
Projet XmasTreats

Nom du projet : SmaXelle

AGEZ Axelle

aagez@ensc.fr

HEBBAR Esma

ehbbbar001@ensc.fr

```
SSSSSSSSSSSSSSS          XXXXXX      XXXXXX      1111111 1111111
SS::::::::::::S          X::::X      X::::X      1::::1 1::::1
S::::SSSSS::::S          X::::X      X::::X      1::::1 1::::1
S::::S      SSSSSS      X::::X      X::::X      1::::1 1::::1
S::::S          mmmmmmm      mmmmmmm      aaaaaaaaaa      XXX::::X      X::::XXX      eeeeeeeeeee      1::::1 1::::1      eeeeeeeeeee
S::::S          mm:::::m      m:::::mm      a::::::::::a      X::::X      X::::X      ee::::::::::ee      1::::1 1::::1      ee::::::::::ee
S::::SSSS      m::::::::::mm::::::::::m      aaaaaaaaaa::::a      X::::X::::X      e:::::eeee:::::eeel::::1 1::::1      e:::::eeee:::::ee
SS:::::SSSSS      m::::::::::m::::::::::m      a::::a      X::::::::::X      e:::::e      e:::::el::::1 1::::1      e:::::e      e:::::e
SSS:::::SSSS      m:::::mmm:::::mmm:::::m      aaaaaa::::a      X::::::::::X      e:::::eeee:::::eeel::::1 1::::1      e:::::eeee:::::ee
SSSSSS::::S      m::::m      m::::m      m::::m      aa::::::::::a      X::::X::::X      e:::::eeee:::::ee      1::::1 1::::1      e:::::eeee:::::ee
S:::::Sm::::m      m::::m      m::::m      m::::m      a::::aaaa::::a      X::::X      X::::X      e:::::eeeeeeeeee      1::::1 1::::1      e:::::eeeeeeeeee
S:::::Sm::::m      m::::m      m::::ma::::a      a::::a      XXX::::X      X::::XXXe:::::e      1::::1 1::::1      e:::::e
SSSSSS      S:::::Sm::::m      m::::m      m::::ma::::a      a::::a      X::::X      X::::Xe:::::e      1:::::ll:::::le:::::e
S:::::SSSSS:::::Sm::::m      m::::m      m::::ma::::aaaa::::a      X::::X      X::::X      e:::::eeeeee      1:::::ll:::::l      e:::::eeeeee
S:::::SSSSSSSSSS      m::::m      m::::m      m::::m      a::::::::::aa::::aX::::X      X::::X      ee::::::::::e      1:::::ll:::::l      ee::::::::::e
SSSSSSSSSSSSSS      mmmmmmm      mmmmmmm      mmmmmmm      aaaaaaaaaa      aaaaXXXXXX      XXXXXX      eeeeeeeeeee      1111111111111111      eeeeeeeeeee
```

Année : 2023/2024

Table des matières

Table des matières.....	2
I. Introduction.....	3
II. Organisation générale.....	4
A. Organisation de l'équipe.....	4
B. Répartition des tâches dans le groupe.....	4
C. Planning de réalisation.....	4
III. Choix techniques.....	5
A. Les choix de modélisation des données.....	5
B. La structuration du code.....	5
1. La boucle do-while.....	5
2. Le découpage en sous-programmes.....	6
3. La dynamique d'exécution.....	6
C. Les fonctions.....	7
1. AfficherLeJeu().....	7
2. PlacerBonbonsAleatoirement ().....	7
3. FusionGoRight ().....	7
4. DecallerLesTreatsDroite().....	8
IV. Bonus mis en place.....	10
A. Arrêt volontaire du jeu.....	10
B. Coups supplémentaires.....	10
C. Affichage de la cause de fin de jeu.....	10

I. Introduction

Le but de ce projet est de programmer en équipe un jeu similaire au jeu 2048 en C#, avec une approche procédurale. Le jeu se déroule sur un plateau de jeu, à l'aide de quatre touches de clavier, représentant les quatre directions de l'espace. L'utilisateur commence sur un plateau de jeu "initialisé" : seuls deux *treats* sont présents. Le jeu s'arrête lorsque toutes les cases du plateau de jeu sont occupées, lorsque le nombre de coups maximum, déterminé par l'utilisateur, est atteint, ou lorsque l'utilisateur le demande.

