Martin Brönnimann

RAPPORT NSI

Projet Pacman

Avant le projet

Je n'ai pas créé les sprites des fantômes en premier afin de me concentrer sur les différents points et aspects du jeu, comme les déplacements, la création du décor, les fantômes…

Création du jeu

Je n'ai pas fait les menus en premier, mais le jeu et ses mécaniques. Pour ça j'ai commencé à afficher le décor à partir d'un fichier **csv**. Ce fichier csv contient des chiffres avec lesquels j'ai pu déterminer leur nature (murs, pac gums, super pac gums, etc..). J'ai ensuite affiché ses éléments sous la forme d'une grille de jeu, chaque case possède une longueur calculée en fonction de la hauteur de la fenêtre divisé par la longueur du premier élément de la liste de liste contenant la grille de jeu additionné avec 0.3 afin que tout soit uniforme avec la hauteur de la fenêtre.

$$\frac{height}{longueur\ grille[0]} + 0,3$$

Je me suis ensuite occupé du joueur (pacman) et de ses déplacements. Pour cela j'ai utilisé le schéma donné par la prof, celui-ci étant de vérifier si pacman se trouvait bien au milieu d'une case avant de vérifier si le déplacement demandé était valide pour enfin déplacer pacman.

Pour cela j'ai créé une fonction **est_au_centre** qui calcule la position de pacman sur une case et retourne un booléen qui permet de savoir si pacman est au centre ou non.

$$X\ ou\ Y \times longueur\ d'une\ case + \frac{longueur\ d'une\ case}{2}$$

La fonction **deplacement_valide**, elle permet de bien vérifier la présence de murs sur le chemin du joueur. Celle-ci retourne un booléen, True si le joueur peut se déplacer, False sinon.

Enfin, j'ai entamé la création des fantômes, pour ça j'ai recyclé la plupart du code que j'ai utilisé pour les déplacements du joueurs, mais j'ai créé une liste contenant les déplacements disponibles (haut, bas, gauche, droite). Le fantôme générera un nombre aléatoire entre **0 et 100**, si le nombre sélectionné est en dessous de 50, le fantôme choisira un élément aléatoire de la liste et vérifiera si le déplacement demandé est valide. Si oui, il avancera.

Pour terminer, j'ai mis en place le système de score, donné la possibilité à Pacman de manger les pac gums et les fantômes lors des super pac gums et la possibilité aux fantômes d'avoir différents sprites.

Lorsque Pacman mange un fantôme quand il est en possession d'une super pac gum, le fantôme ne sera plus affecté par celle-ci. C'est-à-dire que les autres fantômes seront en panique, celui-ci sera de nouveau normal.

Création des menus général et de fin

J'ai créé différentes images sur le logiciel **paint.net** afin d'ajouter un peu de contenu, ainsi que des boutons.

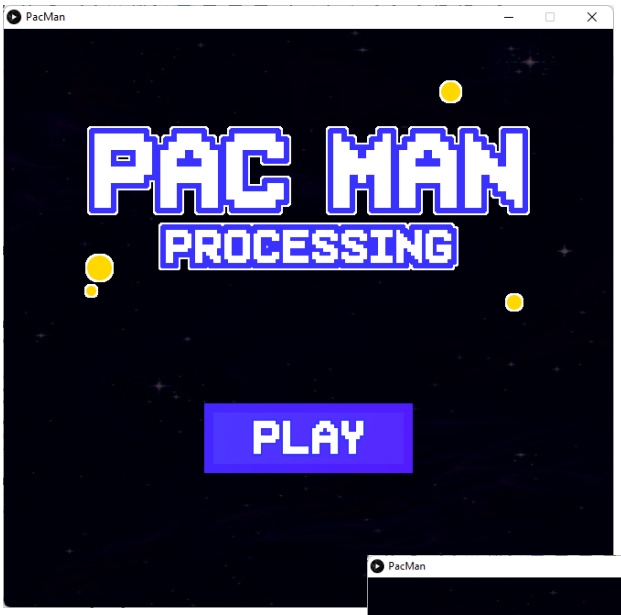
Pour les boutons, j'ai dû vérifier la position de la souris afin de déterminer si le joueur avait posé sa souris dessus pour ensuite détecter son clic.

