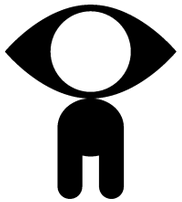

DÉPARTEMENT
Sciences Informatiques

UE LIBRE
INITIATION AU DEVELOPPEMENT DE JEU D'ARCADE EN PYTHON

Romuald GRIGNON – Marc LEMAIRE

CY - Université Cergy Paris
Institut Sciences et Techniques
Département Informatique

Année universitaire 2020-2021

STORIES

Critères d'acceptation généraux

- Pour chaque story qui va suivre, vous devrez vérifier, en plus des critères d'acceptation de la story, que les critères suivants sont également bien respectés :
  - Tenir à jour une liste des variables que vous utilisez (noms, types, structures, rôles)
  - Tenir à jour une liste des fonctions crées avec leurs rôles et l'ordre d'appel au sein de chaque groupe de fonctions (création, mise à jour, affichage, …)
  - Votre code doit être commenté
  - Votre code doit utiliser des constantes pour tous les réglages possibles (positions initiales des éléments, vitesse de déplacement, vitesse de rotation, …)
  -

Story #1 : Création du projet

- En tant que participant au module de développement de jeu d'arcade en Python,
- Je veux créer la base du projet
- Afin d'avoir une base de travail initiale pour développer mon jeu.
- Critères d'acceptation :
  - Récupérer les fichiers sources du projet initial depuis la page du cours [/!\ Lien /!\]
  - Ouvrir le projet depuis l'IDE PyCharm et créer l'environnement virtuel
  - Vérifier la configuration permettant de lancer le projet
  - Lancer le projet et voir une fenetre sur fond noir apparaître
- Commentaires :
  - A ce stade, votre environnement de travail est prêt pour le développement de votre jeu

DEPLACEMENT DU PERSONNAGE

Story #2 : Création du personnage

- En tant que participant au module de développement de jeu d'arcade en Python,
- Je veux créer mon premier sprite fixe
- Afin de savoir comment afficher les éléments graphiques de mon jeu.
- Critères d'acceptation :
  - Définition de constantes qui vont contenir la largeur et la hauteur de votre sprite
  - Création d'une fonction initCharacter() qui va créer la variable contenant le sprite du personnage
  - Création d'une fonction drawCharacter() qui s'occupera de dessiner à l'écran le sprite du personnage
  - Appel des fonctions précédentes dans les fonctions déjà fournies, respectivement **setup()** et **draw()**
  - Affichage de l'image du personnage à l'écran lors de l'exécution du programme
- Commentaires :
  - Utilisez les fonctions mises à votre disposition pour créer un sprite fixe
  - Pour le moment, vous pouvez placer votre personnage n'importe où à l'écran

Story #3 : Déplacement du personnage

- En tant que participant au module de développement de jeu d'arcade en Python,
- Je veux gérer les commandes au clavier
- Afin de pouvoir déplacer mon personnage à l'écran.
- Critères d'acceptation :
  - Création de 2 variables contenant la position X et Y du personnage
  - Création d'une fonction updateCharacter() qui va positionner le centre du sprite aux coordonnées X et Y précédentes
  - Appel de la fonction précédente depuis la fonction **update()**
  - Gestion de l'appui des touches FLECHES GAUCHE/DROITE pour décrémenter/incrémenter la valeur de la position X depuis la fonction **onKeyEvent()**
- Commentaires :
  - A ce stade, en appuyant de manière répétitive sur les touches FLECHES GAUCHE/DROITE vous pouvez déplacer le sprite du personnage horizontalement à l'écran

Story #4 : Déplacement amélioré

- En tant que participant au module de développement de jeu d'arcade en Python,
- Je veux améliorer la gestion des touches de déplacement
- Afin de pouvoir obtenir un déplacement fluide du personnage.
- Critères d'acceptation :
  - Création de 2 variables booléennes moveLeft/moveRight qui seront mises à jours en fonction de l'appui et relâchement des touches FLECHES GAUCHE/DROITE
  - Modification de la fonction updateCharacter() pour décrémenter/incrémenter la coordonnée X du personnage si les variables moveLeft/moveRight sont à la valeur VRAI
- Commentaires :
  - A ce stade, le personnage se déplace de manière fluide dans une direction lors d'un appui long sur la FLECHE
  - Si les 2 touches FLECHES sont appuyées en même temps le personnage ne se déplace plus

