SUPINFO Academic Dept.

Algorithmique et programmation en Python

Second Mini - Projet

SOS Game

Version 1.0
Last update: 13/12/2018
Use: Students/Staff
Author: Laurent GODEFROY

SOMMAIRE

1 PREAMBULE

Cet examen est à réaliser par groupes de deux étudiants. Dans l'unique cas où le nombre d'étudiants de la promotion est impair, un et un seul groupe de trois est autorisé.

Toute forme de plagiat ou utilisation de codes disponibles sur internet ou tout autre support, même de manière partielle, est strictement interdite et se verra sanctionnée d'un 0, d'une mention « cheater », et le cas échéant d'un conseil de discipline.

Ce mini-projet donnera lieu à des soutenances. Vos horaires de passages vous seront communiqués par votre campus.

Les soutenances sont également par groupes de deux. Elles dureront **20 minutes** pendant lesquelles vous montrerez à votre examinateur le bon fonctionnement de votre programme en en faisant la démonstration. Si vous n'avez pas implémenté le projet dans sa totalité, vous exposerez les parties fonctionnelles.

Pour appuyer votre présentation, vous devrez préparer un fichier de type Powerpoint, dans lesquels vous expliquerez les points du code que vous jugez les plus importants et significatifs. Il n'est pas nécessaire d'envoyer ce fichier à votre examinateur, ce dernier le découvrira le jour de la soutenance. Une communication précisant tout cela vous sera envoyé début janvier.

Un barème **indicatif** vous est donné dans la dernière partie de ce sujet.

2 GENERALITES SUR CE PROJET

Remarque importante : aucun code n'est demandé dans cette partie qui n'est qu'explicative.

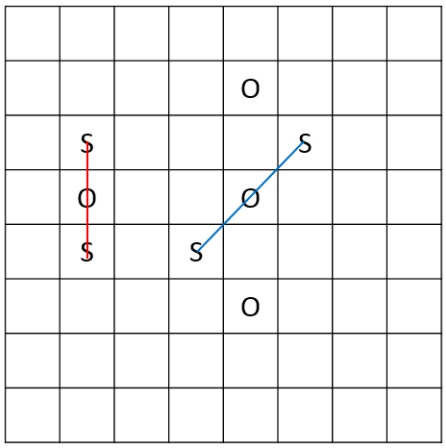
Le but de ce mini-projet est d'écrire en langage Python un programme permettant de jouer au jeu de réflexion et de logique « SOS game ». Ce jeu de papier et crayon est aussi disponible ici sur Google Play (développé par Gasos Technology).

Deux joueurs s'affrontent sur un plateau carré de dimension n, *i.e.* comportant n lignes et n colonnes. Initialement le plateau est vide.

A tour de rôle chacun des joueurs va inscrire sur une case vide soit la lettre « S », soit la lettre « O ». Les joueurs ont totale liberté pour faire ce choix. Le but est de réaliser des alignements verticaux,

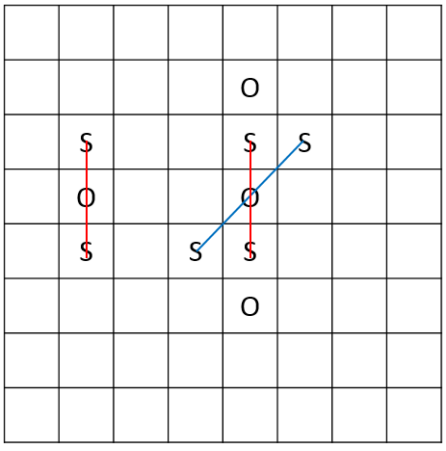
horizontaux ou diagonaux de la séquence de lettres « SOS ». Pour ce faire, un joueur pourra utiliser indistinctement les lettres posées par lui-même ou par son adversaire.

On gardera une trace de qui a réalisé chacun des alignements, en utilisant pas exemple des couleurs, comme sur la figure ci-dessous :

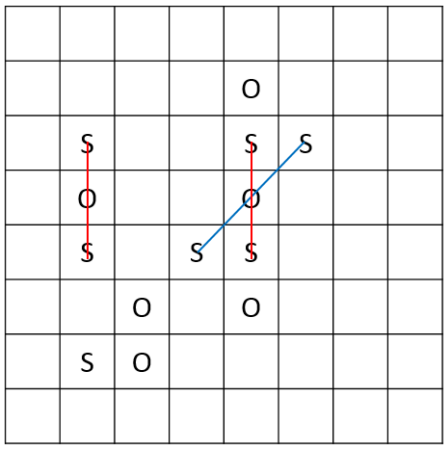


Si après la pose d'une lettre un joueur réalise un tel alignement, son score est incrémenté de 1 et il doit rejouer. Sinon son score reste inchangé et c'est à son adversaire de jouer.

