

---

# PROJET INFORMATIQUE 1

## ING1

### « ECE HEROES »



*Farm Heroes Saga (niveau 190)*

---

I. Présentation. ....	2
II. Objectifs. ....	2
III. Organisation et planning. ....	3
IV. Cahier des charges détaillé. ....	4
1. Fonctionnement du jeu de base. ....	4
2. Contraintes techniques. ....	7
3. IHM. ....	8
4. Extensions. ....	9
V. Travail à réaliser, rendus et modalités d'évaluation .	10

## I. Présentation.

Le projet d'Informatique de ce premier semestre a pour objectif de concevoir un petit jeu de type « match-3 ».



Candy Crush Saga for Android

Les jeux de type « match-3 » sont apparus au début des années 2000 et ont connu un immense succès sur les plateformes mobiles. Ils reposent sur un principe simple et addictif : associer trois éléments identiques pour les faire disparaître. Parmi les plus célèbres, on peut citer *Farm Heroes* ou *Candy Crush*.

**Tout en s'inspirant de ces jeux célèbres, votre jeu devra faire preuve d'originalité avec des règles et des défis revisités.**

**Le jeu sera intégralement développé en langage C et en mode console, avec un affichage textuel.**

## II. Objectifs.

A travers la mise en pratique des démarches et des différentes notions vues durant le cours d'Algorithmique et Programmation structurée, les objectifs de ce projet en termes d'acquisition de compétences sont multiples :

- Comprendre et mener à bien les différentes étapes de conception et développement d'un programme informatique :

- o Analyse d'un cahier des charges
  - o Modélisation et conception de la solution
  - o Réalisation : implémentation dans un langage informatique (langage C)
  - o Tests et validation
  - o Livraison
- Apprendre à travailler en équipe.
  - Planifier un projet et respecter les délais impartis.
  - Apprendre à concevoir un programme modulaire et bien structuré.
  - Renforcer l'apprentissage de l'algorithmique et du langage C.

### III. Organisation et planning.

- Le projet se conduit en équipe de trois ou quatre étudiants d'un même groupe de TD.
- Les étudiants doivent d'inscrire sur la plateforme projet la semaine du 03/11/2025.
- Le projet démarre la semaine du 10/11/2025 (distribution du sujet) et se termine le 19/12/2025 (rendu final déposé par chaque équipe).
- Un premier rendu devra être déposé la semaine du 24/11/2025. Il présentera l'état d'avancement du projet à mi-parcours.
- Des soutenances auront lieu la semaine du 05/01/2026.

03/11/2025	10/11/2025		24/11/2025		15/12/2025		05/01/2026
Constitution des équipes	Distribution du sujet		Rendu mi-parcours		Rendu final		Soutenances

## IV. Cahier des charges détaillé.

A son lancement, le programme doit afficher un menu proposant au moins les fonctionnalités suivantes :

1. Lire les règles du jeu
2. Commencer une nouvelle partie (à partir du 1<sup>er</sup> niveau)
3. Reprendre une partie (au dernier niveau enregistré)
4. Quitter

### 1. Fonctionnement du jeu de base.

#### Déroulement d'une partie

Au début d'une partie, un **nombre de vies** limité est attribué au joueur. Une partie comporte au moins **3 niveaux de difficultés croissantes**. Pour gagner la partie, il faut gagner chacun des niveaux. Pour gagner un niveau, il faut remplir un contrat en un temps limité. A chaque fois que le joueur perd un niveau, il perd une vie.

#### Déroulement d'un niveau

Chaque niveau se présente sous la forme d'un plateau de jeu représenté par un **tableau à deux dimensions** de 25 lignes x 45 colonnes. Au début du niveau, les cases du tableau sont remplies aléatoirement avec 5 types d'**items** (pouvant représenter des bonbons, des légumes ou autres ...).

Le joueur a pour objectif de remplir un **contrat** qui lui est annoncé au début du niveau. Un contrat fixe un **nombre d'items à éliminer** en un **nombre maximum de coups** et en un **temps limité**. Les contrats sont différents sur chaque niveau et de difficultés croissantes.

#### Exemple de contrat :

20 fraises, 50 oignons et 20 mandarines en 30 coups maximum et 1 minute.

**A vous d'imaginer des contrats bien adaptés pour garantir la jouabilité et la progression du jeu.**

### Elimination des items

**Certaines suites ou groupes d'items identiques** forment des figures particulières qui font disparaître les items.

- Une suite horizontale ou verticale de 6 items identiques d'un type X provoque la disparition de tous les items de type X présents sur le tableau, même s'ils sont isolés.

X	*	X	*	X	0	X	X
0	0	X	*	X	X	\$	#
X	\$	0	#	0	#	X	#
X	X	X	#	X	X	0	X
\$	0	0	0	0	0	0	*
X	X	#	X	X	*	X	X
0	#	X	X	*	*	#	*
X	X	0	0	X	\$	X	X

Elimine 15 items 0

- Une croix de 9 items d'un type X provoque la disparition de tous les items de type X présents sur la même ligne ou la même colonne, même s'ils sont isolés.

