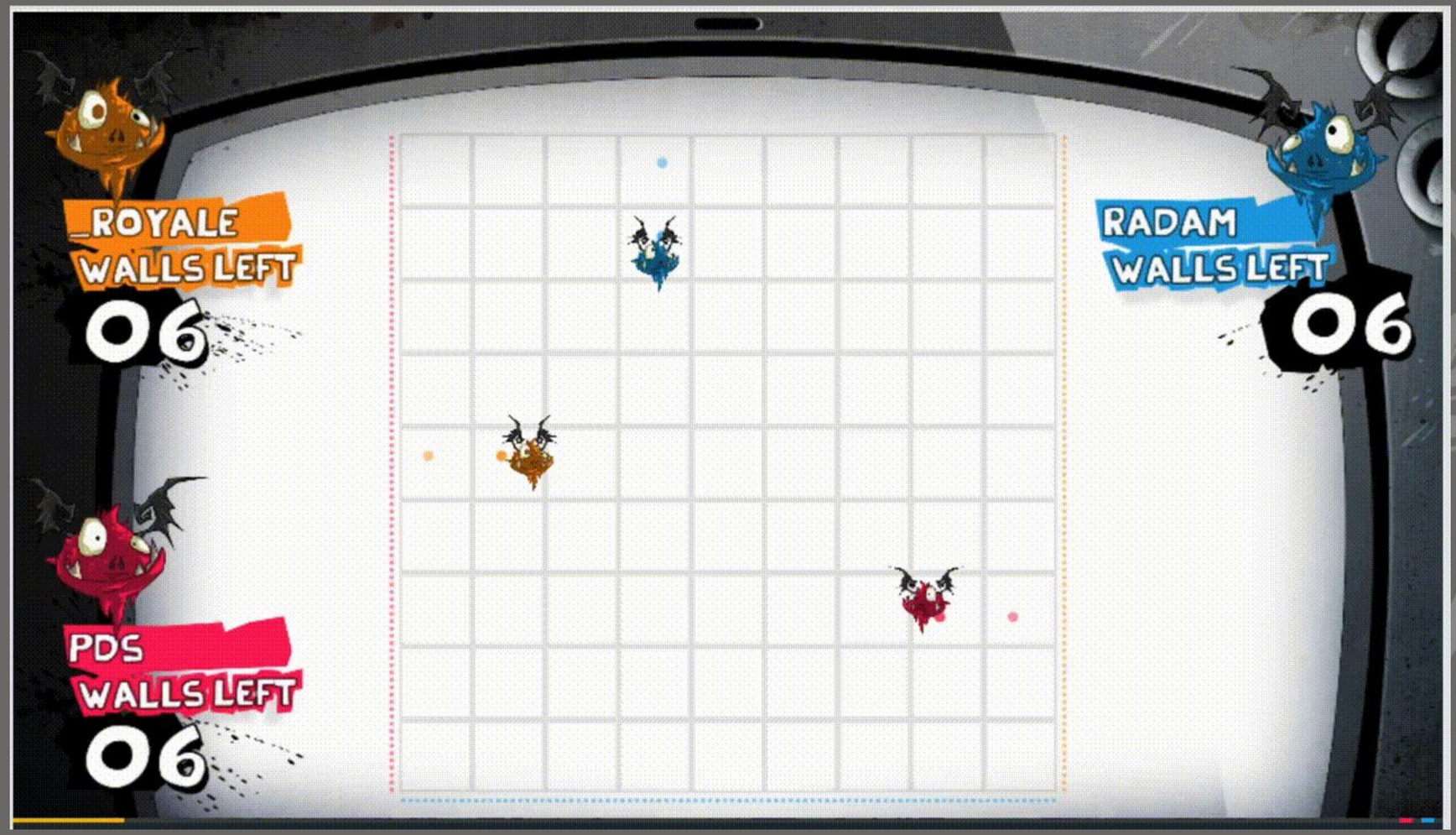
