

Projet POO Java avancé

La revanche de Snoopy

Introduction	2
Présentation du jeu original :	2
Présentation du projet : "La revanche de Snoopy"	5
Représentation d'un niveau	5
Déplacement de Snoopy	5
Mouvement de la balle	6
Gestion des objets	6
Condition de victoire ou de défaite	6
Gestion du temps	6
Gestion des scores	6
Sauvegarde et chargement d'une partie	7
Gestion des mots de passe	7
Mode "pause"	7
Cahier des charges	8
1) Jeu de base (15 points)	8
2) Jeu avancé - Déplacement automatique (5 points)	9
Versioning de votre projet : GIT	9
Planning et organisation du travail	9
Travail demandé, contraintes et consignes	10
1- Analyse et conception générale du diagramme de classes.	10
2- Analyse et conception détaillée	10
3- Développement dans le langage objet Java (codage, tests).	11
4- Critères de notation.	11
5- Deadlines des 2 livrables à déposer sur campus	11

Introduction

Snoopy's Magic Show est un jeu vidéo créé en 1990 qui met en scène le personnage de Snoopy. C'est un jeu de réflexion de type "puzzle game" où le but est de récupérer 4 oiseaux pour passer au niveau suivant... mais le chemin le long des niveaux est semé d'embûches...

Présentation du jeu original :

Créé sur Gameboy en noir et blanc, voici l'écran d'accueil :



Ecran d'accueil original sur GameBoy

Le but de Snoopy est de récupérer 4 oiseaux aux 4 coins du niveau en un temps imparti. Le problème est que ces 4 oiseaux ne sont pas si faciles à récupérer. Une balle rebondit constamment dans le niveau afin de freiner Snoopy dans sa quête. Mais ce n'est pas tout, d'autres pièges sont présents comme des téléporteurs que la balle peut emprunter ou des cases piégées, voir même des blocs à pousser ou à casser...

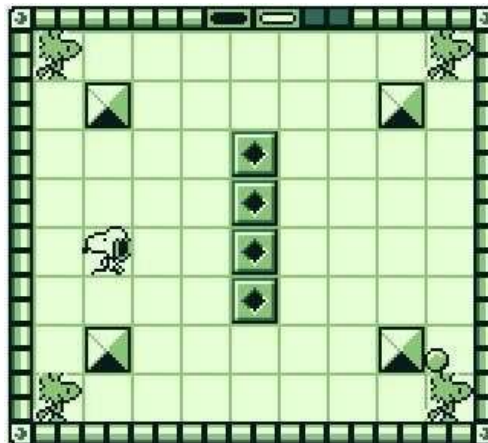
Le jeu original comporte 120 niveaux. Un mot de passe est disponible pour chaque niveau.

Un mode 2 joueurs est aussi disponible. Le principe est le même qu'en mode 1 joueur sauf que chaque joueur joue chacun son tour. On ne s'intéressera pas à cette fonctionnalité dans ce projet.

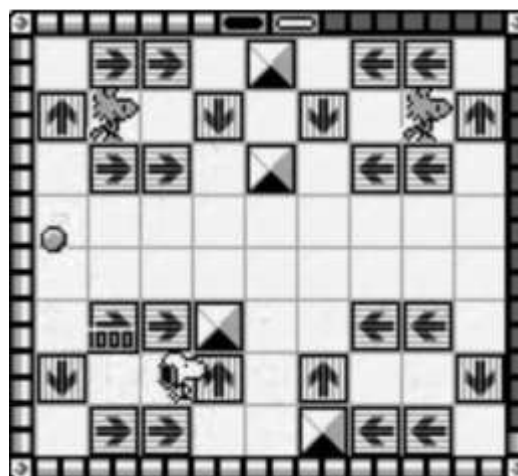
Voici quelques captures d'écran du jeu original :



*Les quatre oiseaux à sauver sont situés dans les quatre coins du niveau.
La balle se trouve dans la zone supérieure gauche du niveau. Dans ce
niveau, Snoopy doit pousser des blocs...*



*Dans ce niveau, Snoopy peut casser les blocs du milieu pour avoir des
objets (bonus ou malus)*



*Les blocs "flèches" sont des sortes de tapis roulant qui suivent la direction de
la flèche*

Le temps imparti est symbolisé par des rectangles qui entourent le niveau.



