

# PLANTAMIZ 2025

---

## Table des matières

Introduction .....	2
Cahier des charges.....	2
Configuration initiale .....	2
Règles pour « manger » un item .....	2
Règles pour compter les points .....	3
Travail à réaliser et évaluation .....	4
Contraintes matérielles et logicielles.....	4
Planning et organisation du travail.....	4
Travail demandé, contraintes et consignes.....	6
1- Analyse et conception générale.....	6
2- Analyse et conception détaillée.....	6
3- Développement dans le langage C (codage, tests). .....	6
4- Critères de notation. ....	6
Annexes .....	7
1- Table ANSI. ....	7
2- La manipulation du curseur sous Windows. ....	8
3- La gestion de la couleur sous Windows. ....	8

## Introduction

Le jeu que nous vous proposons cette année comme projet informatique est inspiré du jeu *"FarmHeroes"* que vous trouvez gratuitement sur smartphone. Ceux qui auront la curiosité de tester ce jeu comprendront le nom de ce projet. Adapté pour convenir au mode console, le jeu garde malgré tout l'esprit du jeu d'origine dans son principe : aligner des items pour les faire disparaître (on retrouve également ce principe dans le célèbre *CandyCrush*).

## Cahier des charges

Le plateau de jeu se présente comme une **matrice de caractères 25 lignes x 45 colonnes** contenant aléatoirement les lettres **"S"** pour *soleil*, **"F"** pour *fraise*, **"P"** pour *pomme*, **"O"** pour *oignon*, et **"M"** pour *mandarine*. Le joueur a pour **objectif de remplir un contrat** qui lui est annoncé au début du tableau. Il correspond à un **nombre préfixé d'items à "manger"** en un **nombre préfixé de coups**. Le contrat est différent sur chaque tableau et la difficulté est croissante.

Exemple de contrats :

- tableau 1 : 20 fraises, 50 oignons et 20 mandarines en 30 coups maximum
- tableau 2 : 55 soleils, 35 pommes, 30 fraises et 45 oignons en 40 coups maximum

Afin que cela soit lisible, **chaque lettre devra avoir une couleur différente** (exemple du rouge pour la fraise, jaune pour soleil...). **pomme = vert, oignon = violet, mandarine = orange**  
Pour réussir un niveau, le joueur aura un **temps imparti** que vous fixerez vous-même en fonction de la difficulté du tableau. Le **décompte du temps doit être visible sur la console**.

## Configuration initiale

Le tableau de caractères 25 x 45 **aléatoirement initialisé** est affiché à l'écran. Vous devez vous assurer qu'**aucune suppression n'est possible au début du niveau** (pas 3 ou 5 items consécutifs de même nature ou de figures spéciales).

**Tout groupe consécutif en ligne verticale ou horizontale de 3 items ou plus disparaît et rapporte des « points contrat »** au joueur si l'item correspond. En cas de **figure spéciale (H ou rectangle)** une bonification s'applique comme décrit ci-dessous.

Les items « mangés » créent des trous qui seront comblés par gravité en faisant tomber les caractères au-dessus. Les trous qui se forment en haut de colonne sont comblés aléatoirement par de nouveau items.

Une fois les trous comblés, les règles de suppression s'appliquent de nouveau automatiquement et les trous sont de nouveau comblés... jusqu'à stabilisation.

Le joueur récupère alors la main sur un tableau qui ne contient plus aucun groupe d'items consécutifs de 3 ou 5 ou de figures spéciales.

## Règles pour « manger » un item

Le joueur doit maintenant constituer des groupes pour remplir son contrat. Pour cela, il doit permuter les items pour les réarranger et former des groupes sur lesquels les règles de suppression s'appliqueront.

Il peut déplacer le curseur à l'aide des touches *z, q, s, d* du clavier (ou les flèches) jusqu'à un item et appuyer sur la barre d'espace. L'item est alors sélectionné et mis en évidence (en passant le caractère en minuscule par exemple). (Un second appui sur la barre d'espace désélectionnerait l'item). Il appuie ensuite de nouveau sur une des touches *z, q, s, d* pour permuter l'item sélectionné avec l'item choisi à gauche, à droite, au-dessus ou en-dessous (attention aux limites du tableau qui rendent impossible les permutations vers l'extérieur des items en bordure par exemple).