La principale contrainte technique a été l'utilisation du langage de programmation, qui devait être du C#, avec une approche procédurale.

II. Organisation générale

A. Organisation de l'équipe

L'équipe se compose de Esma Hebbbar et de Axelle Agez, deux élèves-ingénieures de Première Année à l'ENSC.

La collaboration et la communication dans l'équipe s'est faite par des réunions régulières, des mises à jour de statut du programme, mais également par l'utilisation de GitHub, un outil de collaboration en ligne.

L'équipe a ainsi pu s'appuyer sur des entretiens oraux et de la communication numérique.

B. Répartition des tâches dans le groupe

La répartition des tâches concernant ce projet a été mise en place à la première séance de travail.

Esma Hebbbar a pris en charge la création et l'affichage du plateau de jeu, élément essentiel à la poursuite du projet. Par la suite, elle a géré le déplacement et la fusion des différents bonbons sur ce plateau.

Axelle Agez a, quant à elle, pris en charge les éléments concernant la présentation du jeu dans la console. Elle s'est occupée du fonctionnement du jeu, de ses fonctionnalités environnantes et des bonus.

C. Planning de réalisation

Etapes du projet	Délais associés
Partage et mise en place de l'organisation	2h - 24 novembre
Création et affichage du plateau / début du jeu et consignes	2 jours - 29 novembre au 31 novembre
Création des fonctions annexes	2 jours - 1 au 2 décembre
Création des fonctions pour déplacer et fusionner les bonbons / création des fonctions pour placer des bonbons aléatoirement / création de la boucle "do-while" et de ses composantes	1 semaine - 2 au 9 décembre
Correction des bugs / insertion de bonus	2 jours - 9 au 11 décembre
Finalisation	1 jour - 12 décembre

III. Choix techniques


A. Les choix de modélisation des données




Les choix concernant la modélisation des variables et fonctions sont les suivants :

















- utilisation de la norme camelCase pour les variables et de la norme PascalCase pour les fonctions ;
- utilisation d'un booléen pour savoir si le jeu est bloqué : façon de comparer et de savoir si condition est remplie ;
- utilisation de tableaux imbriqués pour créer notre plateau de jeu. Ce choix s'explique par l'intention de faciliter la compréhension dans l'équipe en s'appuyant sur nos connaissances communes, ainsi que par l'intention de faciliter l'application des autres démarches liées à ce plateau :

Tableau représentatif du plateau de jeu créé

Légende :

 cases de jeu, où seront placés les *treats*

   délimitations du plateau de jeu et des cases de jeu

	0	1	2	3	4	5	6	7	8
0	.	-	-	-	-	-	-	-	.
1									
2		-		-		-		-	
3									
4		-		-		-		-	
5									
6		-		-		-		-	
7									
8	.	-	-	-	-	-	-	-	.

B. La structuration du code

1. La boucle do-while

La boucle principale, qui représente les différents tours de la partie, prend la forme d'une boucle "do-while".

La boucle "do-while" a été choisie puisqu'elle permet d'effectuer un premier tour avant d'insérer les conditions. La première condition est que le nombre de tours doit être inférieur ou égal au nombre de coups maximum choisi par le joueur au début. La seconde condition est que le plateau ne doit pas être bloqué pour poursuivre.

Ainsi, la boucle "do-while" permet de faire reposer un enchaînement d'actions sur deux conditions qui, si une d'elles ou les deux sont remplies pendant le tour, arrêtent le jeu.

2. Le découpage en sous-programmes

A maintes reprises, le programme fait appel à des sous-programmes qui ont été créés en amont. Le but de ce découpage en sous-programme est de permettre une meilleure lisibilité du code.

