LE GOUVERNEMENT
DU GRAND-DUCHÉ DE LUXEMBOURG
Ministère de l'Éducation nationale,
de l'Enfance et de la Jeunesse

EXAMEN DE FIN D'ÉTUDES SECONDAIRES CLASSIQUES

2019

BRANCHE	SECTION(S)	ÉPREUVE ÉCRITE
Informatique	**B**	*Durée de l'épreuve* 180 minutes
		Date de l'épreuve

La grenouille qui traverse la route

Créez le programme `grenouille.py` dans le dossier qui vous a été indiqué !
Ajoutez votre numéro de candidat en tant que commentaire en haut du fichier !

Le programme `grenouille.py` fait appel à la librairie `pygame`. Les seules importations permises sont `randrange` et `randint` du module `random`, `pygame`, `pygame.locals` et `sys`.

I) Initialisation (1+1) 2 p.

1) Effectuez le chargement des bibliothèques nécessaires.

2) La fenêtre d'application mesure 600 x 200 pixels et l'entête de la fenêtre est : `Grenouille - LYCEE - NUMERO`.
La surface de dessin `screen` initialement noire est rafraîchie 25 fois par seconde.

II) La classe `Frog` (1+1+1) 3 p.

La classe `Frog` décrit une grenouille, représentée par un cercle. Les attributs de cette classe sont les coordonnées entières `x` et `y` du centre et la couleur `color`.

1) Le constructeur initialise les attributs sur les valeurs passées aux paramètres. Les valeurs par défaut sont `x=300` et `y=180` et `color=Color('white')`.

2) La méthode `reset_frog` remet les attributs de la grenouille sur les valeurs par défaut et affiche dans la console `'frog reset done'`.

3) La méthode `draw_frog` dessine sur la toile un cercle de couleur `color`, de centre `(x,y)`, de rayon `3` et d'épaisseur `1`.

III) La classe `Car` (2+1+4+2+1) 10 p.

La classe `Car` décrit une voiture représentée par un rectangle creux ne pouvant se déplacer qu'horizontalement. Les attributs de cette classe sont les coordonnées entières `x` et `y` du coin supérieur gauche, la largeur `width` et la hauteur `height` du rectangle et la vitesse horizontale `speed`.

1) Le constructeur initialise les coordonnées `x`, `y`, `width` et `height` sur les valeurs passées aux paramètres. Les valeurs par défaut sont `x=0`, `y=200`, `width=20` et `height=10`. `speed` est un entier aléatoire entre -20 et 20.

2) La méthode `reset_car` remet la coordonnée `x` de la voiture à la valeur par défaut `0` réinitialise la vitesse aléatoirement et affiche dans la console `'car reset done'`.

3) La méthode `drive_car` fait avancer la voiture horizontalement de `speed` pixels. Le déplacement se fait vers la droite ou vers la gauche en fonction du signe de `speed`. Au moment où la voiture a quitté complètement l'écran (à droite ou à gauche), elle doit réapparaître progressivement de l'autre côté de l'écran et continuer son trajet dans le même sens.

4) La méthode booléenne `contains_pixel` prend deux paramètres `x` et `y` et retourne `True` si le pixel de coordonnées `(x,y)` se trouve à l'intérieur du rectangle délimitant la voiture (bords inclus), `False` sinon.

5) La méthode `draw_car` dessine sur la toile le rectangle creux jaune représentant la voiture, avec une épaisseur égale à 1.

IV) La classe `Street` (1+1+4+4+1+1+1+1+6) 20 p.

La classe `Street` gère une liste de voitures de type `Car`. A cet effet elle dispose d'un seul attribut qui est la liste `list_of_cars`.

1) Le constructeur initialise la liste vide `list_of_cars`.

2) La méthode `append_car` obtient comme paramètre un objet de type `Car` et l'ajoute à la liste `list_of_cars`.

3) La méthode `speed_up_cars` prend comme paramètre un nombre entier `amount` strictement positif et modifie la variable `speed` de chaque voiture dans la liste de sorte qu'elle roule plus vite : la vitesse en valeur absolue doit augmenter de `amount`. Le sens de déplacement de la voiture ne doit pas changer, sauf éventuellement si la voiture est au repos. Dans ce cas il faudra choisir un nouveau sens de déplacement au hasard.