X	*	X	X	0	0	X	X
0	X	X	*	X	X	\$	#
X	\$	0	X	0	#	X	#
X	X	#	X	#	0	0	X
\$	X	X	X	X	X	0	X
X	X	#	X	#	*	X	X
0	#	X	X	*	*	#	*
X	X	0	0	X	\$	X	X

Elimine 11 items X

- Un carré formé de 4\*4 items identiques de type X provoque la disparition de tous les items de type X contenus dans le carré.

X	*	X	X	0	0	X	X
0	0	X	*	#	X	\$	#
0	X	X	X	X	#	X	#
X	X	X	\$	X	0	0	X
\$	X	0	X	X	\$	0	*
0	X	X	X	X	*	X	X
0	#	X	#	*	*	#	*
X	X	0	0	X	\$	X	X

Elimine 14 items X

- Une suite horizontale ou verticale de 4 items identiques d'un type X provoque la disparition des items de la suite.

**! Au début du niveau, aucune figure spéciale ne doit apparaître dans le tableau !**

### **Remplacement automatique des cases vides**

Lorsque des items sont éliminés, 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 tirés aléatoirement. Une fois les trous comblés, les règles de suppression s'appliquent de nouveau automatiquement et les trous sont de nouveau comblés ... jusqu'à stabilisation.

### **Sélection et déplacement des items par le joueur**

Le joueur peut se déplacer dans le tableau, sélectionner (ou désélectionner) un item, et choisir le voisin avec lequel il souhaite le permuter. Si grâce à la permutation une figure spéciale se forme, les règles de suppression s'appliquent.

**Chaque permutation compte pour un coup.**

### **Fin d'un niveau**

Echec : si le nombre de coups est atteint ou le temps écoulé et que le joueur n'a pas éliminé tous les items du contrat, le joueur perd une vie. S'il lui reste au moins une vie, il peut retenter le niveau.

Victoire : si le joueur remplit son contrat, il peut passer au niveau suivant.

Dans les deux cas, le programme propose aussi au joueur de quitter le programme en sauvegardant le dernier niveau gagné et le nombre de vies restantes.

### **Fin d'une partie**

Echec : si le joueur n'a plus de vie et n'a pas gagné tous les niveaux.

Victoire : le joueur a remporté tous les niveaux.

Dans les deux cas, le programme réaffiche le menu.

### **Sauvegarde et chargement du niveau atteint**

La sauvegarde des niveaux atteints par les joueurs se fait dans un fichier texte. Pour sauvegarder un niveau atteint, le joueur doit indiquer un pseudo non encore utilisé. Au lancement du programme, un joueur peut reprendre une partie au dernier niveau qu'il a atteint en saisissant son pseudo.

## **2. Contraintes techniques.**

- **Langage : C.**
- **Mode graphique : console Windows.**
- **Bibliothèques autorisées :**
  - Librairies standard du langage C dont <conio.h> pour la lecture non bloquante des saisies clavier
  - Module "affichage\_console.h" fourni en cours pour un affichage console évolué
- **Séparation du moteur de jeu et de l'IHM.**

Les affichages et saisies utilisateur doivent être séparés au maximum de la logique interne du jeu et des algorithmes de traitement. Idéalement, on devrait pouvoir changer l'IHM sans devoir modifier les sous-programmes de traitements des données et d'application des règles du jeu.

Conseil : En mémoire, les tableaux des différents niveaux du jeu peuvent être tout simplement représentés par des matrices d'entiers (un 0 pour une case vide, un numéro différent pour chaque type d'items). Ce n'est qu'au moment de l'affichage à l'écran, que chaque valeur sera « traduite » en un caractère ou une couleur particulière.

- **Programmation modulaire**

Le programme doit être décomposé en plusieurs modules avec au minimum :

- o Un **module affichage** qui regroupe tous les sous-programmes d'affichage : affichage du menu, affichage des tableaux et des informations sur la partie (contrat, nombre de vies ...)
- o Un **module** regroupant tous les sous-programmes de traitement des données du jeu : générer les tableaux, supprimer les items, tester la fin d'un niveau ...
- o Le fichier main.c avec le sous-programme principal.

### 3. IHM.

Le jeu se jouera intégralement en mode console.

#### Affichage :

**A vous de proposer un affichage agréable et efficace dans lequel le joueur retrouvera toutes les informations nécessaires au bon déroulement d'une partie.**

N'hésitez pas à mettre de la couleur, des bordures, à choisir des caractères originaux ou des couleurs bien différenciées pour représenter les items.

A chaque niveau doivent être affichés :

- Le numéro du niveau
- Le tableau de jeu
- Le nombre de vies restantes
- Le contrat à atteindre
- Le temps restant
- Le nombre de coups restants
- Le nombre d'items de chaque type restants à éliminer

### **Saisies utilisateur :**

Pendant une partie, les saisies du joueur doivent être gérées selon le principe de la programmation événementielle, de manière que le joueur n'ait pas à valider chaque saisie.