Story #5 : Limitation du déplacement

- En tant que participant au module de développement de jeu d'arcade en Python,
- Je veux limiter le déplacement de mon personnage aux bords de l'écran
- Afin de garder mon personnage toujours visible.
- Critères d'acceptation :
  - Modification de la fonction updateCharacter() afin de faire saturer la variable de coordonnée X du personnage à 0 ou SCREEN_WIDTH
  - Possibilité de faire saturer la position X avant le bord de l'écran pour des raisons esthétiques et garder ainsi le personnage toujours complètement visible à l'écran, plutôt que de le voir coupé en deux en arrivant sur le bord de l'écran
- Commentaires :
  - A ce stade le personnage se déplace de manière fluide lors des appuis sur les FLECHES et se bloque sur les bords de l'écran même si les touches restent enfoncées

Story #6 : Utilisation du gamepad

- En tant que participant au module de développement de jeu d'arcade en Python,
- Je veux utiliser un contrôleur de jeu USB
- Afin de pouvoir déplacer mon personnage autrement qu'avec le clavier.
- Critères d'acceptation :
  - Modification de la fonction **onAxisEvent()** afin de récupérer la notification de modification de l'axe X du contrôleur de jeu
  - Modification des variables moveLeft/moveRight précédemment utilisées dans la fonction **onKeyEvent()** en fonction de la valeur de l'axe
- Commentaires :
  - A ce stade, le personnage se déplace de manière fluide, et se stoppe sur les bords de l'écran, en utilisant au choix le clavier et/ou n'importe quel contrôleur de jeu branché en USB sur la machine

ANIMATIONS DU PERSONNAGE

Story #7 : Affichage du fond d'écran

- En tant que participant au module de développement de jeu d'arcade en Python,
- Je veux afficher une image de fond
- Afin de créer un décor pour mon personnage.
- Critères d'acceptation :
  - Création d'une fonction initBackground et d'une variable contenant le sprite du fond d'écran
  - Création d'une fonction drawBackground qui affiche le sprite à l'écran
  - Appel des fonctions précédentes dans les fonctions **setup()** et **draw()** respectivement
  - L'appel de drawBackground doit se faire avant l'appel de drawCharacter sans quoi le personnage se retrouvera derrière le fond et ne sera pas visible par l'utilisateur
- Commentaires :
  - A ce stade, nous pouvons voir et déplacer notre personnage comme avant, avec un fond d'écran derrière lui

Story #8 : Animation du personnage (1/3)

- En tant que participant au module de développement de jeu d'arcade en Python,
- Je veux utiliser des sprites animés
- Afin de rendre les déplacements de mon personnage plus vivants.
- Critères d'acceptation :
  - Création d'une variable pour stocker le sprite de l'animation du personnage quand il se déplace (run)
  - Modification des fonctions initCharacter(), updateCharacter() et drawCharacter() si besoin
- Commentaires :
  - A ce stade, votre personnage affiche en boucle l'animation chargée, qu'il soit en mouvement ou non

Story #9 : Animation du personnage (2/3)

- En tant que participant au module de développement de jeu d'arcade en Python,
- Je veux utiliser des sprites animés
- Afin de rendre les déplacements de mon personnage plus vivants.
- Critères d'acceptation :
  - Création d'une nouvelle variable pour stocker le sprite de l'animation du personnage au repos (idle)
  - Modification des fonctions initCharacter(), updateCharacter() et drawCharacter() si besoin
  - Choix du sprite à afficher entre celui de déplacement et celui de repos en fonction de l'état des variables de déplacement
- Commentaires :
  - A ce stade, votre personnage affiche une animation quand il est au repos, et une deuxième animation différente de la première lorsqu'il se déplace
  - Pour le moment, votre personnage est toujours tourné du même côté quelle que soit son sens de déplacement