Illustration de la gestion du temps Chaque unité de temps représente une seconde

Parmi les objets disponibles dans le jeu original, on trouve :

- une horloge : objet pour figer le temps (et donc aussi la balle mais pas Snoopy)
- Des objets symbolisant l'invincibilité
- des blocs qu'on peut pousser
- des blocs qu'on peut casser
- des blocs qui disparaissent et réapparaissent à intervalle de temps régulier
- des flèches "tapis roulant"

Les ennemis du jeu original sont les suivants :

- Les fameuses balles qui rebondissent dans le niveau (il peut y en avoir 2 maximum)
- un méchant Snoppy qui a la caractéristique de suivre le vrai Snoopy :)

Présentation du projet : "La revanche de Snoopy"

Le jeu proposé cette année s'inspire fortement de son ancêtre présenté dans le paragraphe précédent **en mode graphique** ! Pour simplifier, le jeu comportera **au MINIMUM 3 niveaux de difficulté croissante**, une gestion des scores et des mots de passe par niveau et la possibilité de charger une partie sauvegardée.

Une fois lancé, le jeu proposera un menu classique permettant de réaliser les actions suivantes :

1. Jeu de base – déplacement manuel (un seul joueur)
2. Jeu avancé – déplacement automatique (ordinateur)
3. Charger une partie
4. Mot de passe
5. Scores
6. Quitter

Initialement, le joueur possède 3 vies.

Chaque niveau devra être résolu en moins de 60 secondes. Si le temps est écoulé, le joueur perd une vie et recommence le niveau. Le but est de récupérer les 4 oiseaux du niveau sans se faire toucher par la balle (ou les balles) et les ennemis (si présents).

Représentation d'un niveau

Chaque niveau sera représenté par une matrice rectangulaire de caractères de 10 lignes par 20 colonnes, contenant les objets suivants :

- Snoopy (le personnage)
- La balle (ou les balles)
- Les oiseaux à récupérer
- Les blocs poussables
- Les blocs cassables
- Les blocs piégés (nouveau !!)

Déplacement de Snoopy

Snoopy ne peut pas se déplacer en diagonale. Il ne peut se déplacer que dans les 4 directions classiques (Haut, Bas, Gauche et Droite) et d'une seule case à la fois. Evidemment, en cas d'obstacle, Snoopy ne pourra pas effectuer son déplacement. Il ne peut pas sortir du niveau.

Mouvement de la balle

La balle se déplace exclusivement en diagonale et rebondit sur les murs (les bords de la matrice). La vitesse de la balle est fixe. Si le niveau présente plusieurs balles, on ne gèrera pas la collision entre les balles.

Gestion des objets

Un bloc poussable ne peut être poussé **qu'une seule fois** dans une direction précise mais ne peut pas sortir du niveau ! Ce type de bloc n'est pas traversable.

Un bloc cassable ne peut pas être traversé. Pour le casser, il faut appuyer sur une touche spéciale. Il ne libère aucun item.

Un bloc piégé tue instantanément si on le touche.

Condition de victoire ou de défaite

Pour gagner, il faut récupérer les 4 oiseaux du niveau. Un fois un niveau terminé, on charge automatiquement le niveau suivant et ainsi de suite.

Quand le joueur perd toutes ses vies, on affiche un écran de GameOver et le jeu revient au menu principal.

Gestion du temps

Chaque niveau aura un timer initialisé à 60 secondes. Quand le timer atteint 0, le joueur perd une vie. Pour simplifier, le timer sera représenté par un simple affichage (compte à rebours).

Gestion des scores

La gestion des scores s'effectue de cette manière pour chaque niveau :

$$S_{niveau} = temps\ restant * 100$$

Au fur et à mesure des niveaux, les scores s'additionnent pour former le score final.

Exemple : Le niveau 1 est fini en 30 secondes, le niveau 2 en 55 secondes et le dernier niveau en 59 secondes. Le score du joueur à la fin du niveau 3 sera :

$$S_{total} = S_{niveau1} + S_{niveau2} + S_{niveau3} = 30 * 100 + 5 * 100 + 1 * 100 = 3600\ points$$

Sauvegarde et chargement d'une partie

A chaque instant, le joueur peut s'il le souhaite sauvegarder sa partie en appuyant sur la touche 's' du clavier. Dès qu'il le fait, le programme lui demande le nom du fichier de sauvegarde puis retourne sur le menu principal (la partie en cours est donc quittée).