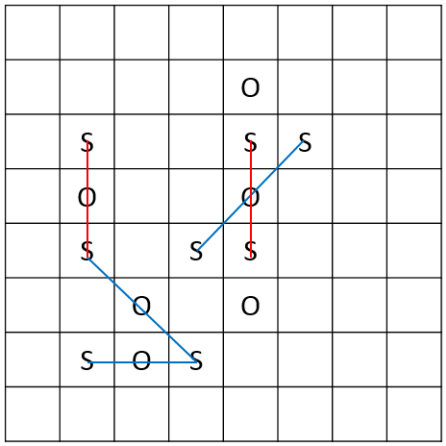
A noter qu'une même lettre peut servir à réaliser plusieurs alignements, comme le « O » situé sur la quatrième ligne et cinquième colonne de l'exemple suivant :



La pose d'une lettre peut provoquer plusieurs alignements simultanément. Si par exemple on est dans cette situation :



Si le joueur en bleu pose un « S » sur la septième ligne et quatrième colonne, il réalise deux alignements :



Son score est alors évidemment augmenté de 2.

Quand le plateau est rempli, est déclaré vainqueur le joueur ayant réalisé le plus d'alignements. Un match nul est bien sûr possible.

Vous aurez peut-être besoin de plusieurs lectures du sujet pour avoir une bonne vue d'ensemble du projet. Prenez donc le temps nécessaire à une bonne compréhension avant de commencer les codes demandés dans les parties suivantes.

Dans la partie 3 on implémentera les algorithmes nécessaires au déroulement du jeu.

Dans la partie 4 on développera l'interface graphique.

Enfin, dans la partie 5 on proposera quelques bonus et extensions possibles.

3 DES ALGORITHMES POUR JOUER A CE JEU

Il vous est fortement recommandé de lire l'intégralité de cette partie avant de commencer à coder.

3.1 STRUCTURES DE DONNEES

Le plateau sera modélisé par une liste à deux dimensions d'entiers. Chacun des éléments de cette liste correspondra à une case du plateau. Une case vide sera représentée par 0, un « S » par 1 et un « O » par 2.

Les joueurs seront représentés par un entier valant 0 ou 1.

Les scores de joueurs seront mémorisés dans une liste de deux entiers.

3.2 ALGORITHMES

Remarques importantes :

- On pourra éventuellement implémenter des sous-programmes en plus de ceux demandés.
- Plutôt que de recalculer régulièrement la dimension de la liste modélisant le plateau, on préfèrera la passer en paramètre de nos sous-programmes.

Notations des paramètres des sous-programmes que l'on va implémenter :

- « board » : liste à deux dimensions d'entiers égaux à 0, 1 ou 2 représentant le plateau de jeu.
- « n » : entier strictement positif égal au nombre de lignes et colonnes de « board ».
- « player » : entier représentant le joueur dont c'est le tour.
- « scores » : liste de deux entiers représentant les scores des deux joueurs.
- « i » : entier quelconque.
- « j » : entier quelconque.
- « lines » : liste à deux dimensions dont les éléments sont des listes de deux tuples de deux entiers. Ces listes de deux tuples correspondent aux coordonnées de deux points, les extrémités d'une ligne lors d'un alignement.
- « l » : entier valant 1 ou 2, représentant un « S » ou un « O ».

Dans un fichier que l'on nommera « sosAlgorithms.py », implémenter les sous-programmes suivants :

- Une fonction « newBoard(n) » qui retourne une liste à deux dimensions représentant l'état initial d'un plateau de jeu à **n** lignes et **n** colonnes.
- Une fonction « possibleSquare(board,n,i,j) » qui retourne **True** si **i** et **j** sont les coordonnées d'une case où un joueur peut poser une lettre, et **False** sinon.
- Une procédure « updateScoreS(board,n,i,j,scores,player,lines) » qui suppose que le joueur **player** ait posé la lettre « S » sur la case de coordonnées **i** et **j**. Elle recherche alors les éventuels alignements de « SOS » que cela a pu engendrer, et met à jour le score du joueur **player** et la liste **lines**.
- Une procédure « updateScoreO(board,n,i,j,scores,player,lines) » qui suppose que le joueur **player** ait posé la lettre « O » sur la case de coordonnées **i** et **j**. Elle recherche alors les éventuels alignements de « SOS » que cela a pu engendrer, et met à jour le score du joueur **player** et la liste **lines**.
- Une procédure « update(board,n,i,j,l,scores,player,lines) » qui commence par mettre à jour le plateau de jeu en affectant la valeur **l** à la case de coordonnées **i** et **j**. Selon les cas elle appelle ensuite l'une des deux procédures précédentes. Lors de l'appel de cette procédure, la liste **lines** est vide.
- Une fonction « winner(scores) » qui retourne une chaîne de caractère indiquant le résultat de la partie.

