiques, et la résistance magique les dégâts magiques. La formule utilisée est : dégâts bruts = dégâts \* 100 (100+ résistance).
* Les dégâts d’attaque (AttackDamage) : une valeur élevée augmente les dégâts infligés par les armes, ainsi que par certains sorts.
* Le pouvoir (AbilityPower) : une valeur élevée augmente les dégâts infligés par les sorts, ainsi que leur durée dans certains cas.
* La vitesse de déplacement (MoveSpeed) : elle est exprimée en cases/seconde.
* La vitesse d’attaque (AttackSpeed) : elle dénombre le nombre d’attaques possibles avec une arme dans une seconde. Ainsi, une AttackSpeed de 1.5 équivaut à 1.5 attaques par seconde, soit 3 attaques par seconde.
* La réduction de délai de récupérations (Cooldown Reduction / CDR) : elle est exprimée en pourcentages (0 -> 1) et permet de réduire le délai de récupération des sorts. Par exemple : si le héros possède 0.2 (20%) de CDR, un sort avec un cooldown de 10s peut être réutilisé 8s après son utilisation au lieu de 10s !
* La régénération de HP (Regen) : elle définit le nombre de HP régénérés passivement par seconde.

## Le rôle

* Mage : le mage octroie des soins, boucliers plus importants aux alliés, ainsi que des silences et roots plus longs.
* Combattant (Fighter) : Le combattant bénéficie de bonus de vitesse d’attaque, de dégâts d’attaques. La durée des effets d’invisibilité est aussi plus importante.
* Tank : Le tank bénéficie de bonus d’armure, de résistance magique, de vitesse de déplacements. La durée des stuns infligés par un tank est aussi plus importante.

# Les altérations d’état

Les altérations d’état sont décrites par la classe StateAlterationModelView. Tous les sorts, et les attaques avec l’arme appliquent des altérations d’état. Ces altérations sont de nature très variées (déterminée par le champ StateAlterationType Type) : il peut s’agir de dégâts (AttackDamage, MagicDamage, TrueDamage), de bonus/malus de statistiques (AttackDamageBuff, MagicDamageBuff, AttackSpeed, MaxHP, CDR, MagicResistBuff, ArmorBuff), de contrôles (Stun, Root, Silence, Blind, décrits ci-après).

Pour les bonus malus de statistiques, ainsi que les contrôles, le champ « Duration » de StateAlterationModelView indique la durée de l’effet.

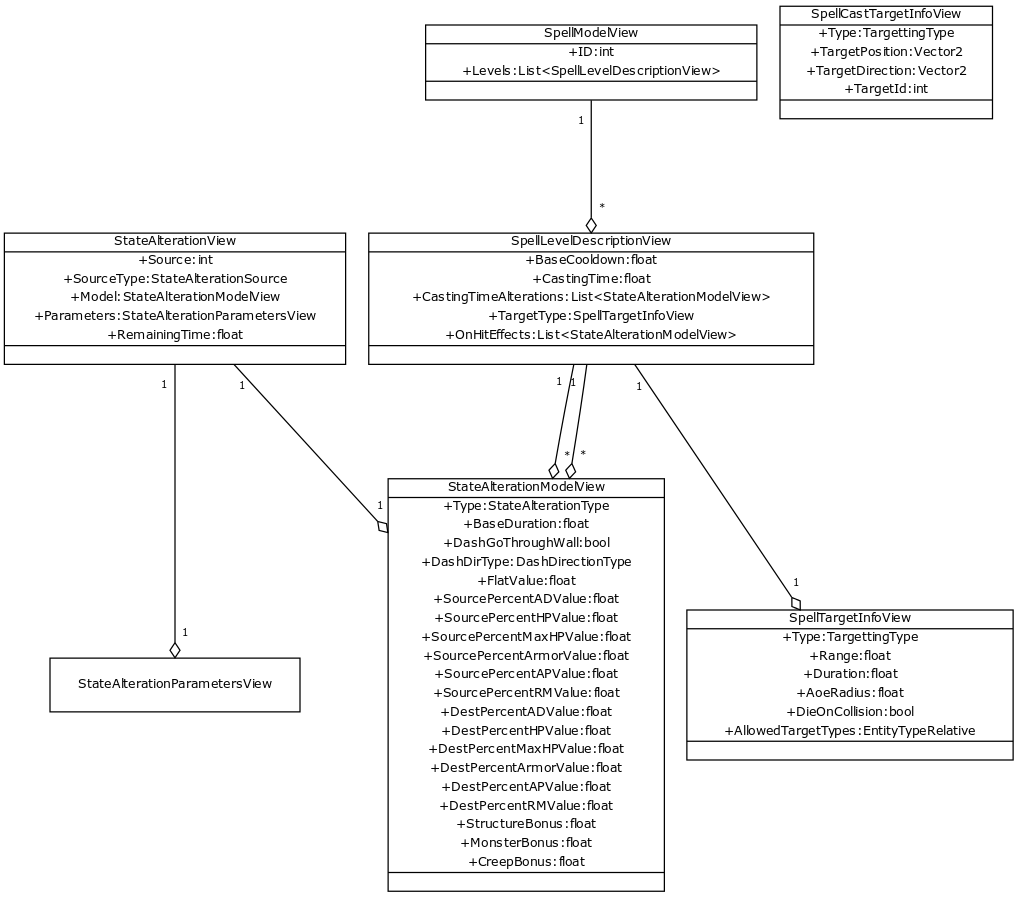
Pour les bonus/malus de statistiques et les effets de dégâts, le champ « FlatValue » indique la valeur du bonus / malus / montant de dégâts. Les champ « SourceADPercentValue », « SourceAPPercentValue » indiquent l’influence des points d’Attack Damage et Ability Power du héros.