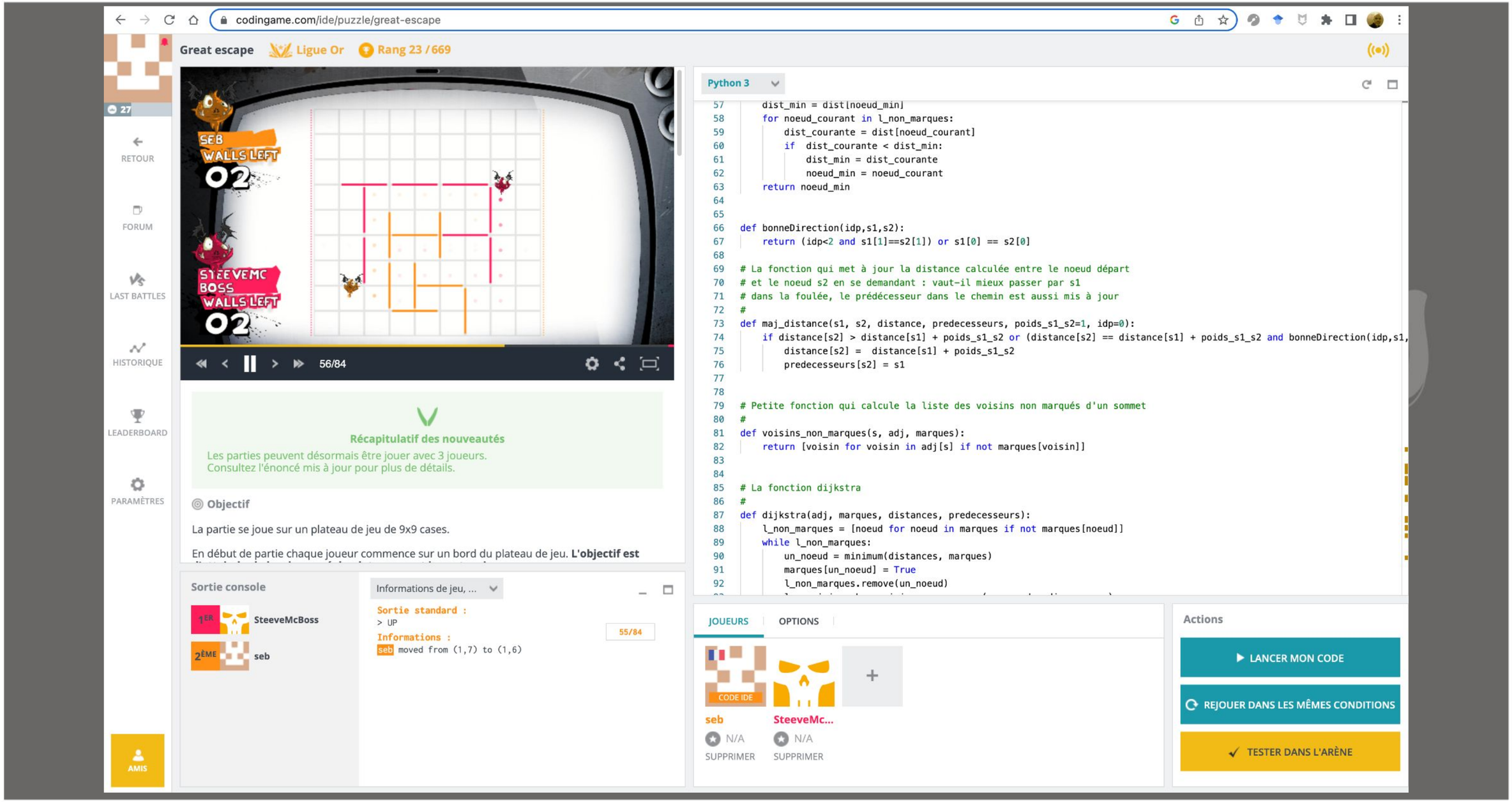
Les Causeries de la Pédagogie

