**Animation des entités du jeu avec *Arcade***

**I/ Animation du joueur**

A ce stade, toutes les textures du joueur ont été chargées et la gestion du clavier (via les flèches directionnelles) a été effectué. Il ne reste donc qu’à s’occuper de l’animation du joueur en fonction de son déplacement et de son statut.

L’attribut *textures* gère **l’ensemble des textures** et est une **liste de listes**. L’attribut *texture* (qui vient de la classe *`Sprite`* gère la texture à afficher). Il faudra en conséquence chercher la bonne texture à afficher.

L’attribut *current\_texture\_indice* a le rôle **d’indice de la liste de textures** correspondant au **statut actuel du joueur** : il ne doit pas dépasser le nombre d’images de disponible !

L’attribut *status* indique le **statut** du joueur niveau ‘*animation’* (WALK\_LEFT, ATTACK\_DOWN etc.).

Voici les 3 cas à considérer :

* Cas 1 : Si le joueur **ne se déplace pas**, **pas d’animation**, il suffit de sortir de la méthode avec l’instruction *return* (Déjà écrit dans le squelette du programme).
* Cas 2 : Si le joueur **change** de **direction**, l’attribut status du joueur doit être modifié et l’attribut *current\_texture\_indice* est ramené à 0.
* Cas 3 : Si le joueur se **déplace** dans la **même direction**, l’attribut *current\_texture\_indice* est incrémenté de 1. S’il dépasse *len(self.textures[self.status])* alors il est ramené à 0 soit au début de l’animation.

|  |  |
| --- | --- |
| Cas 1 : Si les attributs *change\_x* et *change\_y* sont mis à **zéro** (considérés donc comme **False**), le joueur ne bouge plus donc pas d’animation.  Cas 2 : A **compléter**.  Cas 3 : A **compléter**.  Après les traitements, nouvelle texture à appliquer. | *Partie de la méthode update(self) pour l’animation* |

1/ **Compléter** le cas N°2. Vérifier qu’effectivement que l’affichage évolue si le joueur change de direction.

Aide : L’étude du **signe** des attributs *change\_x* et *change\_y* permet de déterminer le statut du joueur.

2/ (\*) **Tester** en appuyant sur deux directions simultanément et en continu : **si l’animation change à chaque déplacement, cela est un bug** : **proposer** une amélioration de l’algorithme.

3/ **Compléter** le cas N°3.

**Appeler le professeur pour validation.**

**II/ Animation des monstres**

**1/ Dans le fichier `mob.py`**

Tout comme pour le joueur, les textures sont chargées à ce stade. Il ne reste plus qu’à régler le **déplacement aléatoire** et **l’animation** des **monstres**.

|  |  |
| --- | --- |
| 4/ A **compléter** (astucieusement) si l’animation est trop rapide 😊.  5/ A **compléter**en comparant les attributs  *len(self.textures[self.status]* et *current\_texture\_indice*.  6/ A **compléter** en suivant l’exemple.  7/ **Mettre à jour** les attributs *center\_x* et *center\_y* en fonction de *change\_x* et *change\_y*.  8/ A **compléter**.  Aide : on considérera les valeurs des variables WALK\_DOWN, WALK\_LEFT, WALK\_UP et WALK\_RIGHT pour utiliser *randint(a,b)*.  9/ A **compléter**.  10/ A **compléter** en suivant l’exemple. |  |

**2/ Dans le fichier `main.py`**