Dorsemaine Joris, El-Wafa Hamidi, Tristan Verdet G7

Cahier des charges du mini projet
- bataille navale -

Choix du projet:

Nous avons choisi ce projet car nous avons bien aimé le système de liste 2d que nous avons découvert lors du TD sur le jeu de la vie, avec en plus une interaction possible avec un autre joueur. De plus ce jeu nous semble assez facile à diviser en différentes fonctions (une fonction placer les bateaux en début de partie, une fonction pour tirer, une fonction pour analyser si le tir a touché ou non….). Aussi, ce qui a motivé notre choix pour ce projet, c'est que nous apprécions particulièrement ce jeu qui nous replonge en enfance. On choisit de travailler avec une grille 10x10 représentée par une liste 2 dimensions de taille 10. Chaque liste représente une ligne de la grille. On a besoin de 2 grilles par joueur, une grille qui contient les bateaux du joueur et une dans laquelle on mémorise les cases dans lesquelles on a déjà tiré et si nos tirs précédents ont touchés des bateaux adverses. Nous prévoyons de nous répartir les tâches comme suit:

Planning:

Séance 1: El-wafa définitions des docstrings des fonctions dont on aura besoin
Joris rédaction de l'algorithme en langage courant
Tristan rédaction du cahier des charges
Discussion sur l'organisation du travail et des choix sur la mise en place des règles de la bataille navale.

Séance 2:
Implémentation des différentes fonctions pour le jeu en joueur contre joueur
El wafa : placement des bateaux à l'initialisation
Tristan : verif victoire, affichage grille de jeu , verif_coordonnées + avec El-wafa placement bateaux
Joris : toute la partie tir / vérifier si le tir est réussi …

Séance 3: Implémentation des différentes fonctions pour le jeu en joueur contre ordinateur

Séance 4: Finitions de la boucle principale

Algorithme :

- Initialisation :

Choix nombre de joueurs : 1 (contre l'ordinateur) ou 2 joueurs
Génération de grilles vides :
2 *grilles de position* remplies correspondant à 2 listes 2 dimensions de taille 10 remplies par des 0/False.
1/True = présence de bateau 0/False = absence de bateau

2 *grilles de jeu* correspondant à 2 listes 2 dimensions de taille 10 remplies par des "-"
X = bateau touché O = Tir dans l'eau - = Aucun tir effectué dans cette cellule

Génération des bâteaux :

Chaque bâteau est associé à une variable correspondant à son niveau de vie :

(Le nombre et les tailles de bateau sont choisis arbitrairement et peuvent être changés facilement au niveau de la taille ou sur les éléments de la liste bateaux.

Porte-avion : *bateau_1 = 5*
Croiseur : *bateau_2 = 4*
Sous-marins : *bateau_3 = 3* et *bateau_4 = 3*
Torpilleur : *bateau_5 = 2*

C'est cette variable niveau de vie qui sera modifiée quand le bateau sera touché.
Ces variables seront stockées dans une liste bateaux qui sera de la forme
bateaux = [niveau de vie bateau_1 , niveau de vie bateau_2, …]
Saisie des coordonnées: Pour la saisie des coordonnées, on demandera via un 'input' au joueur de saisir des coordonnées sous la forme 'B3' ou la lettre correspond à la colonne et le chiffre à la ligne. Il faudra donc une fonction qui va vérifier la validité de la saisie (bien sous la bonne forme, que les lettres sont bien dans la plage des lettres qui correspond à la taille de la grille…)
Arrêt de jeu : 2 variables *somme_1* et *somme_2* qui correspondent respectivement à la somme des niveaux de vie de tous les bâteaux des *joueur_1* et *joueur_2*
Lorsque l'une de ces deux sommes atteint 0 alors l'un des joueurs a perdu tous ses bâteaux donc le jeu est fini.