Story #10 : Animation du personnage (3/3)

- En tant que participant au module de développement de jeu d'arcade en Python,
- Je veux utiliser des sprites animés
- Afin de rendre les déplacements de mon personnage plus vivants.
- Critères d'acceptation :
  - Création en double des sprites animés pour le repos et pour le déplacement en utilisant le paramètre « flipH » qui permet de créer un sprite avec un effet miroir horizontal
  - Création de nouvelles variables pour stocker les sprites avec effet miroir
  - Modification des fonctions initCharacter(), updateCharacter() et drawCharacter() si besoin
  - Creation et stockage d'une variable contenant l'information « dernière direction » (ici, gauche ou droite)
  - Lors de l'update, faire un choix entre les 4 sprites animés (repos gauche, repos droite, déplacement gauche, déplacement droite) en fonction des variables de déplacement et de direction
- Commentaires :
  - A ce stade, votre personnage affiche la bonne animation entre le repos et la course, vers la gauche ou la droite, en fonction de la direction donnée. Il conserve la même direction une fois au repos

CREATION DES ITEMS

Story #11 : Génération des items

- En tant que participant au module de développement de jeu d'arcade en Python,
- Je veux créer des items de manière aléatoire
- Afin d'avoir un objectif de collecte dans mon jeu.
- Critères d'acceptation :
  - Création d'une fonction initItems() pour initialiser une liste vide (stockage des futurs items)
  - Création d'une fonction generateItem() qui va créer un sprite animé de l'item, fixer aléatoirement une vitesse de déplacement pour cet item, et fixer aléatoirement une position tout en haut de l'écran sur toute la largeur de l'écran. Cette fonction sera appelée lors de l'appui sur une touche du clavier (ex : touche ESPACE)
  - Stockage du sprite, de la vitesse et de la position initiale dans un tuple, et stockage du tuple dans la liste des items
  - Creation d'une fonction drawItems() qui va parcourir la liste et afficher les items 1 par 1 sur l'écran
- Commentaires :
  - A ce stade, vous pouvez créer des items et les visualiser à des abscisses aléatoires en haut de l'écran en appuyant sur la touche espace

Story #12 : Déplacement des items

- En tant que participant au module de développement de jeu d'arcade en Python,
- Je veux déplacer les items du jeu
- Afin de visualiser leur mouvement.
- Critères d'acceptation :
  - Création d'une fonction updateItems() qui va modifier la position de chaque item en fonction de sa vitesse
  - Seule la position Y de chaque item sera modifiée afin de les faire progresser vers le bas de l'écran
  - Vous utiliserez l'information de vitesse stockée lors de la génération de l'item
- Commentaires :
  - A ce stade, vous pouvez créer des items à des abscisses aléatoires en haut de l'écran en appuyant sur la touche espace et les visualiser en train de descendre vers le bas de l'écran, chacun avec une vitesse différente

Story #13 : Destruction des items

- En tant que participant au module de développement de jeu d'arcade en Python,
- Je veux détruire les items du jeu lorsqu'ils sortent de l'écran
- Afin de n'avoir en mémoire que les items réellement affichés sur l'écran.
- Critères d'acceptation :
  - Modification de la fonction updateItems() pour supprimer de la liste les items qui sont descendus en dessous d'une ordonnée Y définie
  - Initialisez la valeur de seuil Y avec une valeur visible à l'écran afin de bien visualiser la destruction de chaque item
- Commentaires :
  - A ce stade, vous pouvez créer des items à des positions aléatoires sur l'écran en appuyant sur la touche espace et les visualiser en train de descendre vers le bas de l'écran, chacun avec une vitesse différente
  - Vous pouvez visualiser chaque item disparaître dès qu'il passe en dessous du seuil d'ordonnée Y défini