Les sous-programmes sont utilisés pour les tâches répétitives revenant à chaque tour ou du moins à plusieurs moments du code. Bien que le programme s'appuie sur une boucle pour supprimer la répétition d'un même enchaînement, mais les sous-programmes en facilitent la compréhension.

Les sous-programmes correspondent aux principales actions :

- afficher le plateau de jeu ;
- placer les bonbons aléatoirement dans le plateau de jeu ;
- déplacer et fusionner les bonbons ;
- chercher le nombre de cases vides.

3. La dynamique d'exécution

Le programme suit une dynamique d'exécution précise et réfléchie pour rendre le jeu agréable et lisible.

Tout d'abord, le plateau de jeu s'initialise. En effet, le programme commence par afficher le nom du jeu et les règles, puis il crée le plateau de jeu, représenté par un tableau bidimensionnel de caractères. Ainsi, la phase de remplissage du plateau de jeu commence. Le plateau est rempli avec des caractères spécifiques ("I", "-" et ".") pour dessiner les bordures et les délimitations. Les autres cases sont remplies avec des espaces " " pour indiquer qu'elles sont vides. Cette étape se termine par l'affichage du plateau.

La demande au joueur de son nombre maximum de coups suit l'initialisation.

Par la suite, le jeu débute et les consignes s'affichent. Le programme informe le joueur que le jeu va commencer et énonce toutes les règles. Les directions E, D, S, F, associées aux mouvements sur le plateau, sont explicitées au joueur.

Avant que la boucle principale, qui représente les différents tours, démarre, la fonction PlacerBonbonsAleatoirement prend place et inscrit deux bonbons de niveau 1 dans le plateau de jeu. Par la suite, cette fonction sera appelée plusieurs fois.

La boucle principale du jeu suit ainsi cette initialisation. Le jeu est exécuté dans une boucle do-while qui continue tant que le nombre de tours est inférieur ou égal au nombre de coups maximum et que le plateau n'est pas bloqué. A chaque tour, le joueur choisit une direction, les bonbons fusionnent et le plateau est mis à jour. De nouveaux bonbons sont placés aléatoirement et le plateau est affiché de nouveau.

Lorsque la boucle principale se termine, le processus de fin de jeu est enclenché. Le programme détermine et affiche la cause de l'interruption : le plateau est bloqué, le nombre de coups maximum est atteint, ou le joueur a arrêté sa partie de façon volontaire.

A la suite, le score de la partie est calculé et affiché. Pour cela, le programme parcourt chaque case du plateau, le score dépendant du type et du nombre de bonbons présents dans le plateau. Le score est ainsi affiché et conclut cette partie de jeu Smaxelle.

C. Les fonctions

1. AfficherLeJeu()

```
void AfficherLeJeu(char[] [] tab)
```

Paramètre :

`char[] [] tab` : un tableau bidimensionnel (un tableau de tableau) de caractères ('char')

Description :

Cette fonction sera régulièrement appelée, et permettra alors d'afficher la totalité du plateau de jeu, soit les délimitations du plateau de jeu, ainsi que les *treats* contenues dans les cases de jeu.

Elle ne retourne rien : le simple affichage suffit.

2. PlacerBonbonsAleatoirement ()

```
void PlacerBonbonsAleatoirement (char[] [] tab)
```

Paramètre :

`char[] [] tab` : un tableau bidimensionnel (un tableau de tableau) de caractères ('char')

Description :

Cette fonction sera elle aussi régulièrement appelée, et permettra de placer deux bonbons 'α' de manière aléatoire, dans deux des cases vides du plateau de jeu.

3. FusionGoRight ()

```
void FusionGoRight(char[][] tab)
```

Paramètre :

`char[][] tab` : un tableau bidimensionnel (un tableau de tableau) de caractères ('char')

Description :

Cette fonction permet de fusionner deux éléments identiques, et de mettre l'élément résultant de la fusion dans la case la plus à droite entre les deux cases évaluées, pour toutes les lignes du tableau `tab`.

