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
2021

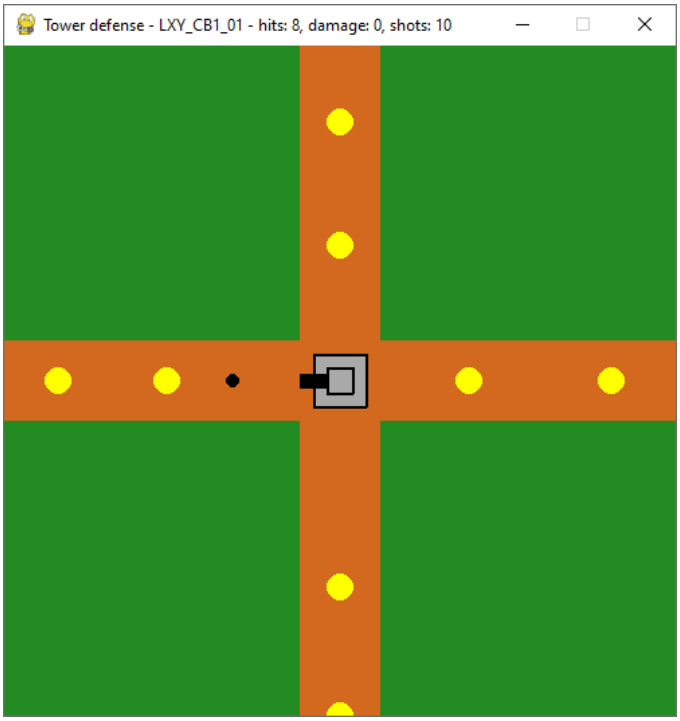
BRANCHE	SECTION(S)	ÉPREUVE ÉCRITE
Informatique	**B**	*Durée de l'épreuve : 180 minutes* *Date de l'épreuve : 01/06/2021*

Créez à l'emplacement défini par votre lycée/professeur un dossier nommé selon votre code d'examen (par exemple : **`LXY_CB1_01`**). Sauvegardez-y régulièrement votre travail. En haut de chaque fichier que vous créerez indiquez votre code d'examen en tant que commentaire. A la fin de l'épreuve imprimez puis signez votre code.

Jeu de style « Tower defense »

Principe du jeu : une tour se trouvant au centre doit se défendre contre une vague d'ennemis (disques jaunes) se déplaçant vers elle. Les flèches du clavier permettent d'orienter le canon de la tour, la touche « espace » permet de tirer des boulets de canon (disques noirs).

Créez le fichier **`tower.py`** dans votre dossier de travail. Ce fichier sera structuré en 3 parties : importations et initialisations, définitions de classes et programme principal.



1. IMPORTATIONS ET INITIALISATIONS [2 pts]

a. Les seules importations permises sont celles de **`pygame`**, **`pygame.locals`**, **`sys`**, des fonctions **`sqrt`** du module **`math`** et **`randint`** du module **`random`**.

b. Définissez les constantes **`SIZE = 500`** et **`CENTER = SIZE // 2`** ainsi que les tuplets **`LEFT = (-1, 0)`**, **`UP = (0, -1)`**, **`RIGHT = (1, 0)`** et **`DOWN = (0, 1)`**. Initialisez pygame. Les dimensions de la surface de dessin carrée sont **`SIZE`**, la fréquence de rafraîchissement par seconde est **`FPS = 30`**.

2. CLASSES [5 + 12 + 12 = 29 pts]

a. La classe **Bullet** représente un boulet de canon. ***(1 + 1 + 3 = 5 pts)***

Bullet
direction : (int,int) x : int y : int radius : int hit : bool
__init__(direction) draw() move()

- Le constructeur initialise l'attribut **direction** sur le paramètre de même nom. **direction** est un tuplet qui indique la direction et le sens du boulet (**LEFT**, **UP**, **RIGHT** ou **DOWN**). Les attributs **x** et **y** représentent les coordonnées du centre du boulet et sont initialisés à celles du centre de la surface de jeu. L'attribut **radius** vaut 5 pixels et l'attribut booléen **hit** est initialisé à **False**. (1 pt)
- La méthode **draw** dessine un disque noir de centre **(x,y)** et de rayon **radius**. (1 pt)
- La méthode **move** déplace le boulet de canon de 10 pixels dans la direction du tir, mais uniquement si le centre du boulet se trouve à l'intérieur de la surface de jeu. Si tel n'est pas le cas, l'attribut **hit** est à modifier afin qu'on sache que le boulet a touché les bords. (3 pts)

b. La classe **Tower** représente une tour avec son canon. ***(2 + 5 + 1 + 2 + 2 = 12 pts)***

