Projet PFA : Bomberman

Il nous a été demandé de réaliser un projet en programmation fonctionnelle avancée, un jeu selon le modèle ECS. C'est pourquoi j'ai choisi de me lancer sur un bomberman, jeu respectant les contraintes.

Pour commencer j'ai décidé de commencer par construire la base de mon projet, l'arène. J'ai donc construit les bords, ainsi que les blocs à l'intérieur représentant les blocks incassables auquel le joueur se confronterait. J'ai ensuite tenté de programmer les murs cassables pour le coup, en me basant sur la classe block. J'ai défini une variable valant true si le mur est explosible par une bombe. Pour cela j'ai essayé de créer une matrice représentant l'occupation de mes cases (x y) et dans laquelle je mets la valeur à true quand j'initialise un nouvel objet, permettant par ailleurs de ne pas créer deux objets au même emplacement. Cependant celle-ci ne semble pas fonctionner correctement, j'ai des objets, des blocks qui se superposent. J'ai également tenté de mettre en place le système d'explosion, je peux poser mes bombes, elles explosent mais elles ne provoquent rien. J'ai tenté de créer un tableau comportant mes objets explosibles en paramètres mais je n'ai pas pu l'exploiter comme je le voulais.

Description de l'organisation du code :
Entités :
block (les différents blocks, destructibles ou non, représentant les blocks en jeu

Bombe : la gestion des bombes ainsi que leur explosion

IA : l'ia se déplaçant dans le jeu, non abouti à cause de la non maitrise des collisions, j'aurais aimé la faire repartir dans une des 3 autres directions à chaque fois qu'il y a collision avec un mur ou bloc

Joueur : représente le joueur

Systèmes :

Collisions (les collisions donc)

Draw : l'ajout des éléments à notre fenêtre

Forces : la représentation des différentes forces agissant sur nos objets

Move : déplacement des objets

Utils :
rect et vector permettent de définir des rectangles et des vecteurs, ce qui a été utile dans nombre de situations, pour la collision, pour la box (dont je me sers comme hitbox, dans le cas de la bombe…) et les vecteurs pour de nombreux calculs

Component_def :
la définition de classes utiles et leurs méthodes

Game.ml : le principal du code, là où le principal est géré, l'initialisation de la fenêtre…

System_defs : les différents modules pour gérer l'affichage, la collision, les forces et les mouvements.

Globalement j'ai rencontré bon nombre de problème que j'aurais pu tenter de régler avec + de temps, le principal et le + impactant étant la gestion des collisions entre les différents objets, mon projet se soldant donc sur un échec. La programmation fonctionnelle n'est vraiment pas aussi simple à appréhender pour ce genre de projet exigeant, sans compter la documentation moins répandue.

Une manière de gérer la destruction des briques serait de créer de nouveaux components que l'on pourrait nommer explode (statut explosé de l'objet) (initialisé à false) , ainsi qu'un component pexpl (component définissant la capacité d'un objet à exploser un autre objet) que l'on implémenterait dans nos classes et qui passerait à explode à true depuis le system collisions si l'objet concerné a son attribut expl (statut explosable) à true et si l'autre objet provoquant la collision a son attribut pexpl à true. Ensuite, l'on pourra créer des méthodes des classes concernés s'activant lorsque le statut explode d'un objet passe à true. (il faudra donc dans le fichier game ajouter les listes correspondant dans lequel on ajouterait les blocks/joueurs/ia puis créer des méthodes update dans chaque classe pour gérer le retrait de l'objet en collision (dont le statut explode est donc passé à true) avec l'explosion.