4) La méthode `slow_down_cars` prend comme paramètre un nombre entier `amount` strictement positif et modifie l'attribut `speed` de chaque voiture dans la liste de sorte qu'elle freine : la vitesse en valeur absolue doit diminuer de `amount`.
Si la vitesse en valeur absolue de la voiture est inférieure à `amount` la voiture est à immobiliser.

5) La méthode `stop_cars` doit arrêter immédiatement toutes les voitures de la liste.

6) La méthode `drive_cars` fait appel à la méthode `drive_car` des voitures dans la liste afin de les déplacer.

7) La méthode `draw_cars` permet de dessiner sur la toile `pygame` toutes les voitures de la liste en faisant appel à la méthode `draw_car`.

8) La méthode `reset_cars` réinitialise chaque voiture en faisant appel à la méthode `reset_car` de chaque voiture.

9) La méthode booléenne `check_collision` qui prend un paramètre du type `Frog`, vérifie s'il y a une collision entre la grenouille et une des voitures dans la liste. Pour cela il suffit d'analyser si l'un des pixels suivants du cercle représentant la grenouille se trouve à l'intérieur d'une voiture : le pixel le plus haut, le pixel le plus bas, le pixel le plus à droite et le pixel le plus à gauche. En cas de collision la méthode retourne `True`, sinon `False`.

V) Le programme principal 25 p.

1) Préparation (1+2) 3 p.

A. Créez une instance `frog` de la classe `Frog` avec les valeurs par défaut et une instance `street` de la classe `Street`.

B. Ajoutez quatre instances de la classe `Car` à la liste de l'objet `street` à l'aide de la méthode `append_car`. Les voitures ont comme abscisse `x=0` et comme ordonnées `y=40`, `80`, `120` et `160` respectivement. Les dimensions `width` et `height` sont les valeurs par défaut.

2) Boucle principale - Réaction aux évènements (1+3+4+4) 12 p.

A. Lorsque l'utilisateur ferme la fenêtre, le programme se termine correctement.

B. Lorsqu'on enfonce la touche **'SPACE'** le jeu est réinitialisé. L'image est repeinte en noir et on fait un reset de la grenouille et des voitures en appelant les méthodes correspondantes.

C. Les mouvements de la grenouille se font grâce aux flèches de direction du clavier. Si l'une des quatre touches est enfoncée, les coordonnées de la grenouille sont changées de 2 pixels dans la direction correspondante. Le mouvement doit se poursuivre aussi longtemps que la touche reste enfoncée.

D. Lorsqu'on enfonce une des touches **'a'** ou **'s'**, toutes les voitures sont accélérées respectivement freinées de la valeur 1. L'accélération et le freinage doivent se poursuivre aussi longtemps que la touche correspondante reste enfoncée.

3) Boucle principale - Actions répétées à chaque itération (1+1+2+1+5) 10 p.

A. L'écran est repeint en noir, la grenouille est dessinée.

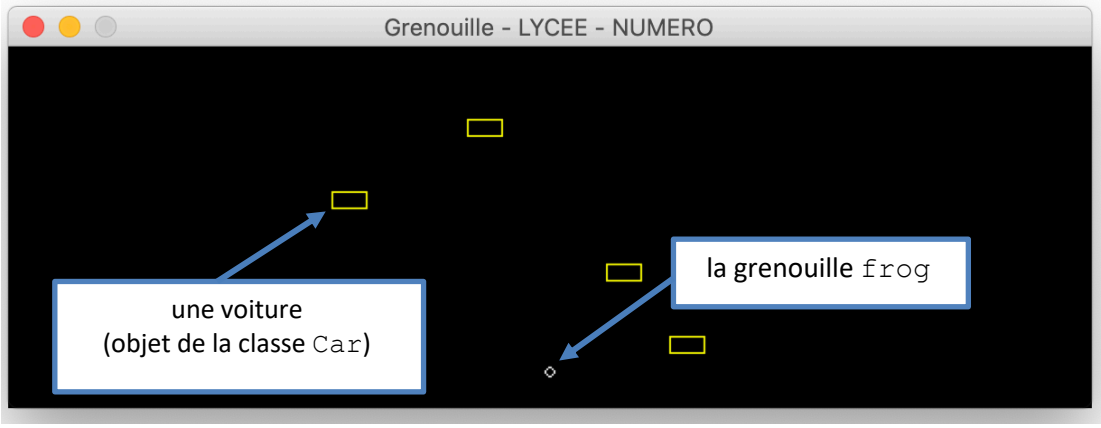
B. Toutes les voitures roulent et sont dessinées.