Tower
x : int y : int dimension : int damage : int hits : int shots : int color : Color bullets : [Bullet] direction : (int,int)
__init__() draw() increase_damage() repair() shoot()

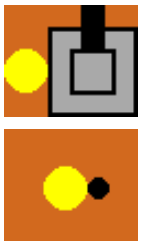
- Le constructeur initialise les attributs **x** et **y** (= coordonnées du centre de la tour) à celles du centre de la surface de jeu. L'attribut **dimension** égale 40 pixels. Les attributs **damage**, **hits** et **shots** sont initialisés à zéro : **damage** représente les dégâts de la tour dus aux attaques des ennemis, **hits** est le nombre d'ennemis touchés et détruits par les tirs du canon, **shots** est le nombre total de boulets tirés par le canon. La couleur de la tour est **darkgrey**. La liste **bullets** est initialement vide. L'attribut **direction** indique la direction du tir (**LEFT**, **UP**, **RIGHT** ou **DOWN**) et doit être initialisé à **LEFT**. (2 pts)
- La méthode **draw** dessine la tour et le canon (voir image ci-contre) : (5 pts)

  a) la base de la tour est un carré gris foncé de taille **dimension** de centre **(x,y)** avec une bordure noire d'épaisseur 2 pixels,
  b) le canon est un segment noir d'origine **(x,y)** de longueur 30 pixels, d'épaisseur 11 pixels et pointant dans la direction **direction**.
  c) le haut de la tour est un carré gris foncé de taille **dimension // 2** de centre **(x,y)** avec une bordure noire d'épaisseur 2 pixels.
- La méthode **increase_damage** augmente de 15 unités la valeur de l'attribut **damage** et met la couleur de la tour sur rouge. (1 pt)
- La méthode **repair** décrémente l'attribut **damage** de 1 si celui-ci est strictement positif. Lorsque **damage** devient 0, la couleur de la tour redevient gris foncé. (2 pts)
- La méthode **shoot** crée un boulet de canon (de type **Bullet**) dont la direction est celle du canon et l'ajoute à la liste **bullets**. L'attribut **shots** est incrémenté de 1. (2 pts)

c. La classe **Enemy** représente un ennemi qui se dirige vers la tour. ***(1 + 1 + 3 + 5 + 2 = 12 pts)***

Enemy
x : float **y : float** **direction : (int,int)** **radius : int** **hit : bool**
__init__(x, y, direction) **draw()** **move(t)** **check_hit(t)** **distance_to(other) : float**

- Le constructeur initialise les attributs **x** et **y** (= coordonnées du centre) et **direction** (**LEFT**, **UP**, **RIGHT** ou **DOWN** suivant la direction de l'ennemi) sur les paramètres de même nom. L'attribut **radius** vaut 10 pixels et l'attribut booléen **hit** est initialisé à **False**. (1 pt)
- La méthode **draw** dessine l'ennemi sous forme d'un disque jaune, de centre **(x,y)** et de rayon **radius**. (1 pt)
- La méthode **move** définit d'abord une variable **speed = 1 + hits/15** où **hits** est l'attribut correspondant de la tour **t** fournie comme paramètre. Ensuite la méthode déplace l'ennemi de **speed** pixels selon sa **direction**. Finalement la méthode vérifie s'il y a une collision en faisant appel à sa méthode **check_hit**. (3 pts)
- La méthode **check_hit** vérifie s'il y a une collision entre l'ennemi et la tour **t** ou un boulet de canon de la tour. Une collision a lieu lorsque la distance entre les deux objets est inférieure ou égale à un certain seuil, à déterminer (voir figures ci-contre, illustrant les instants d'une collision). (5 pts)



a) S'il y a une collision avec la tour, l'attribut **hit** de l'ennemi est mis à jour et la tour fait appel à sa méthode **increase_damage**.

b) S'il y a une collision avec un boulet de canon de la tour, les attributs **hit** de l'ennemi et du boulet sont mis à jour et l'attribut **hits** de la tour est incrémenté de 1.

