Cette tâche a été verrouillée le 19 Janv à 23:59.

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.

Votre rendu s'effectuera sous la forme d'une archive au format .zip contenant vos codes sources.

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.

Ce mini-projet donnera lieu à des soutenances, voir la page dédiée pour plus de détails. Vos horaires de passages vous seront communiqués par votre campus.

Le but de ce mini-projet est de programmer en Python le jeu de réflexion et de logique "Just Get Ten". La première version de ce jeu est due à la compagnie Veewo (Connexions vers un site externe.). Par la suite, de nombreuses applications mobiles reprenant ce concept ont été implémentées, vous allez maintenant en développer une version "bureau".

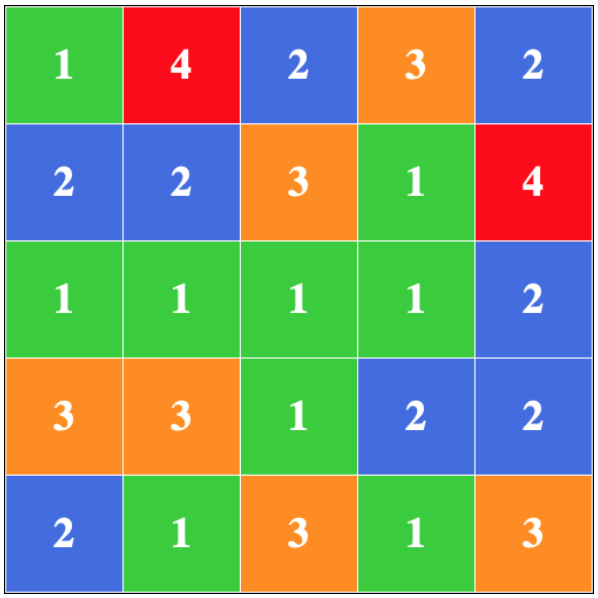
1 - Les règles du jeu

Ce jeu est individuel et se déroule sur une grille carrée de ***n*** lignes et ***n*** colonnes dont les valeurs initiales sont des nombres compris entre **1** et **4** tirés aléatoirement.

Le nombre **1** a une probabilité plus grande d'apparition que le nombre **2**, qui lui-même en a une plus grande que le nombre **3**, etc.

On pourra utiliser les probabilités suivantes : **0,4** pour le **1**, **0,3** pour le **2**, **0,25** pour le **3** et **0,05** pour le **4**.

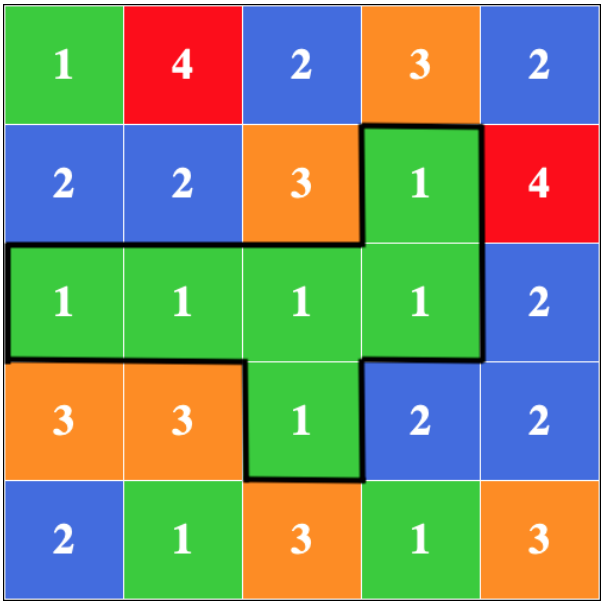
Voici un exemple de configuration initiale pour une grille de 5 lignes et 5 colonnes :



