Rapport Space Invaders

I. Introduction sur le projet :

Dans la première étape, j'ai fait l'install_in du defender en utilisant une image .gif puis dans la classe Game, j'ai ajouté la fonction **keypress(self, event)** qui sert à déplacer le defender horizontalement dans les deux sens en utilisant les flèches gauche et droite et en respectant la largeur du canvas pour que le defender reste toujours visible et toujours dans le cadre du jeu.

La touche espace, fait appelle à la fonction **fire(self, canvas)** dans la class Defender, dans cette fonction, le defender ne peux pas tirer plus que le nombre autorisé (**self.max_fired_bullets**), si cette condition est respecté, on appelle la classe Bullet avec un **self.shooter = defender** , afin de différencier entre les balles des Alien et celles du defender. Il y a affichage d'une balle dans le jeu et l'ajout de cette dernière dans une liste des balles tirés (**self.fired_bullets**).

La fonction **move_bullet(self,canvas)** sert à déplacer les balles tirés par le defender verticalement vers le haut, et les supprimées du canvas et de la liste (**self.fired_bullets**) si les coordonnés de la balle dépasse le bord haut (y<0).

Pour l'affichage du flotte des Alien, j'ai utilisé la fonction **install_in(self, canvas)** dans la classe Fleet qui utilise la fonction **install_in(self,canvas,x,y)** de la classe Alien pour afficher chaque Alien de la flotte et qui retourne l'objet dans la liste (**self.aliens_fleet**).

Le déplacement du flotte d'Aliens se fait grâce à la fonction **move_in(self, canvas)** du classe Fleet qui déplace l'image de l'Alien (self.id) horizontalement et verticalement, en utilisent la fonction **move_in(self, canvas, alien, dx, dy)** du classe Alien.

La gestion des Alien qui sont touchés par les balles du defender se fait par la fonction **manage_touched_aliens_by(self,canvas,defender,score)** , s'il y a intersection de la balle et l'Alien, on appelle la fonction qui montre une animation courte de l'effet de la collision **effet_boom(self,canvas,projectile)**, et on supprime le dessin d'Alien et celui de la balle dans la fonction **touched_by(self,canvas,projectile)**,. Et on modifie le score selon une logique précise, et enfin on supprime l'Alien de sa liste (**self.aliens_fleet**) et la balle aussi de sa liste (**self.fired_bullets**).

II. les améliorations proposées :

Barre de score et de vies disponibles :

Dans la classe Game, j'ai ajouté une fonction **affiche_score(self)** qui affiche le score dès le lancement du jeu, et à chaque fois qu'un Alien est mort le score augmente, le l'affichage est actualisé grâce à la requête qui se trouve dans la fonction **animation(self)** (**self.score_actuel.set("score : "+str(self.score.get_points()))**)

Le logique du score est basé sur le fait que le plus rapide tu tues tous les Aliens le plus de points tu gagnes, j'ai utilisé une variable d'instance dans la classe Score nommé **self.delai_recommended** initialisé à 120s, en utilisant la formule de la fonction **refresh_score(self, end_time)** , on gagne moins de points si on se rapproche du délai recommandé et si on le dépasse on gagne que 10 point par Alien tué.

Pour le defender, il a 3 vies disponibles (**self.lifes**) qui sont affiché avec la fonction **display_defender_lifes(self, canvas)**, s'il est touché par les balles des Alien il perd une vie, et on supprime une vie du canvas, cela est géré avec la fonction **manage_defender_touched_by(self, canvas, fleet)**.

Gestion des joueurs

Le jeu se termine si : tous les Alien sont tués, l'ordonnée du bord bas de la flotte des Alien est supérieure à l'ordonnée du bord haut du defender, ou finalement si le defender n'as plus de vie disponible.

A la fin de la partie, si la partie est perdu, on appelle la fonction **player_lost(self)** qui passe la variable **self.score.winning** à False pour montrer que le joueur a perdu, et appelle la fonction **saveFile(self)** par la suite, qui enregistre le score et toutes les information sous format **json** dans un fichier nommé "LastScore.json", et ensuite l'ajout dans un fichier ou on enregistre tous les scores dans un fichier nommé "AllResults.json" à l'aide de la classe **Resultats** , et affiche un message sur l'écran que le joueur a perdu.

