Classe image :

|  |  |  |
| --- | --- | --- |
| Nom de la classe : | Les fonctions de la classe | Description + type des arguments |
|  |  |  |
| image | \_\_init\_\_(self, path) | path : **string** qui donne le chemin d’accès pour l’image ( = self.picture) |
| Variables utiles :   * self.picture (variable contenant l’image) | load(self) -> None | Charge l’image ( = self.picture) dans pygame |
| load\_with\_transparence(self) -> None | Charge l’image( = self.picture) dans pygame avec de la transparence |
| load\_with\_color\_filter(self, color) -> None | color : **tuple**  Charge l’image ( = self.picture) dans pygame en rendant la couleur spécifiée transparente |
| resize(self, size) -> None | size**:** **tuple**  Permet de changer la taille de l’image ( = self.picture) au format de size |
| Infos(self) -> None | Permet d’afficher la taille de l’image ( = self.picture) |
|  |  |  |
| Button | \_\_init\_\_(self, rect, colors, active\_color, text\_list, active\_text) -> class ‘button’ | rect: **pygame.Rect**   * rectangle du bouton   **colors** : **liste de tuples**   * couleurs que le buton peut prendre   active\_color : **tuple**   * couleur active   text\_list : **liste de string**   * liste de texte que le bouton peut afficher   active\_text : string   * texte « actif » pouvant être affiché |
| Variables utiles :   * self.rect   (le rectangle qui représente le bouton)   * self.colors   (une liste de tuples qui représente les couleurs pouvant être prises par le bouton)   * self.active\_color   (la couleur actuelle du bouton)   * self.text\_list   (une liste de string qui représente les textes pouvant être affichés sur le bouton)   * self.active\_text   (le texte actuel du bouton) | change\_color(self) -> None | Change ‘active\_color’ pour la couleur suivante dans ‘colors’ |
| change\_text(self) -> None | Change ‘active\_text’ pour la couleur suivante dans ‘text\_list’ |
| get\_color(self) -> tuple | Récupère la couleur actuelle du bouton |
| get\_text(self) -> string | Récupère le texte actuel du bouton |
| is\_colliding(self, position) -> bool | position : **tuple**   * tuple qui représente une position : (x,y)   Renvoi ‘True’ si les coordonnées de position est dans le bouton (= self.rect)  Sinon renvoi ‘False’ |
| detect\_collision(self, position) -> bool | Renvoi ‘True’ si les coordonnées de position est dans le self.rect  Sinon renvoi ‘False’  [+] change la couleur et le texte (passe aux valeurs suivantes) |
|  |  |  |
| Icon | \_\_init\_\_(self, img, rect, screen, mark\_color, background\_color, mark\_thickness, mark\_wait\_color, Entity) -> icon | img : **pygame.Surface**   * l’image de l’icon. NB : image.picture est de ce type…   rect : **pygame.Rect**   * le rectangle définissant la taille et position de l’icon   screen : **pygame.Surface**   * représente l’ écran sur lequel on peut tout afficher   mark\_color : **tuple**   * la couleur du contour de sélection de l’icon   background\_color : **tuple**   * couleur du fond d’écran   mark\_thickness : **int**   * épaisseur du trait de contour de sélection de l’icon   mark\_wait\_color : **tuple**   * couleur du contour de l’icon lorsqu’elle n’est plus sélectionnée mais qu’un autre groupe d’icon est en train d’être selectionné   Entity : **Entity ou False**   * permet de créer un lien entre une Entity et l’icon ou pas |
| Variables utiles :   * self.rect   (le rectangle qui représente l’icon) | is\_colliding(self, position) -> bool | position : **tuple**   * tuple qui représente une position : (x,y)   Renvoi ‘True’ si les coordonnées de position est dans l’icon (= self.rect)  Sinon renvoi ‘False’ |
| mark\_rect(self, color) -> None | color : **tuple**   * permet d’afficher color en tant que couleur de contour de l’icon |
| draw(self) -> None | Permet de dessiner l’icon |
|  |  |
| Icons | \_\_init\_\_(self, icon\_list): | Icon\_list : **list of icon**   * permet de regrouper les icon dans un groupe |
| Variables utiles :   * self.activated\_icon\_index   le numéro **(int**) de l’icon « active »   * self.number\_of\_icons   le nombre (**int**) d’icon dans la liste   * self.icon\_list   **liste d’icon** regroupée ensemble par la classe icons | draw(self, selected\_icon\_should\_being\_draw) -> None | selected\_icon\_should\_being\_draw : **Bool**   * dessine toutes les icon * La valeur de l’argument permet de dessiner ou non le contour de sélection « actif » sur une icon |
| detect\_collision(self, mouse\_pos) -> bool | mouse\_pos : **tuple**   * True si le self.rect d’une icon contient les coordonnées de mouse\_pos   Sinon renvoi False   * Permet aussi de changer, si il y a une collision, le numéro de l’icon sélectionnée ( = self.activated\_icon\_index) |
| mark\_selected\_icon(self) -> None | Permet d’afficher le contour de selection « actif »de l’icon dans la liste self.icon\_list à l’index self.activated\_icon\_index |
| mark\_selected\_wait\_icon(self) -> None | Comme « mark\_selected\_icon(self) » mais permet d’afficher la couleur de ‘’pause’’ |
| change\_to\_right(self) | Permet de changer le self.activated\_icon\_index vers la gauche de la self.icon\_list |
| change\_to\_left(self) | Permet de changer le self.activated\_icon\_index vers la droite de la self.icon\_list |
|  |  |  |
| Main | \_\_init\_\_(self, screen\_size, background\_color, icon\_mark\_color, mark\_thickness) -> class main | screen\_size : tuple   * Taille de l’écran   background\_color : tuple   * couleur du fond de la fenêtre   icon\_mark\_color : tuple   * Pour l’instant pas encore implémenter dans le code, mais correspondra à la couleur de sélection par défaut   mark\_thickness : int   * Epaisseur du trait de sélection des icon |
|  | Regarder le code (relativement bien commenté) |  |
|  |  |  |
|  |  |  |
|  |  |  |