C. S'il y a eu une collision entre une voiture et la grenouille, toutes les voitures sont arrêtées et la grenouille devient rouge.

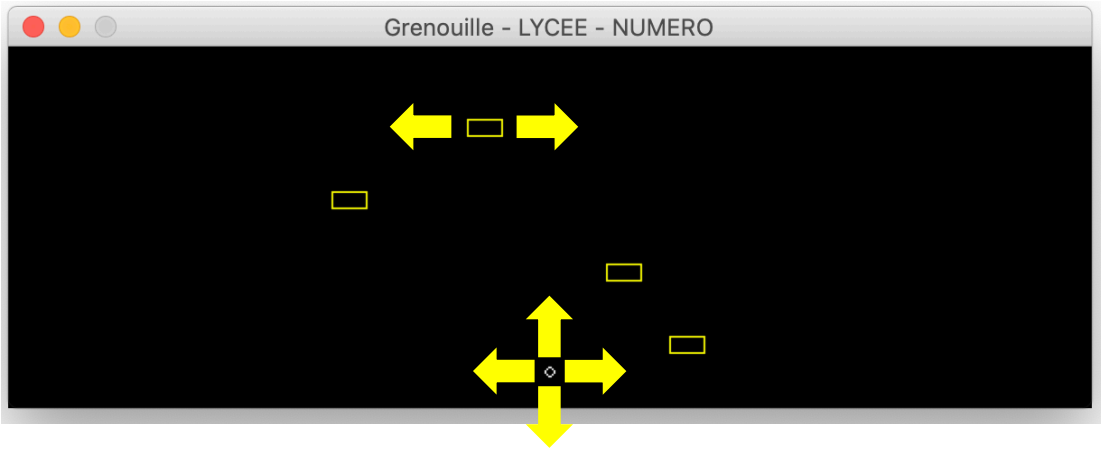
D. Si le point le plus haut de la grenouille dépasse le bord supérieur de la fenêtre, toutes les voitures sont arrêtées et la grenouille devient verte. Elle a traversé la route saine et sauve.

E. Servez-vous d'une variable booléenne `game_stopped` afin d'empêcher le déplacement de la grenouille et le changement de vitesse des voitures après une collision ou lorsque la grenouille a traversé la route.

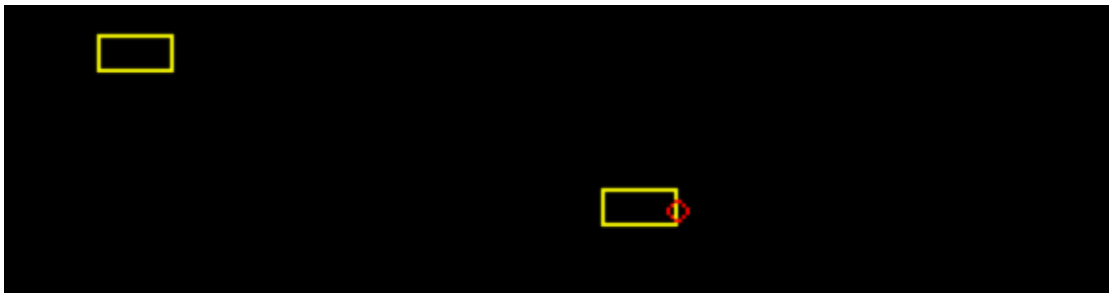
Capture 1 - Situation de jeu peu après le lancement.



Capture 2 – Mouvements possibles des objets (voir flèches jaunes)



Capture 3 - Une collision avec une voiture (agrandissement)



Capture 4 - La grenouille regagne l'autre bord de la route (agrandissement)