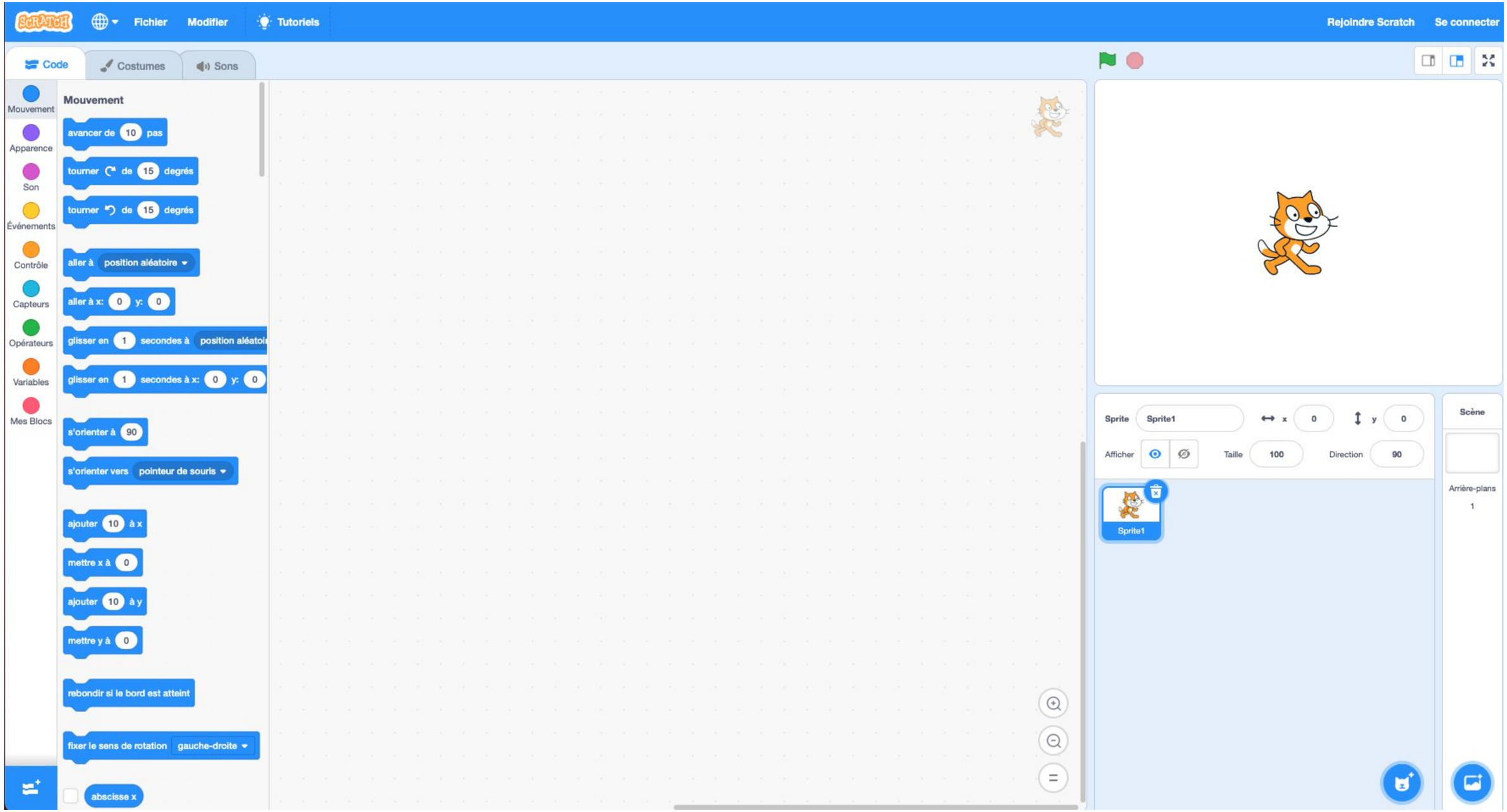
ENSEIGNEMENT ET INNOVATION #6

Production de ressources de qualité et motivation

Avec Dr Sébastien HOARAU, Maître de Conférences en Informatique
Chercheur au Laboratoire d'Informatique et de Mathématiques de l'Université de La Réunion