Si l'item nouvellement positionné forme un groupe consécutif de 3 minimum ou une figure, les règles de suppression s'appliquent. Si la permutation ne forme aucun groupe, le joueur peut de nouveau déplacer le curseur pour effectuer une autre permutation. (dans cette version de jeu, il peut donc permuter des items isolés). Les règles de suppression s'appliquent après chaque permutation.

### Règles pour compter les points

- un groupe de 4 items (horizontal ou vertical) de même nature rapporte 4 points
- un groupe de 6 items (horizontal ou vertical) de même nature provoque la disparition de tous les items identiques présents sur le tableau, même s'ils sont isolés, et remporte autant de points que d'items disparus
- un groupe d'items de même nature en forme de H rapporte  $2 \times X$  points avec X le nombre d'items formant le H
- un groupe d'items de même nature formant un rectangle rapporte  $2 \times (L \times l)$  points avec L = longueur et l = largeur du rectangle.

Exemples :

OOMP

OOOOF

FFMSF

4 oignons à récupérer ici, qui valent 4 points *oignons*

MFSOSFM

MMFSOSF

OSFFPSM

MOSOSPF

OSSOPMF

PMMMMM

Cette configuration rapporte au joueur 13 points *mandarine*

MSSSOSF

OSFFFSM

MSFSMPF

OSSSFOO

Cette configuration rapporte au joueur 24 points *soleil* car on a un rectangle de 4 x 3 soleils. Lorsque des items sont mangés, les items présents au-dessus tombent pour prendre leur place. Les vides qui sont laissés au sommet de chaque colonne sont comblés par de nouveaux items aléatoires qui tombent eux aussi pour combler les trous. Si le joueur remplit son contrat dans le nombre de coups préfixé, le jeu lui propose de passer au tableau suivant ou de quitter le jeu. En cas d'échec, le joueur consomme une vie et le jeu lui propose de recommencer le même niveau ou de quitter le jeu.

Au bout de 5 vies, le joueur a perdu et le jeu s'arrête. Le programme enregistre dans un fichier le nom du joueur et les niveaux accomplis pour que la prochaine partie commence sur le prochain tableau à accomplir. Le fait d'accomplir un tableau ou de quitter le jeu remet à cinq le compteur de vie.

## Travail à réaliser et évaluation

Pour le jeu de base, nous vous demandons de **créer 3 tableaux de difficulté croissante** en termes de contrat à remplir.

Le jeu de base devra également permettre au joueur de **sauvegarder son nom et sa progression avant de quitter le jeu** sur un tableau terminé ou raté.

Le jeu de base sera évalué sur 16 points. Les 4 points de bonus restants seront attribués pour des projets offrant des **fonctionnalités supplémentaires** (par exemple : taille de matrice paramétrable, sauvegarde/rechargement, ...) ou présentant une originalité particulière laissée à la libre imagination des auteurs. Toute originalité est **INTERDITE** dans le premier tableau qui doit respecter strictement le CDC.

L'originalité peut s'exprimer sur les aspects suivants (liste non exhaustive) :

- items originaux en plus des 5 proposés et comportement spécifique pour les items originaux
- musique (attention, cette option doit être désactivable via un menu paramètre)
- nouveau type de contrat sur certains tableaux
- le COVID-19 arrive et infecte des lettres aléatoirement qui deviennent inutiles...
- ...

## Contraintes matérielles et logicielles

Le programme sera codé en C sous l'IDE **Code::Blocks** sur Windows ou équivalent sur MacOS et devra fonctionner correctement sur votre PC et le PC de votre correcteur.

Le cahier des charges (CDC) du sujet est à respecter impérativement pour la soutenance. Commencez par le décomposer en étapes et réalisez chacune des étapes proprement avant de passer à la suivante. Seulement une fois terminé le CDC, laissez libre court à votre originalité pour des petits ajouts appréciables.

Nous vous recommandons d'utiliser un outil de versionning type GIT pour faciliter la collaboration entre les membres de l'équipe. Dans tous les cas, vous conserverez toujours une version "exécutable" partielle en parallèle de la version dans laquelle vous continuez à développer l'étape suivante de votre analyse (Vous aurez donc logiquement à la fin autant de versions de votre projet que d'étapes) : voir les différentes étapes d'analyses, de conception et de développement dans la section [Travail demandé, contraintes et consignes](#).

