Description du diagramme de classe

Modèle :

Métier :

Positions possède quatre attributs de type *int* qui correspondent aux quatre positions x1, x2, y1 et y2 grâces auxquelles on peut retrouver les positions des quatre sommets du carréJoueur avec les couples (x1,y1), (x1,y2), (x2,y1) et (x2,y2). Elle permet de « géolocaliser » l'élément.

La classe **Element** représente un élément du jeu (**CarréJoueur**, **Mur**, **Bouton**), elle possède un attribut de type Positions.

La classe **Bouton** est un élément particulier qui possède un attribut appuyé de type *boolean*. Cette classe représente un bouton qui peut avoir deux états distincts : appuyé ou relâché. Chaque bouton a les mêmes dimensions qu'un carréJoueur.

La classe **Sortie** possède un attribut ouvert de type *boolean* également. Elle représente un élément sortie qui peut être soit ouvert soit fermé. La sortie est ouverte quand tous les boutons sont appuyés.

La classe **Niveau** est une classe qui représente un niveau, elle possède une liste de murs et une liste de boutons mais aussi une sortie et bien sûr un carréJoueur.

La classe **Monde** est pourvu d'une liste de niveaux.

Gestion :

Le **Manager** fait le lien entre la Vue et le Modele, il commence par charger le monde et mapper les touches, il crée un déplaceur et une liste de collisionneurs. Ensuite, il a une méthode traiterTouche() qui est appelé dans le code behind de la Fentre du jeu quand une touche du clavier est enfoncé par l'utilisateur. Cette méthode traite les touches de directions ou la touche pour appuyer sur un bouton. Enfin la méthode finMouvement() s'occupe de savoir si le niveau est terminé grâce à la classe GestionSortie.

Le **Collisionneur** gère les collisions entre les éléments et s'occupe de donner la position de la collision la plus proche du carréJoueur.

Le **CollisionneurBouton** regarde pour chaque bouton du niveau et en fonction de la direction que prend le carréJoueur dans son déplacement si ce dernier se trouve sur le trajet d'un des boutons. Si c'est le cas, il va renvoyer au manager la position de ce bouton pour que le déplaceur puisse amener le carréJoueur jusqu'à celle-ci de sorte que le carréJoueur se trouve sur le bouton.

Le **CollisionneurMur** quant à lui vérifie pour chaque mur et toujours en fonction de la direction que le carréJoueur se trouve sur le trajet d'un des murs pour envoyer in fine la position « collé au mur » au déplaceur. Le déplaceur va ensuite transposer le carréJoueur à la position juste avant le mur.

Le **DéplaceurCarré** a pour responsabilité de déplacer le carré jusqu'à la position donné par le collisionneur.

GestionSortie a une méthode ouvrirSortie() qui regarde si tous les boutons du niveau sont appuyés, elle a aussi une méthode sortieElement dont le but est de savoir si le carréJoueur se trouve collé à la sortie pour pouvoir prévenir le manager que le niveau est complété.

La classe **ObsTempsBoutonConcret** implémente ObsTempsBouton qui implémente ObsAppuye. Cette classe a une méthode updatebouton() qui lance un chrono. Elle a aussi une méthode updatechrono() qui désactive le bouton, cette méthode est appelé par le chrono quand le temps est expiré.

Chrono :

ChronoRefreashConcret a une méthode beep() appelé toutes les 1000/60 millisecondes par la méthode run(). La méthode beep() va appeler la méthode déplacer de tous les déplaceurs de la liste du chrono (pour l'instant le jeu comporte un seul déplaceur). Le déplaceur va donc déplacer son élément toutes les 16,7 millisecondes tant que c'est possible c'est à dire tant que l'élément n'excède pas la position renvoyée par le collisionneur.

ChronoBouton a une méthode beep() qui appelle la méthode updatechrono() de tous ses observateurs. Elle se spécialise ensuite en trois sous-classes ChronoBoutonLent, ChronoBoutonMoyen et ChronoBoutonCourt qui possèdent une méthode run() qui va appeler la méthode beep() après avoir appeleé la méthode sleep(n) avec n différent pour chaque sous-classe.

Data :

La classe **Stub** génère un monde avec ses niveaux et les attributs de ses niveaux.

Launch :

La classe **Launcher** a pour responsabilité d'instancier le manager des vues et le manager du modele