- <u>Si 2 joueurs :</u>

Joueur 1 place ses bateaux sur la *grille de position 1*
Joueur 2 place ses bateaux sur la *grille de position 2*
Tant que *somme_1* est non nulle ou *somme_2* est non nulle:

Tour de *joueur 1* :
Affichage *grille de jeu 1*
Saisir coordonnées de tir (Ajout d'une vérification de la validité des coordonnées)
Si le joueur a déjà tiré dans cette cellule :
Afficher "Vous ne pouvez pas tirer 2 fois au même endroit"
joueur 1 joue à nouveau
Sinon :
Si bateau touché :
Placer un X sur la cellule visée de la *grille de jeu 1*
Soustraire 1 à la niveau de vie touché
Soustraire 1 à *somme_1*
Si niveau de vie du bateau touché = 0 :
Afficher "touché, coulé"
Sinon :
Afficher "touché"
joueur 1 joue à nouveau
Si aucun bateau n'est touché :
Placer un O sur la cellule visée de la *grille de jeu 1*
Tour de *joueur 2*

Tour de joueur 2 :

Affichage *grille de jeu 2*
Saisir coordonnées de tir
Si coordonées invalides :
Afficher Message d'erreur
joueur 1 joue à nouveau
Sinon :
Si bateau touché :
Placer un X sur la cellule visée de la *grille de jeu 2*
Soustraire 1 à la niveau de vie du bâteau touché
Soustraire 1 à *somme_2*
Si niveau de vie du bateau touché = 0 :
Afficher "touché, coulé"
Sinon :
Afficher "touché"
joueur 2 joue à nouveau
Si aucun bateau n'est touché :
Placer un O sur la cellule visée de la *grille de position 2*
Tour de *joueur 1*

- <u>Fin de jeu :</u>

Si tous les bâteaux du joueur 1 sont touchés en premier :
Afficher "Le joueur 2 remporte la partie"
Si tous les bâteaux du joueur 2 sont touchés en premier :
Afficher "Le joueur 1 remporte la partie"

BONUS : L'objectif est de faire jouer l'ordinateur autrement que seulement en tirant aléatoirement. Celui-ci tire aléatoirement jusqu'à toucher un bâteau. Dès lors, il tire au voisinage de la cellule contenant le bâteau touché jusqu'à ce que celui-ci soit coulé. Il reprend ensuite le processus jusqu'à la fin du jeu.
Cette partie du programme ne sera réalisée que si nous disposons du temps suffisant.

- <u>Si un joueur :</u>

Ajout d'une variable booléenne : *tir_prededent_reussi*
Si un bateau est touché au tir précédent : True Sinon : False
Ajout d'une variable *tir_precedent* qui stocke les coordonnées du tir précédent

Joueur 1 place ses bâteaux sur la *grille de position 1*
Génération aléatoire des positions des bâteaux sur la *grille de position 2*
Tant que *somme_1* est non nulle et *somme_2* est non nulle:
Tour de *joueur 1* :
Affichage *grille de jeu 1*
Saisir coordonnées de tir
Si coordonées invalides :

Afficher Message d'erreur
joueur 1 joue à nouveau
Sinon :
Si bateau touché :
Placer un X sur la cellule visée de la *grille de jeu 1*
Soustraire 1 à niveau de vie du bâteau touché
Soustraire 1 à *somme_1*
Si niveau de vie du bateau touché = 0 :
Afficher "touché, coulé"
Sinon :
Afficher "touché"
joueur 1 joue à nouveau
Si aucun bateau n'est touché :
Placer un O sur la cellule visée de la *grille de jeu 1*
Tour de *ordinateur*

Tour de ordinateur :
Si *tir_prededent_reussi* a pour valeur *False* :
Génération coordonnées de tir aléatoires
Si l'ordinateur a déjà tiré dans cette cellule :
ordinateur joue à nouveau
Sinon :
Si bateau touché :
Soustraire 1 à niveau de vie
Soustraire 1 à *somme_2*
tir_prededent_reussi prend la valeur True
tir_precedent prend pour valeur les coordonnées du tir
Si niveau de vie du bateau touché = 0 :
Afficher "touché, coulé"
Sinon :
Afficher "touché"
ordinateur joue à nouveau
Si aucun bateau n'est touché :
Placer un O sur la cellule visée de la *grille de position 2*
Tour de *joueur 1*

Si *tir_prededent_reussi* a pour valeur *True* :
Ordinateur tir dans une cellule voisine (haut, bas, droite, gauche) de la cellule de coordonnées : *tir_precedent*.
Si l'ordinateur a déjà tiré dans cette cellule :
ordinateur joue à nouveau
Sinon :
Si bateau touché :
Soustraire 1 à niveau de vie
Soustraire 1 à *somme_2*
tir_prededent_reussi prend la valeur True
tir_precedent =
Si niveau de vie du bateau touché = 0 :

Afficher “touché, coulé”
Sinon :
Afficher “touché”
ordinateur joue à nouveau
Si aucun bateau n’est touché :
Placer un O sur la cellule visée de la *grille de position 2*
Tour de *joueur 1*