Story #14 : Génération automatique des items

- En tant que participant au module de développement de jeu d'arcade en Python,
- Je veux pouvoir générer automatiquement les items à collecter
- Afin d'avoir un environnement de jeu autonome.
- Critères d'acceptation :
  - Création d'une constante contenant l'intervalle de temps entre 2 items
  - Création d'un compteur de temps qui va s'incrémenter du 'deltaTime' à chaque appel de la fonction update()
  - Test du compteur de temps pour générer un item dès qu'il dépasse la constante définie précédemment (appel à la fonction generateItem() codée dans une story précédente)
  - Toutes ces évolutions sont à intégrer dans les fonctions liées aux items
- Commentaires :
  - A ce stade, vous voyez des items se créer régulièrement dans le temps, descendre chacun avec sa vitesse, et se détruire en arrivant au bas de l'écran

SCORE et VIE

Story #15 : Collisions

- En tant que participant au module de développement de jeu d'arcade en Python,
- Je veux créer des fonctions pour vérifier la collision de surfaces simples
- Afin de pouvoir détecter si le personnage « attrape » les items.
- Critères d'acceptation :
  - Création d'une fonction qui prend en entrée les positions et les tailles de 2 rectangles et qui renvoie un booléen si ces 2 rectangles sont en collision
  - Création d'une fonction qui prend en entrée les positions et les tailles de 2 cercles et qui renvoie un booléen si ces 2 cercles sont en collision
- Commentaires :
  - A ce stade, votre programme ne contient pas d'améliorations fonctionnelles mais vous disposez de fonctions utilitaires pour tester les collisions

Story #16 : Score

- En tant que participant au module de développement de jeu d'arcade en Python,
- Je veux ajouter une fonctionnalité de score
- Afin de créer un objectif à ce jeu d'arcade.
- Critères d'acceptation :
  - Ajout d'une variable contenant le score du joueur (et une fonction initScore() pour l'initialiser)
  - Ajout d'une fonction drawScore() qui va afficher le score au format texte
  - Création d'une fonction checkPlayerItemCollisions() qui va boucler sur chaque item et verifier la collision avec le joueur.
  - Vous pouvez choisir entre des collisions de type rectangle ou cercle
  - Si l'item est en collision avec le joueur, la variable de score doit être incrémentée, ET l'item correspondant doit être supprimé de la liste
- Commentaires :
  - A ce stade, vous pouvez voir votre score affiché à l'écran et s'incrémenter à chaque fois que le personnage est en contact avec un item. L'item correspondant doit également disparaître de l'écran

Story #17 : Points de vie

- En tant que participant au module de développement de jeu d'arcade en Python,
- Je veux ajouter des points de vie au personnage
- Afin de mettre une limite à ce jeu d'arcade.
- Critères d'acceptation :
  - Ajout d'une variable contenant les points de vie du personnage (et une fonction initLife() pour l'initialiser).
  - Ajout d'une fonction drawLife() qui va afficher les points de vie au format texte
  - Création d'une fonction checkItemLost() qui va boucler sur chaque item pour vérifier si il est passé en dessous d'une limite horizontale
  - Si un item passe en dessous de cette limite, il sera supprimé de la liste, ET le nombre de points de vie sera décrémenté
- Commentaires :
  - A ce stade, vous avez un affichage des points de vie, qui diminuent quand le personnage rate un item et l'item correspondant disparait

GESTION de l'ETAT du jeu

Story #18 : Etats du jeu

- En tant que participant au module de développement de jeu d'arcade en Python,
- Je veux ajouter une gestion de l'état du jeu
- Afin de pouvoir définir précisément le début et la fin d'une partie.
- Critères d'acceptation :
  - Ajout d'une variable d'état qui peut prendre 3 valeurs (*ready*, *running*, *finished*)
  - Création de plusieurs fonctions qui serviront à séquencer les états du jeu. Par ex : isReady(), isRunning(), isFinished(), startGame(), stopGame(), rewindGame()
  - L'appui sur la touche ENTREE du clavier, ou sur le bouton MENU d'un contrôleur permet de passer le jeu de l' état *ready* à *running* (startGame()) ou de passer de l'état *finished* à *ready* (rewindGame()). La fonction rewindGame() réinitialise toutes les variables à leurs valeurs de départ
  - Le passage des points de vie à 0, change l'état de *running* à *finished* (stopGame())
  - Le personnage ne peut attraper les items uniquement dans l'état *running*
- Commentaires :
  - A ce stade, vous pouvez lancer plusieurs parties les unes à la suite des autres sur demande mais rien n'indique clairement dans quel état du jeu vous êtes

