**Création du GUI**

**I/ Dans le fichier *main.py***

|  |  |
| --- | --- |
| 1/ Dans la méthode *`setup(self)`* de la classe `*My\_Game*`, **écrire** à l’endroit indiqué le code à droite qui créé une instance de la classe `*Gui`.* |  |

2/ Dans la méthode *`on\_draw(self)`* de la classe `*My\_Game*`, **écrire** à l’endroit indiqué l’instruction permettant à l’attribut *gui* d’appeler la méthode *`draw(self)`*.

**II/ Dans le fichier *gui.py***

Voici la description du constructeur de la classe *`Gui`* :

|  |  |
| --- | --- |
| La **création** de l’attribut *game* permet d’accéder à tous les **attributs** de la classe `*My\_Game*`, par exemple, l’instruction *self.game.player* permet d’accéder aux attributs du joueur 😊.  Menu général du jeu : bouton ***Quit*** ici.  Affiche à l’écran les caractéristiques du joueur.  Affiche celles du monstre cliqué (s’il y en a un), sinon rien ne doit s’afficher. |  |

Voici la description de la méthode *create\_menu\_bar(self)* qui affiche le menu du jeu. Pour l’instant, il n’y a que le bouton ***Quit*** qui sera inséré (mais on peut imaginer d’autres options) :

|  |  |
| --- | --- |
| **Création** de la box qui contiendra tous les éléments du menu (sous forme de boutons ici) : l’instruction *vertical=False* permet de l’orienter à l’horizontale.  **Création** d’un bouton *Quit*, instance de la classe `*QuitButton*`.  **Ajout** du bouton précédent dans le menu  **Ajout** de ce menu à l’interface : il sera en bas (*bottom*) à droite (*right*) de l’écran. |  |

**Zoom** sur la dernière instruction :



Elle permet deux choses :

* **d’ajouter** le menu du jeu via l’argument *self.menu\_bar*. On remarquera que le nom du paramètre *child* est bien choisi puisqu’une GUI se représente sous forme d’un arbre,
* **d’ancrer** ce menu en bas (en abscisses avec *anchor\_x*) et à droite (en ordonnées avec *anchor\_y*) ce menu.

Placement des menus en fonction des valeurs de *anchor\_x* et *anchor\_y* :

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Les menus sont placés de manière **relatives** (grâce à des données directionnelles) et non **absolues**, ce qui permet de s’adapter à la taille de la fenêtre du jeu. | *Placement du menu en fonction des valeurs de* ***anchor\_x*** et de ***anchor\_y***   |  |  |  | | --- | --- | --- | | ***anchor\_x***: *left*  ***anchor\_y***: *top* | ***anchor\_x***: *center*  ***anchor\_y*** : *top* | ***anchor\_x*** : *right*  ***anchor\_y*** : *top* | | ***anchor\_x***: *left*  ***anchor\_y*** : *center* | ***anchor\_x***: *center*  ***anchor\_y*** : *center* | ***anchor\_x***: *right*  ***anchor\_y*** : *center* | | ***anchor\_x***: *left*  ***anchor\_y*** : *bottom* | ***anchor\_x***: *center*  ***anchor\_y*** : *bottom* | ***anchor\_x***: *right*  ***anchor\_y*** : *bottom* | |

3/ **Recopier** la méthode précédente *create\_menu\_bar(self)*.

Voici la description de la méthode *setup(self)* qui permet d’afficher le menu interactif avec l’utilisateur et la description du joueur :

|  |  |
| --- | --- |
| **Création** de l’interface qui sera rattachée à la fenêtre de jeu.  Appel à la méthode précédente créant le menu du jeu (ici composé d’un bouton *Quit*). |  |

4/ **Recopier** la méthode précédente *setup(self)*.

|  |  |
| --- | --- |
| 5/ **Ecrire** l’instruction suivante dans la méthode *draw(self)*. |  |

**Exécuter** le programme, un bouton ***Quit*** doit s’afficher en bas à droite et fonctionner correctement.

Remarque : ne pas hésiter à modifier le bouton *Quit* (taille, texte …) et son placement sur la carte.