## Planning et organisation du travail

Le planning, sujet, discord et évaluation de trouve sur la page BoostCamp.

**Période de réalisation du projet** : de la semaine du 20/10/2025 à la semaine du 24/11/2025 incluse.

### **Soutenances :**

- Les soutenances en TP sur le **livrable de mi-parcours de conception** auront lieu la semaine du 17/11/2025, contenant le compte-rendu de votre livrable sans code (au format **.pptx .pptx** ou **.pdf**), exactement dans le deadline du dimanche 16/11/2025 23h55 sur BoosCamp, selon les consignes précises à respecter qui vous seront fournies.

- Les soutenances de la **version finale** auront lieu la semaine du 01/12/2025, contenant le code de la version finale de soutenance, selon les consignes précises à respecter qui vous seront fournies. Suite au dernier livrable final à rendre, les **soutenances finales** auront lieu **la semaine suivante du 01/12 en séance de TP et par équipe**. Votre note de soutenance de dernier livrable final comptera pour 75% de votre note de projet.

**Équipes** : 3 ou 4 dans un même groupe de TD.

**Evaluation par rendu et soutenance :**

- La note du projet sera la moyenne pondérée de version finale de soutenance Coef 3 (75%) de votre code sur 20 contre un coef 1 ((25%)) et du PowerPoint sur 20 Coef 1 (25%) de la version de mi-parcours, les 2 livrables étant à déposer sur BoostCamp.
- Après calcul de cette moyenne, seront appliquées d'éventuelles pénalités (retard de dépôt, fichiers manquants dans le dépôt, plagiat, manque d'implication individuelle etc.). Ce qui donnera enfin votre note finale de projet.

La soutenance sera d'environ 20 minutes par équipe, incluant une présentation PowerPoint pour la conception et une démonstration du projet, en respect du **cahier des charges demandé ci-dessus, ainsi que des contraintes et consignes ci-dessous**

## Travail demandé, contraintes et consignes

### 1- Analyse et conception générale.

- A partir du cahier des charges (CDC) : extraire les données pertinentes, si besoin les regrouper (structures, tableaux), spécifier et caractériser les fonctionnalités (sous-programmes) nécessaires, en déduire les principaux algorithmes de ces fonctionnalités, ...
- Interface Homme-Machine : lister les choix à offrir au démarrage, déterminer l'organisation et le contenu de l'écran de jeu, ...
- Répartir les tâches au sein de l'équipe.

### 2- Analyse et conception détaillée

- A partir de vos algorithmes, déduits des fonctionnalités, lister les prototypes de toutes les sous-programmes requis en précisant ses paramètres d'entrée et de sortie.
- Réaliser progressivement une maquette du jeu en testant au fur et à mesure du codage et en tenant compte des différents scénarios.
- Entre autres critères de qualité, le programme final devra être très facilement adaptable par tout autre développeur (exemples : changement des valeurs d'initialisation, changement des caractères et couleurs d'affichage, ...).

### 3- Développement dans le langage C (codage, tests).

Dans le langage C, implémenter le jeu en respect de votre analyse des étapes précédentes : organisation modulaire multi-fichiers (avec headers(s) .h et fichier(s) source(s) .c), commenter les prototypes des sous-programmes (dans le header en précisant les paramètres IN/OUT), vos sous-programmes, votre programme principal. **N'oubliez pas de tester votre code après chaque sous-programme.**

### 4- Critères de notation.

Vous devez réaliser l'ensemble du cahier des charges. Le langage de programmation doit être le langage C... **Votre code devra être modulaire, respecter l'interface du jeu en mode console et bien commenté !**

**Un code qui ne compile pas ou qui plante au démarrage ne vaut pas plus de 10/20. Tester donc votre programme avant de le déposer sur BoostCamp...**

Votre travail sera jugé sur les critères suivants :

- Le respect rigoureux du cahier de charge (CDC) du jeu énoncé précédemment
- La modularité de votre conception et donc de votre code
- La bonne répartition des tâches entre les membres de l'équipe
- L'intérêt, l'originalité, la jouabilité et toutes les caractéristiques que vous prendrez soin de mettre en avant lors de la soutenance.