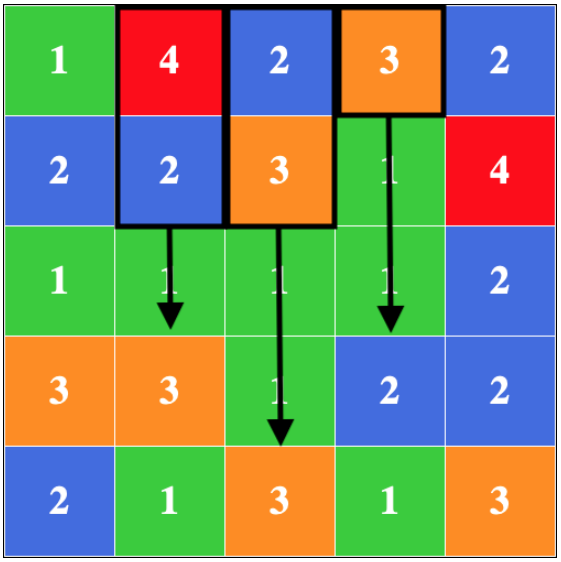
Le but est d'obtenir la valeur **10** en fusionnant successivement des cellules adjacentes de même valeur. Lorsque l'on fusionne un tel ensemble de cellules on obtient une cellule dont la valeur est incrémentée de **1**. Les autres cellules de l'ensemble disparaissent, les cellules non impactées au dessus de l'ensemble "tombent" par gravité, et les colonnes non pleines sont remplies aléatoirement avec les mêmes probabilités que lors de la création de la grille.

A noter que le terme "adjacent" n'inclut pas les cellules se touchant juste par le sommet, en diagonale.

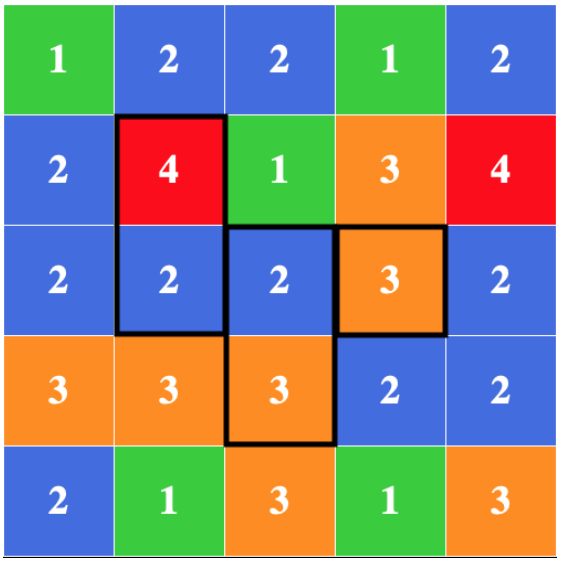
Si l'on reprend l'exemple précédent et que l'on clique sur la cellule située sur la 3ème ligne et 1ère colonne, les six cellules adjacentes constituées de **1** (entourées ici de noir pour mieux les visualiser) fusionnent en un **2** situé donc sur cette 3ème ligne et 1ère colonne :



Les autres cellules du groupe disparaissent, les cellules au-dessus de ce groupe tombent et les colonnes incomplètes sont remplies aléatoirement. Ainsi dans la 2ème colonne, le 2 et le 4 chutent d'une cellule, dans la 3ème colonne, le 3 et le 2 chutent de deux cellules et dans la 4ème colonne, le 3 chute de deux cellules (on les a ici entourés de noir pour mieux les visualiser) :



On obtient alors ceci :