- La méthode **distance_to** retourne la distance entre l'ennemi actuel et un autre objet (paramètre **other**) : c'est par définition la distance entre les centres des deux objets. (2 pts)

3. PROGRAMME PRINCIPAL [7 + 1 + 4,5 + 3 + 13,5 = 29 pts]

a. Ecrivez le code nécessaire pour créer l'environnement du jeu : ***(7 pts)***

- création d'une liste vide **enemies**,
- création de 4 tuplets qui représentent les portes d'entrée (*angl. : gate*) des ennemis : **g1**, **g2**, **g3** et **g4**. Chaque variable stocke deux informations sous forme de tuplet : la position de la porte et la direction des ennemis :

tuplet	*position (x, y)*	*direction*
g1	**(**bord gauche**,** mi-hauteur**)**	**RIGHT**
g2	**(**mi-largeur**,** bord supérieur**)**	**DOWN**
g3	**(**bord droit**,** mi-hauteur**)**	**LEFT**
g4	**(**mi-largeur**,** bord inférieur**)**	**UP**

par exemple : **g1 = ((0, CENTER), RIGHT)**,

- création d'une liste nommée **gates** composée des tuplets **g1**, **g2**, **g3** et **g4**,
- création d'une série de 25 ennemis par porte d'entrée. Pour chaque série, les ennemis doivent être placés à une distance aléatoire comprise entre 80 et 120 pixels les uns des autres. Les ennemis sont à placer en dehors de la surface de jeu, sur leur axe d'attaque,

le premier se trouvant déjà à une distance aléatoire comprise entre 80 et 120 pixels de sa porte d'entrée. Chaque ennemi créé est ajouté à la liste **`enemies`** ;

- création d'un objet **`tower`** de type **`Tower`**,
- définition d'une variable **`game_status`** (str) initialisée sur « **`running`** ».

b. Une boucle principale permet de gérer les événements, d'actualiser l'état du jeu et d'actualiser l'affichage. Ensuite l'environnement pygame et l'application sont clôturés et quittés correctement. ***(1 pt)***

gestion des événements : ***(4,5 pts)***

- un clic sur la croix de fermeture permet de quitter la boucle principale,
- les touches du clavier réalisent les actions suivantes si le jeu n'est pas terminé :
  a) les flèches de direction du clavier (constantes K_LEFT, K_UP, K_RIGHT et K_DOWN) ajustent la direction du canon de la tour ;
  b) la touche « espace » (constante K_SPACE) permet de tirer un boulet.

actualisation de l'état du jeu : ***(3 pts)***

- si les dégâts de la tour sont trop importants (**`damage`** ≥ 50) **`game_status`** devient « **`lost`** » ;
- s'il n'y a plus d'ennemis et :
  a) la tour en a détruit ≥ 50 : **`game_status`** devient « **`won`** » ;
  b) la tour en a détruit < 50 : **`game_status`** devient « **`lost`** »

actualisation de l'affichage : ***(13,5 pts)***

- si le jeu est en cours, l'arrière-plan est de couleur **`forestgreen`** et l'entête de la fenêtre est actualisée avec le texte

  « Tower defense - *code d'examen* - hits: **`h`**, damage: **`d`**, shots: **`s`** »

  si le joueur a gagné, l'arrière-plan devient noir et l'entête de la fenêtre est actualisée avec le texte

  « You WON: **`h`** of 100 enemies hit, accuracy: **`a`** % »

  si le joueur a perdu, l'arrière-plan devient de couleur **`firebrick`** et l'entête de la fenêtre est actualisée avec le texte

  « You LOST. Try to avoid getting hit or to make more hits. ;-) »

  Dans ces strings, les lettres **`h`**, **`d`** et **`s`** désignent respectivement les attributs **`hits`**, **`damage`** et **`shots`** de la tour. **`a`** est le rapport du nombre d'ennemis touchés au nombre de tirs réalisés, exprimé en pourcentage et affiché avec 2 décimales après la virgule ;
- les routes d'attaque sont deux rectangles de largeur 60 pixels et de couleur **`chocolate`** traversant la surface de jeu au centre, respectivement de gauche à droite et de haut en bas ;
- si le jeu est en cours,
  a) un ennemi détruit est effacé de la liste sinon il est dessiné puis déplacé ;
  b) un boulet de canon ayant touché un ennemi ou quitté la surface de jeu est effacé de la liste sinon il est dessiné puis déplacé ;
- la tour est réparée puis dessinée,
- la surface **`pygame`** est mise à jour et l'horloge avance d'un tic.