## Annexes

### 1- Table ANSI.

La table ANSI ci-dessous vous permet d'afficher des caractères semi-graphiques à l'écran : pour le caractère choisi, afficher **0x** pour le code hexadécimal suivi du chiffre de la colonne et du chiffre de la ligne. Par exemple, le code ANSI en hexadécimal du cœur est 0x03, celui du carré est 0xDB etc.

Page de codes 850

	0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240
	0-	1-	2-	3-	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
3	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
5	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
6	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
7	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
9	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
10	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
11	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
12	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
13	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
14	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
15	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Exemple en C :

```
int main ()
{
    // exemple pour afficher un cœur suivi d'un carré

    printf("%c %c", 0x03, 0xDB); // 0x pour le codage hexadécimal
    return 0;
}
```

## 2- La manipulation du curseur sous Windows.

Pour ce projet, vous aurez besoin d'une fonction de manipulation du curseur à l'écran. La fonction `gotoligcol` ci-dessous vous permet de positionner le curseur à la ligne "lig" et à la colonne "col". Vous devez copier les lignes suivantes avant votre main (l'exemple des instructions du main ci-dessous ne sont bien sûr pas à copier, sauf la dernière toujours obligatoire).

```
#include <windows.h>
```

```
void gotoligcol( int lig, int col )
```

```
{  
    // ressources  
    COORD mycoord;  
  
    mycoord.X = col;  
    mycoord.Y = lig;  
    SetConsoleCursorPosition( GetStdHandle( STD_OUTPUT_HANDLE ), mycoord );  
}
```

```
int main ()
```

```
{  
    // exemple d'appel au niveau du main  
    // ....  
  
    gotoligcol(5,15); // place le curseur ligne 5 colonne 15 à l'écran  
  
    printf("bonjour"); // et écrit bonjour à cet endroit  
  
    return 0;  
}
```

## 3- La gestion de la couleur sous Windows.

Vous aurez aussi besoin d'une fonction de gestion de couleur à l'écran. La fonction `color` ci-dessous vous permet de changer la couleur du texte dans la console, ainsi que la couleur du fond pour la ligne. Vous devez copier les lignes suivantes avant votre main (l'exemple des instructions du main ci-dessous ne sont bien sûr pas à copier, sauf la dernière toujours obligatoire).

```
#include <stdio.h>  
#include <stdlib.h>  
#include <windows.h>
```

```
////////// PROTOTYPES DES SOUS-PROGRAMMES  
void Color(int couleurDuTexte,int couleurDeFond);
```



```
////////////////////
// Nom du sous-programme : COLOR
// Rôle : change la couleur du texte dans la console, ainsi que la couleur du fond pour la
// ligne
// Compatibilité : windows XP
////////////////////
void Color(int couleurDuTexte,int couleurDeFond) // fonction d'affichage de couleurs
{
    HANDLE H=GetStdHandle(STD_OUTPUT_HANDLE);
    SetConsoleTextAttribute(H,couleurDeFond*16+couleurDuTexte);
}

int main()
{

    printf("Hello world!\n");

    /*
    0 : Noir
    1 : Bleu foncé
    2 : Vert foncé
    3 : Turquoise
    4 : Rouge foncé
    5 : Violet
    6 : Vert caca d'oie
    7 : Gris clair
    8 : Gris foncé
    9 : Bleu fluo
    10 : Vert fluo
    11 : Turquoise
    12 : Rouge fluo
    13 : Violet 2
    14 : Jaune
    15 : Blanc
    */

    // essai numero 1
    Color(12,3);
    printf("Le texte va avoir la couleur 12 et le Fond la couleur 3\n");

    // essai numero 2 : on repasse au noir et blanc
    Color(15, 0);
    printf("Hello world!\n");

    // essai 3 : un peu de turquoise
    Color(11, 0);
```

```
printf("Hello world!\n");  
  
// et la derniere couleur reste jusqu'à ce qu'on change de nouveau  
printf("Hello world!\n");  
  
//...  
  
return 0;  
}
```