**Appeler le professeur (si besoin)**

Voici la description imagée de la méthode *create\_player\_box(self)*:

|  |  |
| --- | --- |
| ***player\_texture\_button***  **v\_box\_2**  ***player\_box*** |  |

6/ **Recopier** la méthode *create\_player\_box(self)* et **compléter** la méthode *create\_clicked\_mob\_box(self, clicked\_mob)* en suivant la même progression.

7/ **Compléter** désormais la méthode *setup(self)* pour qu’elle prenne en compte le descriptif du joueur :

|  |  |
| --- | --- |
| **Création** de l’interface qui sera rattachée à la fenêtre de jeu.  **Création** du menu du jeu (ici composé d’un simple bouton *Quit*).  **Création** du descriptif du joueur.  **Ancrage** en haut à gauche du descriptif du joueur. |  |

**Exécuter** le programme, le descriptif du joueur doit s’afficher en bas à droite en plus du bouton *Quit* (qui doit toujours fonctionner correctement 😊 ).

Remarque : ne pas hésiter à modifier les ***paramètres*** affichés du joueur ou même l’interface.

**Appeler le professeur pour validation**

**III/ Adaptation entre la GUI et les interactions du joueur avec le RPG**

Jusqu’à présent, la GUI affichée était statique et indépendante des actions du joueur. Pour être efficace, une GUI doit bien sûr s’adapter aux choix de l’utilisateur tout en respectant également un souci de design et de cohérence.

Le jeu RPG proposé permet d’afficher les **caractéristiques d’un monstre ciblé** (pour vérifier s’il est attaquable ou pas par exemple) : pour cela, il va falloir notamment relier « *clic de souris* » et « *entité ciblée*».

Exemple ici :



**1/ Dans le fichier *gui.py***

Voici le descriptif de la méthode *add\_clicked\_mob(self, clicked\_mob)* :

|  |  |
| --- | --- |
| **Création** de l’interface qui sera rattachée à la fenêtre de jeu.  **Création** du menu du jeu (ici composé d’un simple bouton *Quit*).  **Création** du descriptif du joueur.  **Création** des caractéristiques de l’entité cliquée (paramètre *clicked\_mob*).  **Création** d’une box horizontale qui accueillera le descriptif du joueur et celle de l’entité cliquée.  **Placement** de la box précédente dans le manager et à haut / gauche de la fenêtre de jeu. |  |

1/ **Recopier** le code précédent dans la méthode *add\_clicked\_mob(self, clicked\_mob)*.

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

La classe *`My\_Game`* propose des **méthodes** permettant au **joueur** **d’interagir** avec l’écran que ce soit par le clavier ou la souris.

La méthode *on\_mouse\_press(self, button, x, y, modifier)* permet de gérer les **clics** de souris.

Voici le descriptif de ses paramètres :

* *button* indique le **type** de bouton de la souris cliqué (*right*, *left*, *center*).
* *x, y* sont les **coordonnées relatives** du clic effectif de la souris.
* *modifier* n’a pas d’intérêt à ce stade*.*

Voici le code qui sera à recopier et compléter dans le programme :

|  |  |
| --- | --- |
| **Coordonnées** absolues du clic. Il faudra les déterminer (voir plus bas).  **Récupère** les **entités cliquées** de l’attribut *self.sprites\_list* (principe de la boundingbox).  En cas de **clic** sur une entité, **affichage** de ses caractéristiques sinon on utilise la gui de base. |  |

Détermination des coordonnées absolues *new\_x* et *new\_y*.

MAP\_WIDTH

SCREEN\_WIDTH

SCREEN\_HEIGHT

C *(new\_x, new\_y)* ***(x, y)***

MAP\_HEIGHT

P ***(center\_x, center\_y)***

***(0,0)***

2/ Sachant que le **joueur est toujours centré** sur l’écran, **déterminer** les coordonnées *(new\_x, new\_y)* représentées par la flèche bleu en pointillés en fonction de *x, y, player.center\_x, player.center\_y, SCREEN\_WIDTH* et *SCREEN\_HEIGHT.*

3/ **Proposer** une amélioration du programme empêchant les caractéristiques de s’afficher deux fois si l’on clique dessus.

4/ (\*) **Proposer** une autre interface.

**Appel au professeur pour validation**