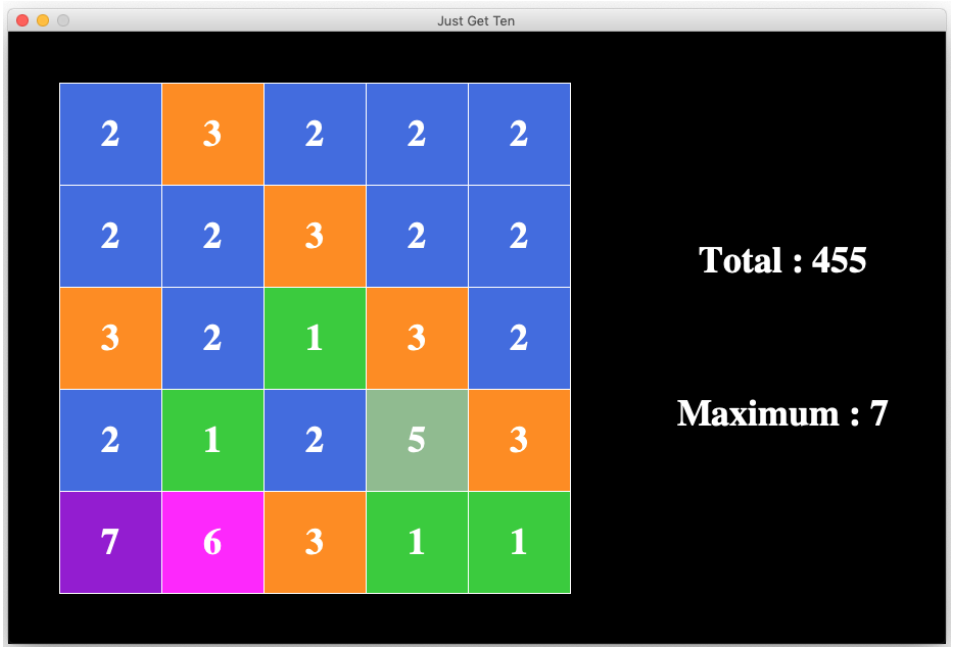


De proche en proche le but est donc d'atteindre 10. Si l'on ne peut plus jouer avant cela, *i.e.* il n'existe plus de cases adjacentes de même valeur, la partie est perdue.

Si l'on atteint 10 la partie continue bien sûr avec l'objectif d'obtenir le maximum possible.

On mémorisera également la somme des cellules fusionnées, la fusion de l'exemple précédent rapportant ainsi 6 points.

L'interface ressemblera donc à :



Remarque (évidente) : les contours noirs de certaines captures ci-dessus n'étaient là que pour expliciter les règles et ne sont donc pas à implémenter.

2 - Implémentation de ce jeu en Python

L'usage de la librairie graphique Tkinter est **obligatoire**, tout autre choix ne sera pas pris en compte.

Une approche orientée objet est **obligatoire** et le principe d'encapsulation devra être scrupuleusement respecté.

Vous implémenterez une ou plusieurs classes selon votre choix.

Le design de l'application est libre, mais votre programme devra au moins comporter les fonctionnalités suivantes :

- Choix d'une taille de grille avec au minimum cinq possibilités (4x4, 5x5, 6x6, 7x7 et 8x8).
- Initialisation aléatoire de la grille selon les règles.
- Affichage de la grille avec une couleur différente par numéro.

- Sélection à la souris d'une case conformément aux règles.
- Fusion des groupes de cellules adjacentes.
- Gestion de la gravité et mise à jour de la grille après fusion.
- Calcul des scores (maximum et total).
- Gestion des tours de jeu et de la fin de partie (affichage du résultat, proposition d'une nouvelle partie, etc.).

On prendra soin de découpler au maximum les méthodes algorithmiques, *i.e.* celles qui interagissent avec la structure de données modélisant la grille, des méthodes graphiques qui elles s'occupent de l'affichage et de la gestion des événements.

3 - Bonus

Les bonus suivants sont facultatifs et ne rentrent donc pas dans le barème de base. Ils apporteront des points supplémentaires et la satisfaction personnelle du devoir accompli.

- Mémoriser dans un fichier et afficher dans l'interface les meilleurs scores.
- Créer des niveaux de difficultés différentes en proposant le choix entre plusieurs répartitions de probabilités.
- Possibilité de mettre en pause une partie et de la reprendre plus tard.
- Limiter le nombre de fusions autorisé pour atteindre 10.
- Limiter le temps pour atteindre 10.
- Limiter le temps de chaque coup.
- Proposer une gestion différente de la gravité, par exemple alternativement dans les quatre directions cardinales.

Durée : 20 minutes de présentation sans interruption puis 10 minutes de questions/réponses.

Il s'agit d'une **soutenance technique** (et non commerciale) qui devra également comporter une **démonstration** du projet.

Vous exposerez votre travail à l'aide d'un document powerpoint (ou équivalent). Il n'est pas nécessaire d'envoyer ce support à votre examinateur en amont.

Il devra comporter les éléments suivants :

- Titre (1 slide)
- Sommaire / plan de la présentation (1 slide)
- Contextualisation / résumé de l'esprit du projet (2 ou 3 slides)
- Présentation et explication des points les plus importants du code (quelques slides). Il convient d'être pertinent dans vos choix.
- Bilan des difficultés rencontrées (1 ou 2 slides)

- Extensions possibles que ce soit du code en lui même, ou de la thématique en général (1 ou 2 slides)

Barème :

- Aisance oratoire : 2 points
- Qualité du powerpoint de présentation : 2 points
- Introduction et résumé du projet : 2 points
- Qualité de l'argumentation : 4 points (à propos du code, mais aussi des difficultés rencontrées et des extensions possibles)
- Choix pertinent des points de codes expliqués : 2 points
- Démonstration du projet : 3 points
- Questions/réponses : 5 points