Pareil si le joueur a gagné sauf qu'on appelle cette fois la fonction **player_won(self)** et on passe la variable **self.score.winning** à **True** pour montrer que le joueur a gagné, et finalement on affiche un message sur l'écran qui montre que le joueur a gagné.

Les Alien peuvent tirer

La fonction **install_aliens_bullets(self, canvas)** dans la class Fleet permet l'installation des balles des Aliens par hasard en utilisant le **randint** et la fonction **aliens_fire(self, canvas, x, y)** pour respecter que les Aliens ne tirent pas plus que le nombre autorisé (**self.max_fired_bullets**) et aussi pour qu'il y aura un écart d'une seconde entre chaque tire (**self.temps_ecart**).

Le dessin des balles se fait avec la fonction **install_aliens_bullets(self, canvas, x, y)** de la calasse Bullet, et son mouvement vers le bas se fait à l'aide de la fonction **move_aliens_bullets(self, canvas)** qui supprime le dessin et la valeur de la liste, si elles déplace le bord bas du jeu.

Le Defender peut s'abriter derrière des bunkers de Protection

Dans la classe **LesBunckers** et à l'aide de la fonction **install_in(self,canvas)** on ajoute des bunckers où le defender peut s'abriter pour ne pas être touché par les balles des aliens, les bunckers arrêtent les balles des Aliens mais laisse passer les balles du defender. Si les balles des alien touche un des bunckers, il y a l'intervention de la fonction **manage_bunckers_statut(self,canvas,fleet)** qui diminue la santé du buncker (**self.health**) de -25. Et affiche un effet de collision grâce a la fonction **boom_effect(self, canvas, x, y)** et par la suite elle suprime le dessin de la balle et sa valeur de la liste. Ensuite, en appelle la fonction **refresh_buncker_img(self, canvas)** de la classe Buncker pour changer l'image selon l'état du buncker.

Si le buncker est touché 4 fois avec les balles des aliens, et son self.health devient 0, on le supprime et on enlève sa valeur de la liste (**self.bunckers_liste**).

Si le buncker est touché par un Alien, dans la fonction **buncker_touched_by_alien(self, canvas, fleet)** on supprime l'alien et le buncker ,et on enlève leurs valeurs de la liste (**self.bunckers_liste** et **aliens_fleet**).

Ajout d'animations

Pendant le déplacement des Alien, ils sont animés avec la fonction **animation_aliens_img(self, canvas)** qui change l'image des Aliens après chaque seconde (temps - self.animation_start_time) grâce à la fonction **refresh_img(self, canvas)** de la classe Alien.

III. Propositions supplémentaires :

Dans La classe **SpaceInvaders** le jeu ne peut être lancé qu'après que le joueur entre son nom, une fois c'est fait, en clique sur la touche entrer et on appelle la fonction **lancer (event)** qui désactive le champs de saisie et lance le jeu.

Quand le jeu se lance, dans la fonction **start_animation(self),** on appelle la fonction **welcome(self)** qui affiche un message de bienvenue plus le nom du joueur.

Dans le classe Defender, j'ai ajouté une fonction **collision_between_bullets(self, canvas, fleet)** pour pouvoir arrêté les balles des Aliens avec les balles du defender. Au cas de la collision des deux balles, on affiche un effet visuel de la collision.

J'ai ajouté l'effet de collision pour la colision entre les balles et le buncker, entre le defender et les balles des Aliens, et entre les Aliens et les bunckers.

IV. Difficultés envisagées :

J'avais un problème dans la 1ere étape pour le déplacement de la flotte d'Alien mais j'ai pu corriger le problème.

Et dans la 2eme partie, j'avais un problème avec les bunckers car je n'avais pas la possibilité d'accéder à self.health pour pouvoir le diminuer si une balle le touche, mais j'ai changé le retourne de l'install_in qui était return self.id par return self pour pouvoir accéder à toutes les variables d'instances de chaque buncker. Le même problème avec le changement d'image si le buncker est touché j'ai fait la fonction dans la classe LesBunckers mais après j'ai changé sa place et je l'ai mis dans la classe Buncker et comme ça j'ai résolu ce problème.