La sauvegarde se fera au choix soit dans un fichier texte ou un fichier binaire et comprend les éléments suivants :

- La position de Snoopy
- La position de la balle (ou des balles)
- La position de tous les éléments du décor (blocs, oiseaux, ...) restant
- Le temps restant
- Le nombre de vies
- Le score courant

Pour charger une partie, il faut passer par le menu principal et choisir "Charger une partie". Le joueur est ensuite invité à entrer le nom de son fichier de sauvegarde.

Gestion des mots de passe

Chaque niveau sera accessible par un mot de passe unique. Un joueur peut donc s'il connaît le mot de passe accéder au niveau de son choix à partir du menu principal. Ce mode devrait être très utile en soutenance pour montrer rapidement vos fonctionnalités...

Mode "pause"

Si le joueur appuie sur une touche de « pause », le jeu se met en pause, c'est à dire :

- la(es) balle(s) se fige(nt) (et les ennemis si présents)
- Snoopy ne peut plus se déplacer
- le timer s'arrête

Pour enlever la pause, il suffit d'appuyer à nouveau sur la touche de « pause ».











Cahier des charges

1) Jeu de base (15 points)

Votre jeu commence par le menu.

Implémenter le jeu de base en suivant les instructions suivantes :

- Pour chaque niveau de jeu, le plateau de jeu sera représenté par une matrice 2D en mémoire et un chronomètre défile.
- Les plateaux de jeu seront stockés dans des fichiers texte, chaque élément graphique du décor étant identifié par un chiffre :

Identifiant (explication)	Élément du décor graphique
0 (case vide)	
1 (bloc cassable)	
2 (bloc poussable)	
3 (bloc piégé)	
4 (bloc invincible)	
5 (bloc disparition/apparition)	
6 (bloc de tapis roulant)	
7 (balle)	
8 (Snoopy)	
9 (oiseau)	

- Chaque identifiant d'élément sera représenté par une icône (image) à l'écran.
- Le déplacement du personnage se fera case par case (gérer les collisions) manuellement avec des touches du clavier (indiquez ces touches à l'écran pour guider l'utilisateur). Le joueur peut mettre le jeu en « pause » ou annuler cette « pause » à tout moment.
- L'utilisateur peut revenir au menu et sauvegarder son niveau de jeu en cours : le niveau, le chronomètre et le plateau avec les identifiants associés aux éléments du décor. Il peut accéder à un niveau directement par un mot de passe.
- Quand on termine un niveau (c'est-à-dire que Snoopy a récupéré les 4 oiseaux), un score est établi en fonction du temps mis et du niveau, puis on charge le niveau suivant. Les scores sont additionnés de niveau en niveau.

Pour vous aider, voici une liste possible de méthodes pour gérer le jeu de base :

- Une méthode qui charge un niveau à partir d'un fichier texte
- Une méthode qui vérifie si le niveau est résolu
- Une méthode qui déplace Snoopy, en tenant compte du type de bloc
- Une méthode qui dessine à l'écran le plateau de jeu

Et n'oubliez pas de tester que vos niveaux sont jouables à l'aide du clavier...

2) Jeu avancé - Déplacement automatique (5 points)

1. Implémenter un algorithme naïf permettant de résoudre un niveau par force brute. Pour cela, on testera tous les déplacements du personnage, en tenant compte des blocs (exemple : pousser un bloc poussable, un bloc piégé tue Snoopy etc.) et des balles, jusqu'à trouver une séquence qui résout le niveau (les 4 oiseaux récupérés).

Nota. L'ordre de visite des successeurs se fera exclusivement dans cet ordre : HAUT/BAS/GAUCHE puis DROITE.

2. Puis implémenter une méthode (ou plusieurs) permettant de visualiser votre solution (le personnage se déplace, en tenant compte des blocs et des balles).
3. Implémenter un algorithme permettant de résoudre un niveau avec un plus court chemin BFS « *Best First Search* » (parcourt en largeur) puis un DFS « *Depth First Search* », (parcourt en profondeur).

Versioning de votre projet : GIT

Vous utiliserez l'outil de **versioning GIT** pour partager votre code de ce projet : voir page campus du cours « Versioning » : [Versioning avec Git](#).

Lors de la soutenance, vous devrez montrer les différentes phases de développement de votre projet (les branches des "versions" du projet). Vous devez aussi être en mesure de montrer "qui a fait quoi" dans le projet.