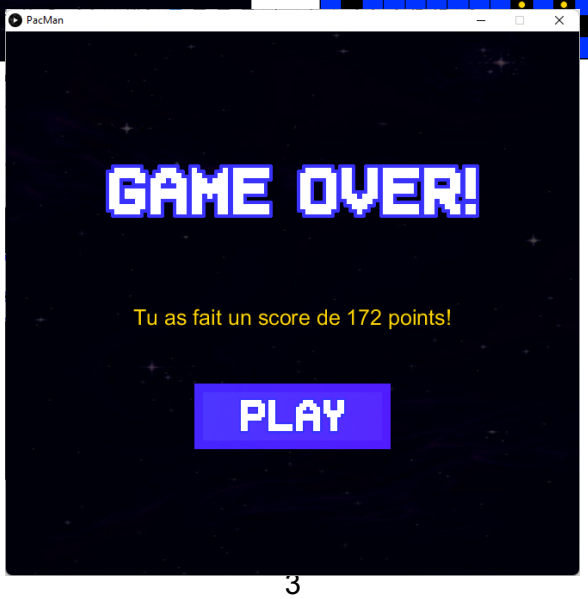
```
# Regarde les coordonnees du bouton jour afin de regarder si il pose sa souris dedans
xBoutonMin, xBoutonMax = (width * 0.5) - self.playButton.width // 2, (width * 0.5) + self.playButton.width // 2
yBoutonMin, yBoutonMax = self.logoImage.height * 1.2, self.logoImage.height * 1.2 + self.playButton.height

if xBoutonMin < mouseX < xBoutonMax and yBoutonMin < mouseY < yBoutonMax: # Le joueur a mit sa souris sur le bouton
    if mousePressed: # Si l'utilisateur clique sur le bouton
        self.status = "jeu" # Change le status en jeu
```

Pour le fond, j'ai utilisé celui d'un vieux projet réalisé en NSI étant le Space Invader et pour pouvoir alterner entre les différents menus, j'ai ajouté une variable **status** qui varie entre *jeu, gameover, menu.*

Résultat du projet "Pac Man"





Arborescence du programme

PacMan (main.py):

- **setup()** Permet d'initialiser processing et ses options. C'est ici qu'est appelée la classe *Jeu*.
- **draw()** Dessine le menu principal, le jeu et l'écran de game over.

Decor.py:

- **Class Decor**:
  - **__init__(self, laby)** Charge le csv du labyrinthe a partir de l'argument **laby** et définit la longueur des cases.
  - **toList(self, csv)** Transforme le csv du labyrinthe en liste de liste et la retourne.
  - **reset(self)** Réinitialiser les pac gums, fonction appelée lors d'un game over.
  - **dessine(self)** Dessine le labyrinthe sur l'interface utilisateur.
  - **afficheScore(self, score)** Affiche le score du joueur passé en argument sur l'interface utilisateur.
  -

Fantome.py:

- **Class Fantome:**
  - **__init__(self, longueurCase, couleur)** Définit les caractéristiques du fantôme, tel ses coordonnées, ses images, sa couleur etc…
  - **reset(self)** Cette fonction reset le fantôme lors du démarrage d'une partie, fonction appelée lors d'un game over ou lorsque pacman mange un fantôme.
  - **dessine(self)** Cette fonction permet de dessiner le fantôme sur la grille
  - **est_au_centre(self)** Cette fonction permet de vérifier si le fantôme est au centre de la case dans laquelle il se situe.
  - **deplacement_valide(self, grille)** Cette fonction va vérifier si le déplacement du fantôme est valide (Si il y a un mur ou non).
  - **avancer(self, vitesse)** Cette fonction permet de faire avancer le fantôme dans une direction avec une vitesse définie.
  - **deplacement(self, grille, pacman)** Cette fonction permet de contrôler les déplacements du fantôme

Jeu.py:

- **Class Jeu:**
  - **__init__(self, fichier)** Définit toutes les caractéristiques du jeu, tel le décor, le joueur, les fantômes, le timer, les images etc…
  - **menuPrincipal(self)** Cette fonction affiche le menu principal.
  - **reinitialise(self)** Cette fonction réinitialise le jeu.
  - **gameOver(self)** Cette fonction affiche le game over
  - **jouer(self)** Cette fonction permet d'exécuter tous les éléments du jeu (déplacements de pacman, fantômes, manger les pac gums, etc…)

Pac_Man.py:

- **Class PacMan:**
  - **__init__(self, longueurCase)** Définit les caractéristiques de pacman, tel ses coordonnées etc…
  - **reset(self)** Cette fonction reset pac man lors du démarrage d'une partie, fonction appelée lors d'un game over.
  - **dessine(self)** Cette fonction permet de dessiner pacman sur la grille
  - **est_au_centre(self)** Cette fonction permet de vérifier si pacman est au centre de la case dans laquelle il se situe.
  - **deplacement_valide(self, grille)** Cette fonction va vérifier si le déplacement de pacman est valide (Si il y a un mur ou non).
  - **verifiePacGums(self, grille)** Cette fonction va vérifier si pacman se trouve sur une pac gum, si oui retourne un tuple (Booléen si il est sur une pac gum, la coordonnée X et Y).
  - **avancer(self, vitesse)** Cette fonction permet de faire avancer pacman dans une direction avec une vitesse définie.
  - **deplacement(self, grille, pacman)** Cette fonction permet de contrôler les déplacements de pacman