Story #19 : Affichage de messages

- En tant que participant au module de développement de jeu d'arcade en Python,
- Je veux afficher des messages à chaque état du jeu
- Afin d'indiquer au joueur ce qu'il doit faire.
- Critères d'acceptation :
  - Affichage d'un message dans l'état *ready* pour indiquer au joueur d'appuyer sur la touche ENTREE ou le bouton MENU pour démarrer une partie
  - Faire clignoter le message de départ (afficher ou non le message en fonction du temps système)
  - Affichage d'un message de type « game over » dans l'état *finished* afin de laisser le temps au joueur de voir son score final
- Commentaires :
  - A ce stade vous avez un jeu qui intègre les contrôles du personnage, l'affichage des différents éléments de jeu, la condition de fin d'une partie et le séquencement des différents états du jeu

SONS

Story #20 : Sons

- En tant que participant au module de développement de jeu d'arcade en Python,
- Je veux utiliser des sons lors des différentes actions
- Afin d'ajouter une dimension sonore au jeu.
- Critères d'acceptation :
  - Création de variables contenant les sons pour la capture et la perte d'un item
  - Lecture du son de capture au moment où la collision entre le personnage et l'item est valide
  - Lecture du son de perte d'un item quand ce dernier disparaît sous le seuil horizontal défini précédemment
- Commentaires :
  - A ce stade, vous avez ajouté une dimension sonore à votre jeu

MEILLEURS SCORES

Story #21 : Meilleurs scores

- En tant que participant au module de développement de jeu d'arcade en Python,
- Je veux ajouter une fonctionnalité de meilleurs scores
- Afin de pouvoir mémoriser mes records de jeu.
- Critères d'acceptation :
  - Lecture d'un fichier pour récupérer les meilleurs scores. Si le fichier n'existe pas, il est créé
  - Création d'une liste en mémoire contenant les N meilleurs scores (remplie depuis le fichier lu)
  - Création d'une fonction pour sauvegarder les meilleurs scores (écriture de la liste vers le fichier)
  - Affichage des meilleurs scores, soit dans l'état *ready*, soit l'état *finished*
- Commentaires :
  - A ce stade, vous êtes capable de mémoriser sur votre disque dur, les meilleurs scores de toutes les parties jouées sur votre machine
  - Vous disposez donc d'un jeu minimaliste, néanmoins complet en termes d'affichage, de sons, de contrôles, de gestion des états, de conditions de fin, et de sauvegarde. Pour les stories suivantes, vous pouvez choisir l'ordre, car elles ne sont destinées qu'à améliorer la base du jeu existant. Vous pouvez aussi proposer d'autres évolutions qui vous paraissent plus pertinentes

AMELIORATIONS
(affichage)

Story #A1 : Items tournants

- En tant que participant au module de développement de jeu d'arcade en Python,
- Je veux faire tourner les items
- Afin d'améliorer l'ambiance visuelle du jeu.
- Critères d'acceptation :
  - Modification de la propriété « angle » des sprites
- Commentaires :
  - A ce stade, vous voyez les items tourner en même temps qu'ils chutent

Story #A2 : Effets de particules *burst* (1/2)