Détails de la fonction :

n° colonne	0	1	2	3	4	5	6	7	8
ligne n° i									
		var4		var3		var2		var1	

Pour chaque ligne n°i, avec i prenant les valeurs {1, 3, 5, 7} car ce sont les seules "lignes de jeu", le but est d'évaluer dans quelle situation nous nous trouvons, afin de fusionner les bons *treats*, et mettre le résultat de la fusion dans la case la plus à droite, entre les deux cases évaluées.

Il existe différents cas, récapitulés dans le tableau suivant :

FusionGoRight : l'utilisateur a demander d'aller à droite.							
n°	condition correspondante au if/else if			exemple de situation			
#0	var1==var2 && var1==var3 && var1==var4 et var1!= ' '			o	o	o	o
#1	var1 != ' ' && var1 == var2			X	X	@	@
#2	var1!=' ' && var1!=var2 && var2!=' ' && var2==var3 && var3==var4			@	@	@	o
#9	var1!=' ' && var1==var2 && var2==var3 && var4!=var1			X	@	@	@
#3	var2 != ' ' && var3 == var2			X	@	@	X
#4	var4 != ' ' && var3 == var4			@	@	X	X
#5	var4 != ' ' && var2 == ' ' && var3 == ' ' && var1 == var4			@			@
#6	var1 != ' ' && var2 == ' ' && var3 == var1			X	@		@
#7	var4 != ' ' && var1 == ' ' && var3 == ' ' && var2 == var4			@		@	
#8	var4 != ' ' && var1 != ' ' && var3 == ' ' && var2 == var4 && var1!=var2			@		@	o
				var4	var3	var2	var1

Légende : les ' X ' signifient que la case peut-être occupée par n'importe quoi (vide ' ', 'α', '@', 'o' ou encore 'J'). Plusieurs ' X ' sur une même ligne peuvent prendre des *char* identiques ou différents, peu importe ici.

Remarque : par soucis de lisibilité, les *treats* illustrées ici sont des réglisses '@' et des cookies 'o', néanmoins les conditions restent les mêmes dans les cas des autres *treats*.

Le principe est exactement le même pour les fonctions `FusionsGoLeft()` , `FusionGoUp()` et `FusionGoDown()`.

4. DecallerLesTreatsDroite()

```
void DecallerLesTreatsDroite(char[][] tab)
```

Paramètre :

`char[][] tab` : un tableau bidimensionnel (un tableau de tableau) de caractères ('char')

Description :

Cette fonction est appelée après l'utilisation de la fonction `FusionGoRight(char[][] tab)` afin "d'écraser" les *treats* vers la droite.

Le principe est exactement le même pour les fonctions `DecallerLesTreatsGauche()` , `DecallerLesTreatsHaut()` et `DecallerLesTreatsBas()`.

IV. Bonus mis en place

A. Arrêt volontaire du jeu

A la fin de chaque tour, le programme offre au joueur la possibilité d'arrêter sa partie, même si le nombre de coups maximum n'est pas atteint et que le plateau de jeu est encore utilisable.

S'il le souhaite, il doit écrire dans la console "oui".

Cette demande prend la forme d'une boucle if.

B. Coups supplémentaires

Si le joueur a atteint le nombre maximum de coups qu'il s'est autorisé en début de partie, ce bonus met en place une énigme afin de lui permettre de gagner 3 coups supplémentaires.

L'énigme n'est modifiable que dans le programme, puisque le joueur ne peut accéder à cette option qu'une seule fois au cours de sa partie.

S'il l'utilise, la variable "nbDeFoisEnigme" est incrémentée de 1 et empêche un second usage de ce bonus.

L'énigme est présentée par une boucle if.

C. Affichage de la cause de fin de jeu

Le jeu peut se finir de trois manières différentes :

- lorsque le plateau de jeu est bloqué et ne permet plus aucun déplacement
- lorsque le nombre de coups maximum, choisi par le joueur au début de la partie, est atteint
- lorsque le joueur met fin intentionnellement à la partie

Ainsi, avant l'annonce du score, le joueur se voit annoncer la cause de l'arrêt de la partie. Cet affichage lui permet de comprendre la raison de cette interruption si elle n'était pas volontaire.

L'affichage n'utilise pas une fonction mais une boucle if.