Par exemple, une altération décrite ainsi : Type = StateAlterationType.MagicDamage, FlatValue = 5, SourceAPPercentValue = 1 => infligera un nombre de dégâts égal à 5 + 1\*AP du héros.

### Les effets de contrôle (CC)

* Root : la cible ne peut plus bouger pendant une durée définie.
* Silence : la cible ne peut plus lancer de sorts pendant une durée définie.
* Blind : la cible ne peut plus utiliser son arme pendant une durée définie.
* Stun : la combinaison des 3 CC du dessus.

# Les sorts : fonctionnement



Le héros possède des sorts auquels sont associés un modèle (SpellModelView). Ce modèle possède un identifiant unique (ID),. Les sorts sont améliorables et ont donc plusieurs niveaux (de 0 à 2). Ces niveaux sont représentés par l’attribut Levels de SpellModelView. A chaque niveau du sort est associé :

* **BaseCooldown** : temps entre 2 utilisations consécutives du sort
* **CastingTime** : un casting time (en secondes) :temps entre la prise de décision de lancer du sort, et l’exécution du sort
* **CastingTimeAlterations** : des altérations appliquées sur le lanceur pendant ce casting time.
* **OnHitEffects** : les effets appliqués sur la cible à l’impact du sort.
* **TargetType**: décrit le type de cible de ce spell (quelles entités peuvent être affectées), ainsi que :
  + **Type**: Direction, Position ou Targetted. Si position : le sort est lancée à une position donnée (il tombe du ciel), si Direction : le sort se déplace dans une direction donnée, si Targetted : le sort suit une entité en particulier.
  + **Range**: Si Type = Position : la distance max entre le lanceur du sort et la position du lancer, si type = Direction : la distance max parcourue par le projectile, si type = Targetted : la distance max entre le lanceur du sort et l’entité ciblée au moment du lancé.
  + **AllowedTargetType**: masque de bits indiquant les types d’entités pouvant être ciblées par le sort (EnnemyPlayer, Creep, Monster, ou des combinaisons telles que EnnemyPlayer | Monster etc…).
  + **(les autres champs sont un peu annecdotiques)**

Pour lancer un sort, on peut utiliser la fonction de la classe State :

bool UseMySpell(int spellId, SpellCastTargetInfoView target)

* **spellId** : le numéro du spell (0, ou 1 : correspondent aux spells du joueur !!).
* **target** : indique le ciblage du sort. Les champs à remplir dépendent du sort (et plus précisément du **TargetType** vu plus haut) : Type : Direction, Position ou Targetted (il faut que ce type soit en accord avec le type du sort à utiliser => sinon le serveur renvoie un code d’erreur).
  + **Si Type = Direction**: remplir le champ TargetDirection (direction du sort)
  + **Si Type = Position**: remplir le champ TargetPosition (position de destination du sort)
  + **Si Type = Targetted**: remplir le champ TargetID (id de l’entité à cibler)

## L’arme

L’arme fonctionne quasiment pareil que les sorts !

## Gagner des PA !!