- En tant que participant au module de développement de jeu d'arcade en Python,
- Je veux ajouter des effets de particules quand le personnage attrape un item
- Afin d'améliorer l'ambiance visuelle.
- Critères d'acceptation :
  - Création d'une liste contenant les générateurs de particules de type *burst*
  - Creation d'un générateur et ajout dans la liste à chaque fois qu'un item est attrapé par le personnage
  - Mise a jour de chaque générateur dans la fonction **update()**
  - Affichage de chaque générateur dans la fonction **draw()**
  - Suppression d'un générateur de la liste quand il a terminé son travail
- Commentaires :
  - A ce stade vous devriez voir apparaître des particules a chaque fois que le personnage attrape un item
  - En fonction des paramètres de votre générateur, et de l'image que vous avez choisie, le rendu peut être différent de ce que vous attendiez : testez plusieurs combinaisons

Story #A3 : Effets de particules *burst* (2/2)

- En tant que participant au module de développement de jeu d'arcade en Python,
- Je veux ajouter des effets de particules quand le personnage rate un item
- Afin d'améliorer l'ambiance visuelle.
- Critères d'acceptation :
  - Création d'une liste contenant les générateurs de particules de type *burst*
  - Création d'un générateur et ajout dans la liste à chaque fois qu'un item est raté par le personnage
  - Mise a jour de chaque générateur dans la fonction **update()**
  - Affichage de chaque générateur dans la fonction **draw()**
  - Suppression d'un générateur de la liste quand il a terminé son travail
- Commentaires :
  - A ce stade vous devriez voir apparaître des particules a chaque fois que le personnage rate un item
  - En fonction des paramètres de votre générateur, et de l'image que vous avez choisie, le rendu peut être différent de ce que vous attendiez : testez plusieurs combinaisons

Story #A4 : Effets de particules *continuous*

- En tant que participant au module de développement de jeu d'arcade en Python,
- Je veux ajouter des effets de particules quand le personnage se déplace
- Afin d'améliorer l'ambiance visuelle.
- Critères d'acceptation :
  - Création d'un générateur de particules de type *continuous*
  - Mise a jour du générateur dans la fonction **update()** (avec la mise a jour de la position également)
  - Affichage du générateur dans la fonction **draw()**
- Commentaires :
  - A ce stade vous devriez voir apparaître des particules là où le personnage se déplace
  - En fonction des paramètres de votre générateur, et de l'image que vous avez choisie, le rendu peut être différent de ce que vous attendiez : testez plusieurs combinaisons

Story #A5 : Barre de vie

- En tant que participant au module de développement de jeu d'arcade en Python,
- Je veux afficher mes points de vie sous forme graphique
- Afin d'améliorer l'ambiance graphique du jeu.
- Critères d'acceptation :
  - Création d'un sprite pour le cadre
  - Créations de plusieurs sprites pour chaque bâton de la barre de vie
  - Affichage du nombre de bâtons en fonction de la variable contenant les points de vie
  - La couleur de chaque bâton peut être modifiée (ex : gradient du rouge au vert en passant par le jaune)
- Commentaires :
  - Ici vous avez le même comportement de jeu qu'auparavant, seul l'affichage des points de vie a été modifié

Story #A6 : Parallax (1/2)

- En tant que participant au module de développement de jeu d'arcade en Python,
- Je veux afficher un décor de fond de type parallax
- Afin d'améliorer l'ambiance graphique du jeu.
- Critères d'acceptation :
  - Création d'une liste de sprites contenant les calques de fond
  - Duplication de chaque calque pour faire les raccords visuels
  - Création d'une fonction qui va modifier la position des calques en fonction de la position du personnage
  - Affichage des différents calques dans le bon ordre + affichage de certains calques devant le personnage si besoin
- Commentaires :
  - Ici vous avez le même comportement de jeu qu'auparavant, seul l'affichage du fond a été modifié
  - La position des items par rapport au décor de fond ne sera donc plus cohérent pour le moment

Story #A7 : Parallax (2/2)

