PROJET MASTERMIND

Le jeu du Mastermind a été inventé en 1970 par Mordecai Meirowitz. Deux joueurs s'y affrontent en tentant tour à tour de deviner le code secret de son adversaire en un minimum de tentatives.

Le jeu est composé d'un plateau (pouvant contenir la combinaison secrète, dix combinaisons candidates, et leur évaluation), de pièces de combinaison (de couleur rouge, verte, bleue, jaune, blanche ou noire), et de pions d'évaluation (de couleur blanche ou noire).

Le défenseur commence par choisir une combinaison (une séquence ordonnée d'exactement quatre pièces) qui sera occultée par la suite.

A chaque tour, L'attaquant formule une combinaison candidate. Le défenseur évalue ensuite la proposition : il place un pion noir pour chaque pièce bien placée (une pièce candidate de la même couleur que la pièce secrète correspondante) et un pion blanc pour chaque pièce mal placée (une pièce candidate qui est de la même couleur qu'une pièce secrète n'ayant pas déjà rapporté un pion).

Le défenseur gagne si la combinaison secrète n'est pas proposée parmi les dix tentatives permises. L'attaquant gagne s'il trouve la combinaison secrète en moins de dix coups, son dégré de réussite étant indiqué par le nombre de coups employés (les meilleurs joueurs n'ont jamais besoin de plus de six coups).

a. Objectif

Réaliser une application qui permet à une personne de jouer contre la machine, soit en tant qu'attaquant, soit en tant que défenseur.

Lorsque le jeu démarre, il affiche un menu qui propose au joueur d'être soit attaquant, soit défenseur (avec une option pour quitter la partie). Si le joueur choisit la défense, un nouvel écran lui permet de sélectionner (à la souris) sa combinaison secrète. Lorsque sa sélection a été validée, le programme formule une proposition (sans se baser sur la combinaison secrète, bien sûr) et affiche en même temps les pions qui évaluent sa proposition. Le joueur peut ensuite cliquer pour passer au coup suivant, jusqu'à ce que le vainqueur soit décidé.

Si le joueur choisit l'attaque, le programme choisit aléatoirement une combinaison secrète, puis affiche l'écran de jeu. Le joueur peut alors construire (toujours à la souris) sa proposition de combinaison. Lorsqu'il valide son choix, le programme affiche les pions correspondants et (s'il n'a pas gagné ou perdu) permet au joueur de proposer une nouvelle combinaison.

À l'issue d'une partie, la combinaison secrète est révélée, et le gagnant (joueur ou machine) est annoncé ainsi que le nombre de coups employés. Durant cette phase, les combinaisons proposées par l'attaquant et les évaluations doivent rester visibles. On pourra améliorer le programme en proposant une solution pour ramener le joueur au menu.

b. Travail à faire

Les instructions qui suivent vont vous guider pas à pas dans la réalisation du programme demandé. Il s'agit d'un travail **individuel**, mais vous pouvez échanger des idées avec vos camarades, mais pas de code cela ne serait pas productif ...

Q1. ***Modèle.*** *Aucun code ne doit être écrit pour l'instant ! En premier lieu, comme vous le feriez pour une base de données, faites la liste des entités décrites dans le texte précédent. Imaginez pour chaque information quel type du langage est le plus approprié. Puis décidez, si nécessaire, comment regrouper ces informations : sous forme de tableau, de fonctions, ou autre.*

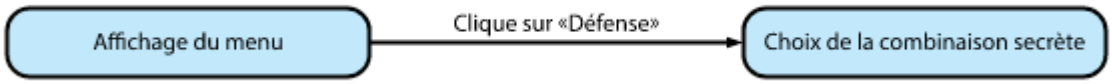
Q2. ***Vue.*** *Nous allons ensuite nous intéresser à l'aspect graphique du projet. Dessinez pour chaque écran souhaité un croquis qui représente l'organisation de l'écran en identifiant les différents composants. Pour chaque composant de chaque écran, si nécessaire, faites un croquis de chaque version possible. Indiquez également lesquelles des entités étudiées dans la première partie influent sur l'apparence de chaque composant.*

Q3. ***Contrôleur.*** *Il nous faut à présent concevoir le comportement du programme. Nous pouvons décomposer ce comportement en* états *et en* transitions*. Un état est une situation particulière durant l'exécution. Une transition est la façon dont on passe d'un état à un autre.*

Pour décrire un état, il faut simplement lui donner un nom suggestif (par exemple, «Affichage du menu»). Pour décrire une transition, il faut répondre à plusieurs questions. Dans quel état part-on ? Qu'est-ce qui provoque la transition (dans ce projet, certainement une action du joueur) ? Quel effet a cet évènement sur les données étudiées dans la première partie ? Quel effet sur l'affichage décrit dans la deuxième partie ? Et enfin, dans quel état arrive-t-on après cela ?

Par exemple, lorsque l'on est dans l'état «Affichage du menu» (état de départ), si le joueur clique sur l'option «Défense» (déclencheur), alors on crée une combinaison secrète vide (effet sur le modèle) et on affiche l'écran de choix de combinaison secrète (effet sur la vue). On est à présent dans l'état «Choix de la combinaison secrète» (état d'arrivée).

Les états ne se succèdent pas forcément de façon linéaire comme les chapitres d'un livre. Parfois des retours arrières sont possibles ; souvent plusieurs chemins divergent. Pour s'y retrouver, on utilise une sorte de storyboard nommé Diagramme états-transitions.



Q4. ***Développement.*** *Il est temps de coder le programme demandé, en vous basant sur le travail de conception des parties précédentes. Chaque entité du modèle doit correspondre à une définition de type. Chaque version de chaque composant de la vue doit correspondre à une fonction d'affichage. Chaque transition doit correspondre à une fonction de traitement.*