### **Sélection et déplacement des items par le joueur**

- Sur un tableau, le joueur peut déplacer le curseur à l'aide des touches *z,q,s,d* ou des flèches du clavier.
- Une fois positionné sur l'item à déplacer, le joueur doit appuyer sur la barre d'espace. L'item sélectionné est alors mis en évidence (par un changement de casse ou de couleur par exemple). Un second appui sur la barre d'espace permet d'annuler la sélection.
- Le joueur doit ensuite choisir l'item à permuter en utilisant de nouveau les touches du clavier pour indiquer la direction (haut, bas, gauche ou droite).

## **4. Extensions.**

**En plus du jeu de base, vous devez proposer au moins deux extensions au choix :**

- o Une nouvelle figure spéciale ayant un effet différent de ceux déjà proposés dans le jeu de base.
- o Un item « bonus » de votre choix (exemples : un item qui peut remplacer n'importe quel autre pour former une figure, un item qui assemblé en figures spéciales fait gagner une vie ...)
- o Un item « malus » de votre choix (exemples : un item qu'on ne peut pas assembler en figures et qui se

propage de case en case, un item qui assemblé en figures spéciales fait perdre une vie ...)

- o Des cases « murs » sans item et qui empêchent les items de tomber.

**! Ces extensions ne doivent pas apparaître au premier niveau qui doit respecter strictement le cahier des charges du jeu de base !**

Vous pouvez ensuite laisser libre court à votre imagination pour proposer des niveaux supplémentaires avec d'autres extensions originales et stimulantes.

## **V. Travail à réaliser, rendus et modalités d'évaluation**

Vous devez appliquer la démarche de conception et de développement expliquée et vue en cours. Après avoir lu attentivement et étudié le cahier des charges (n'hésitez pas à poser des questions sur le CDC et à demander des précisions sur certains aspects), vous devez réfléchir pour concevoir une solution avant de passer aux phases de codage et de tests.

**Dans le premier rendu à mi-parcours (fin de semaine du 24/11/2025) vous présenterez votre démarche de conception et la solution proposée pour le jeu de base :**

### 1. Analyse et conception générale

- Analyse chronologique descendante (ACD), de préférence sous forme de logigramme, montrant le fonctionnement global du programme.
- Tableau récapitulant les principales fonctionnalités avec pour chaque fonctionnalité, les données essentielles et les principaux traitements associés.
- Logigrammes détaillant le fonctionnement d'une partie et d'un niveau.

- Tableaux listant les principales données nécessaires au déroulement d'une partie et d'un niveau. Pour chaque donnée, préciser son type.
- Découpage modulaire

## 2. Analyse et conception détaillée

- Découpage en sous-programmes et répartition des sous-programmes dans les différents modules.
- Pour chaque sous-programme, préciser ses données en entrée, ses résultats en sortie, et indiquer son prototype commenté.
- Détailler les algorithmes les plus complexes sous-forme de logigrammes.
- Graphe d'appels montrant comment l'enchaînement des appels de sous-programmes permettra de réaliser les fonctionnalités du jeu.

## 3. Maquettes de l'interface graphique (mode console) : comment sera organisé l'affichage de l'écran de jeu.

## 4. Planning prévisionnel et répartition des tâches au sein de l'équipe.

Une fois les étapes de conception menées à bien, vous pourrez démarrer les phases de codage et de tests. Testez vos sous-programmes au fur et à mesure. Vérifiez régulièrement que les différentes parties développées par chaque membre de l'équipe s'intègrent et fonctionnent correctement ensemble pour former le programme final.

**Le rendu final (fin de semaine du 15/12/2025) contiendra le code du projet et un compte rendu dans lequel vous devrez :**

1. Donner le planning réellement suivi et la répartition réelle des tâches réalisées par chacun.

2. Indiquer les éventuelles modifications apportées à la solution proposée dans le rapport de mi-parcours.
3. Montrer le contenu de vos fichiers header.
4. Faire le graphe d'appels final du programme.
5. Montrer les tests réalisés et les résultats obtenus à l'aide de captures d'écran commentées.
6. Rédiger des bilans individuels et collectifs sur le déroulement du projet et le travail réalisé.
7. Préciser vos sources et la part d'aide (humaine ou artificielle) dont vous avez bénéficiée.

**Des soutenances auront lieu la semaine du 05/01/2026.**

**L'évaluation du projet tiendra compte de votre travail collectif et individuel.**

**Travail collectif :**

- Qualité des rendus et présentations.
- Pertinence et rigueur de la démarche de conception.
- Qualité du code (découpage modulaire et en sous-programmes, commentaires et indentations ...).
- Respect du cahier des charges
- Créativité et originalité de vos extensions dans la limite des libertés offertes par le cahier des charges.
- Ergonomie et jouabilité du jeu final.

**Travail individuel :**

- Implication dans le projet, importances et difficultés des tâches réalisées.
- Qualité des réponses individuelles fournies au jury lors des soutenances.

**Bon projet !**

Julienne Palasi, Jean-Pierre Segado, Baligh Mnassri et  
toute l'équipe enseignante.