- En tant que participant au module de développement de jeu d'arcade en Python,
- Je veux modifier la position des items lors de l'utilisation du parallax
- Afin d'avoir une cohérence dans la position des éléments à l'écran.
- Critères d'acceptation :
  - Modification de la fonction qui met a jour la position des items afin de modifier l'abscisse. L'abscisse sera modifiée en fonction du décalage du parallax
- Commentaires :
  - Ici vous avez le même comportement de jeu qu'auparavant, seul la position des items pourra être modifiée sur l'axe X, et ils pourront disparaître de l'écran momentanément si vous vous éloignez d'eux
  - N'oubliez pas de sauvegarder l'abscisse de référence de chaque item car c'est cette valeur qui servira dans tous les calculs

Story #A8 : Nuages

- En tant que participant au module de développement de jeu d'arcade en Python,
- Je veux ajouter des nuages qui se déplacent dans le fond
- Afin d'améliorer l'ambiance visuelle.
- Critères d'acceptation :
  - Création de listes, fonctions, …, comme pour la génération des items
  - Les nuages sont créés d'un côté de l'écran, et sont détruits de l'autre coté
- Commentaires :
  - Le jeu comporte maintenant des éléments en mouvement de plus qu'avant mais les différentes mécaniques de jeu ne sont pas modifiées

AMELIORATIONS
(mécaniques de jeu)

Story #M1 : Générateur Pseudo-Aléatoire

- En tant que participant au module de développement de jeu d'arcade en Python,
- Je veux créer mon propre générateur Pseudo-Aléatoire
- Afin de comprendre comment fonctionne ce type de générateur.
- Critères d'acceptation :
  - Création d'une variable contenant la valeur pseudo-aléatoire
  - Définition des constantes du générateur (coef, offset, modulo)
  - Utilisation de votre générateur au lieu du module *random* de Python
- Commentaires :
  - Votre jeu ne devrait pas être foncièrement modifié d'un point de vue utilisateur. Tout comme avec le module *random* de Python, vous pouvez fixer une graine fixe ce qui vous donnera toujours la même séquence de génération de vos items, ce qui peut être un critère de mécanique de jeu
  - Vous trouverez un peu plus d'informations sur les générateurs pseudo-aléatoires ici : https://fr.wikipedia.org/wiki/G%C3%A9n%C3%A9rateur_congruentiel_lin%C3%A9aire

Story #M2 : Déplacement rapide

- En tant que participant au module de développement de jeu d'arcade en Python,
- Je veux pouvoir déplacer mon personnage plus rapidement
- Afin d'ajouter une nouvelle mécanique de jeu.
- Critères d'acceptation :
  - Lors de l'appui sur la barre ESPACE ou sur n'importe quel bouton d'un contrôleur, le personnage voit sa vitesse de déplacement augmentée.
  - La vitesse reste élevée tant que la touche ou le bouton reste enfoncé
  - Quand la vitesse du personnage est élevée, il est impossible de changer de direction : même le fait de relâcher les flèches de direction, ou le stick du contrôleur, ne vont pas stopper le personnage
- Commentaires :
  - Cette nouvelle mécanique de jeu vous permettra d'attraper des items plus éloignés, mais tant qu'elle est active, elle vous empêche de changer de direction, il faudra donc relâcher le bouton, changer de direction et rappuyer sur le bouton pour repartir dans l'autre sens à toute vitesse

Story #M3 : Trajectoires des items

- En tant que participant au module de développement de jeu d'arcade en Python,
- Je veux pouvoir modifier le déplacement des items
- Afin d'ajouter une nouvelle mécanique de jeu.
- Critères d'acceptation :
  - Chaque item va partir du bas de l'écran, monter, puis une fois en haut, va redescendre comme c'est le cas actuellement
  - Le personnage ne peut attraper les items que lorsqu'ils descendent
  - Les items ont une apparence différente lorsqu'ils montent (ex : transparence plus importante, taille réduite légèrement)
  - Les items sont affichés sur un plan différent lorsqu'ils montent ou descendent (en cas d'utilisation de plusieurs plans de fond, cf. Parallax)
- Commentaires :
  - Cette nouvelle mécanique de jeu vous permettra d'anticiper la trajectoire des items et donc de planifier plus facilement l'ordre dans lequel il faut les attraper pour ne pas en rater