**Voici les métriques à déterminer**

* + Points de vie des tours.
  + Points de vie d’un magicien (fixe ou variable)
  + Cadence de tir d’un magicien.
  + Dommages du magicien (pourrait varier selon les états).
  + Rythme de spawning des magiciens.
  + Nombre maximal de magiciens actifs par camp (recyclage)
  + Portée d'engagement des magiciens (pourrait varier selon les états).
  + Vitesse de déplacement des magiciens.
  + Forêt : effet sur la vitesse et les dommages subis.

**De plus certains éléments pourraient varier selon l’état (à votre choix)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **État** | **Régénération\*** | **Cadence de tir** | **Dommage** | **Portée**  **(unités)** | **Vitesse** |
| Normal | Normal / 0 en combat |  |  |  |  |
| Intrépide | Normal |  |  |  |  |
| Fuite | Normal |  | aucun |  |  |
| Planqué | X2 / 0 en combat |  |  |  | Toujours immobile |
| En sûreté | X3 |  | aucun |  | Toujours immobile |
| Last stand | Normal |  |  |  | Toujours immobile |

\* Valeurs présentées dans l’exemple. Vous pouvez modifier ces valeurs selon votre concept.

**Conseils techniques**

**Listes et collections**

Puisqu’en C# les listes et autres collections ne contiennent de que des références d’objets, vous pouvez vous en servir allègrement pour construire dynamiquement des listes d’objets

* Listes de cibles pour un magicien.
* Listes de tours actives (on retire les tours désactivées).
* Etc.

**Projectiles : communication entre le tireur et la cible.**

Si jamais dans votre projet vous employez des projectiles standards, sachez que vous pouvez de manière simple, si vous en avez besoin communiquer des informations entre le tireur et sa cible peu importe quelle sera sa cible.

* Le projectile garde une référence sur le GameObject de son tireur d’origine.
* À l’impact, il peut connaitre la cible par collision.
* À ce moment là si vous en avez besoin, les deux objets peuvent s’échanger de l’information.
* ATTENTION : il n’est pas garanti que vous ayez besoin de faire ceci, mais si jamais vous en avez besoin…

**Les lignes de couleur dans l’exemple.**

GameObject avec un composant LineRenderer.

* Attention : n’est pas le plus aisé des composants à manipuler pour avoir un bon résultat.
* Peut-être portée comme GameObject attribut par le magicien qui l’utilise (activé et désactivé au besoin).
* Attention : ne pas les créer et détruire au besoin. Activation et désactivation.
* Aura besoin du material et du sortingLayerName du magicien qui l’utilise.