4 UNE INTERFACE GRAPHIQUE

Il vous est fortement recommandé de lire l'intégralité de cette partie avant de commencer à coder. On devra nécessairement utiliser la librairie graphique Pygame. L'usage d'une autre librairie ne sera pas pris en compte.

Remarque 1 : dans cette partie on utilisera bien sûr les mêmes structures de données que dans la partie précédente. Les paramètres auront également le même sens, avec en plus « mySurface » qui désignera la surface sur laquelle on dessine.

Remarque 2 : vous êtes libre de concevoir le graphisme de votre jeu comme bon vous semble tant que les fonctionnalités sont présentes. Les captures d'écran de cette partie ne sont là que pour vous donner une idée.

Remarque 3 : tous les événements seront déclenchés par la souris et non par le clavier.

4.1 LES PROCEDURES D'AFFICHAGE

Le but de cette partie est d'implémenter les procédures permettant d'afficher le plateau de jeu et les scores des joueurs.

Initialement le plateau vide sera affiché comme ceci :

Chaque case vide est subdivisée en deux, le joueur dont c'est le tour cliquera sur la partie gauche pour poser un « S » sur cette case et sur la partie droite pour y poser un « O ».
La flèche à droite des scores indique le joueur qui doit jouer.

Après quelques tours de jeu l'affichage devra ressembler à celui-ci :

A noter qu'après la pose d'une lettre on ne redessine pas l'intégralité du plateau, on écrit juste cette lettre et on dessine les éventuels alignements. En utilisant dans les deux cas la couleur du joueur.

Enfin, une fois la partie terminée, son résultat sera également affiché :

Dans un fichier que l'on nommera « sosGUI.py », implémenter les sous-programmes suivants :

- Une procédure « drawBoard(mySurface,n) » qui dessine le plateau initial.
- Une procédure « displayScore(mySurface,n,scores) » qui affiche le score des deux joueurs.
- Une procédure « displayPlayer(mySurface,n,player) » qui indique au joueur **player** que c'est son tour.
- Une procédure « drawCell(mySurface,board,i,j,player) » qui dessine le contenu de la case de coordonnées **i** et **j** de la couleur du joueur **player**.
- Une procédure « drawLines(mySurface,lines,player) » qui trace les lignes correspondant aux éventuels alignements dont les informations sont contenues dans la liste **lines**.
- Une procédure « displayWinner(mySurface,n,scores) » qui affiche le résultat de la partie.

4.2 LA PROCEDURE DE JEU

Dans le fichier précédent, implémenter les sous-programmes suivants :

- Une fonction « selectSquare(mySurface,board,n) » qui fait choisir une case et une lettre, et qui retourne ces données sous la forme d'un tuple **(i,j,l)**.
- Une procédure « gamePlay(mySurface,board,n,scores) » qui gère une partie complète.

4.3 LA PROCEDURE PRINCIPALE

Dans le fichier précédent, implémenter le sous-programme suivant :

- Une procédure « SOS(n) » qui créera une fenêtre graphique, initialisera les structures de données, et gèrera une partie complète.

5 DES BONUS

Vous êtes libre d'implémenter zéro, un ou plusieurs des bonus suivants.

5.1 SAUVEGARDER UNE PARTIE

Rajouter une option permettant de sauvegarder l'état d'une partie dans un fichier texte.
Implémenter également une possibilité de reprendre le cours d'une partie mémorisée dans un fichier texte.

5.2 MULTI-JOUEURS

Implémenter un mode où plus de deux joueurs peuvent disputer une partie.

5.3 IA RANDOM

Implémenter un mode de jeu solo contre l'ordinateur, celui-ci jouant de façon aléatoire.

5.4 IA LEGEREMENT INTELLIGENTE

Implémenter un mode de jeu solo contre l'ordinateur, celui-ci adoptant cette intelligence artificielle :

- Si un « SOS » est réalisable, le faire.

- Sinon, dans la mesure du possible, poser un « S » ou un « O » afin qu'au coup suivant un « SOS » ne soit pas réalisable.

6 BAREME INDICATIF

Ce barème peut-être amener à évoluer, il n'est donc qu'i**ndicatif**.

- Partie 3 : 15 points
- Partie 4 : 25 points
- Bonus : 10 points

Ce qui fait un total de 40 points, votre soutenance étant évaluée sur 20 points. La note totale sur 60 points sera alors ramenée sur 20 points par proportionnalité.