Grâce au versioning, vous n'aurez plus de problèmes et d'excuses du type :

- c'est mon camarade qui a tout le projet, je n'ai pas pu corriger les erreurs...
- mon disque dur est mort la veille de la soutenance...
- on m'a volé mon ordinateur le jour de la soutenance...
- j'ai renversé du liquide sur mon clavier...
- je ne comprends pas, mon code a été écrasé ? et je n'ai pas fait de backups...
- ...

Planning et organisation du travail

Période de réalisation du projet :

- **Groupe de TD 4 et 5** : de la semaine du 19 septembre à la semaine du 7 novembre 2022 incluse. Des soutenances auront lieu la semaine du 14 novembre 2022.
- **Groupe de TD 1, 2 et 3** : de la semaine du 10 octobre à la semaine du 5 décembre 2022 incluse. Des soutenances auront lieu la semaine du 12 décembre 2022

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

Évaluation : La première version à présenter à votre chargé de TP donnera lieu à une note de suivi et la version finale sera présentée à la soutenance du projet qui donnera lieu à la note de projet.

Un diagramme de classes doit être présenté pour montrer la conception objet, y compris avec héritage, en respect du **travail demandé, contraintes et consignes ci-dessous**.

Travail demandé, contraintes et consignes

1- Analyse et conception générale du diagramme de classes.

- A partir du cahier des charges (CDC) : extraire les données pertinentes, les regrouper en grandes entités / objets, spécifier et caractériser les attributs et les fonctionnalités de chaque objet, identifier les interactions entre les différents objets ainsi que les différents scénarios possibles, ...
- En déduire le **diagramme de classes**, mettant en relation les classes en y intégrant si possible de **l'héritage et du polymorphisme**. Pour chaque classe, définir les attributs (en général *private*, ou *protected* en cas d'héritage), et les méthodes (inutile d'y mentionner, les constructeurs, les getters et les setters).
- IHM : lister les choix à offrir au démarrage, lister les événements à gérer, déterminer l'organisation et le contenu de l'écran de jeu (maquette), ...
- Rédiger une Analyse Chronologique Descendante (ACD) du programme principal.

Définir une organisation modulaire multi-fichiers respectant l'approche **Modèle-Vue-Contrôleur** : [Modèle Vue Contrôleur](#) (wikipedia) et [Adopter une architecture MVC](#) (openclassrooms). Votre diagramme doit montrer ce découpage modulaire : encadrer avec 3 couleurs différentes les classes faisant partie du Modèle (couleur 1), de la Vue (couleur 2) ou du Contrôleur (couleur 3)

- Répartir les tâches au sein de l'équipe.

2- Analyse et conception détaillée

- Pour chaque classe, lister les prototypes de toutes les méthodes requises 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 objet Java (codage, tests).

Dans le langage objet **Java**, implémenter le jeu en respect de votre analyse des 2 étapes précédentes : **diagramme de de classes**, organisation modulaire multi-fichiers selon l'approche **Modèle-Vue-Contrôleur**, avec **héritage et polymorphisme**, commenter les prototypes des méthodes (fonctionnalités) en précisant les paramètres d'entrée/sortie et commenter aussi l'ACD de ces méthodes et du programme principal.

4- Critères de notation.

Vous devez réaliser l'ensemble du cahier des charges de base (expliqué dans la paragraphe précédent). Le langage de programmation doit être le langage objet Java avec héritage... Votre code devra être modulaire, respecter les interfaces en mode graphique 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 Campus...

Votre travail sera jugé sur les critères suivants :

- Le respect rigoureux des règles du jeu énoncé précédemment (CDC)
- 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.

5- Deadlines des 2 livrables à déposer sur campus.

Les dates et toutes les consignes sur les 2 livrables (les 2 versions) à déposer sont spécifiées dans la section [Cours : POO Java \(app\), Section : Projet SNOOPY pour les étudiants de niveau avancé en Java ... \(omneseducation.com\)](#) :

- Une première version avec le **PowerPoint et le code du jeu de base en partie ou totalement fonctionnel** : sur [Version 1 du livrable \(à présenter à votre chargé de TP\)](#). Elle sera à montrer à votre chargé de TP à mi-parcours.
- La version finale du **code incluant le jeu de base totalement fonctionnel et avancé** sur [Version finale du livrable avec jeu avancé \(version soutenance\)](#). La réalisation et son bon fonctionnement sera présenté lors d'une soutenance.