_ROYALE
WALLS LEFT
06
RADAM
WALLS LEFT
06
PDS
WALLS LEFT
06


codingame.com/ide/puzzle/great-escape
Great escape
Ligue Or
Rang 23 / 669
RETOUR
FORUM
LAST BATTLES
HISTORIQUE
LEADERBOARD
PARAMÈTRES
AMIS
SEB
WALLS LEFT
02
STEEVEMC
BOSS
WALLS LEFT
02
56/84
Récapitulatif des nouveautés
Les parties peuvent désormais être jouer avec 3 joueurs.
Consultez l'énoncé mis à jour pour plus de détails.
Objectif
La partie se joue sur un plateau de jeu de 9x9 cases.
En début de partie chaque joueur commence sur un bord du plateau de jeu. L'objectif est
Sortie console
1ER SteeveMcBoss
2ÈME seb
Informations de jeu, ...
Sortie standard :
> UP
Informations :
seb moved from (1,7) to (1,6)
55/84
Python 3
57 dist_min = dist[noeud_min]
58 for noeud_courant in l_non_marques:
59 dist_courante = dist[noeud_courant]
60 if dist_courante < dist_min:
61 dist_min = dist_courante
62 noeud_min = noeud_courant
63 return noeud_min
64
65
66 def bonneDirection(idp,s1,s2):
67 return (idp<2 and s1[1]==s2[1]) or s1[0] == s2[0]
68
69 # La fonction qui met à jour la distance calculée entre le noeud départ
70 # et le noeud s2 en se demandant : vaut-il mieux passer par s1
71 # dans la foulée, le prédécesseur dans le chemin est aussi mis à jour
72 #
73 def maj_distance(s1, s2, distance, predecesseurs, poids_s1_s2=1, idp=0):
74 if distance[s2] > distance[s1] + poids_s1_s2 or (distance[s2] == distance[s1] + poids_s1_s2 and bonneDirection(idp,s1,
75 distance[s2] = distance[s1] + poids_s1_s2
76 predecesseurs[s2] = s1
77
78
79 # Petite fonction qui calcule la liste des voisins non marqués d'un sommet
80 #
81 def voisins_non_marques(s, adj, marques):
82 return [voisin for voisin in adj[s] if not marques[voisin]]
83
84
85 # La fonction dijkstra
86 #
87 def dijkstra(adj, marques, distances, predecesseurs):
88 l_non_marques = [noeud for noeud in marques if not marques[noeud]]
89 while l_non_marques:
90 un_noeud = minimum(distances, marques)
91 marques[un_noeud] = True
92 l_non_marques.remove(un_noeud)
JOUEURS
OPTIONS
CODE IDE
seb
N/A
SUPPRIMER
SteeveMc...
N/A
SUPPRIMER
Actions
LANCER MON CODE
REJOUER DANS LES MÊMES CONDITIONS
TESTER DANS L'ARÈNE


Fichier
Modifier
Tutoriels
Rejoindre Scratch
Se connecter
Code
Costumes
Sons
Mouvement
Apparence
Son
Événements
Contrôle
Capteurs
Opérateurs
Variables
Mes Blocs
Mouvement
avancer de 10 pas
tourner de 15 degrés
tourner de 15 degrés
aller à position aléatoire
aller à x: 0 y: 0
glisser en 1 secondes à position aléatoi
glisser en 1 secondes à x: 0 y: 0
s'orienter à 90
s'orienter vers pointeur de souris
ajouter 10 à x
mettre x à 0
ajouter 10 à y
mettre y à 0
rebondir si le bord est atteint
fixer le sens de rotation gauche-droite
abscisse x
Sprite
Sprite1
x
0
y
0
Afficher
Taille
100
Direction
90
Sprite1
Scène
Arrière-plans
1

Initiation à la programmation en Python - Edition des programmes par blocs

- Instructions simples
- Instructions composées
- Expressions arithmétiques
- Expressions logiques
- Expressions de chaînes
- Variables entières
- Variables logiques
- Variables chaînes


```
from random import randint


print("bonjour")
```

Exécuter

Entier Booléen Chaîne

OUVRIR ENREGISTRER

iremi974.gitlab.io/python-de-la-fournaise/sautemouton.html

Activité : Jeu de saute-mouton

Joue : Emmène tous les moutons blancs à droite et tous les moutons noirs à gauche ! Les moutons blancs peuvent se déplacer vers la droite vers une case vide ou en sautant par dessus un mouton noir. Les moutons noirs peuvent se déplacer vers la gauche vers une case vide ou en sautant par dessus un mouton blanc.

Programme : Le fonctionnement du bouton **UNDO** n'est pas satisfaisant. Modifie le programme pour qu'il permette de revenir à la configuration précédente.

Enregistre : A la fin de la session, enregistre ton activité.

```
jeu = ["B", "B", "B", "X", "N", "N", "N"]
historique = []
```

```
def saute(i):
    if jeu[i]=="B" and i<6 and jeu[i+1]=="X":
        jeu[i], jeu[i+1] = "X", "B"
    elif jeu[i]=="B" and i<5 and jeu[i+1]=="N" and jeu[i+2]=="X":
        jeu[i], jeu[i+2] = "X", "B"
    elif jeu[i]=="N" and i>0 and jeu[i-1]=="X":
        jeu[i], jeu[i-1] = "X", "N"
    elif jeu[i]=="N" and i>1 and jeu[i-1]=="B" and jeu[i-2]=="X":
        jeu[i], jeu[i-2] = "X", "N"
def undo():
    global jeu
    jeu = ["B", "B", "B", "X", "N", "N", "N"]
```

RESET UNDO

Aide pour gérer l'historique