Mini-Projet POO Java
La revanche de Snoopy

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 reflexion 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 :

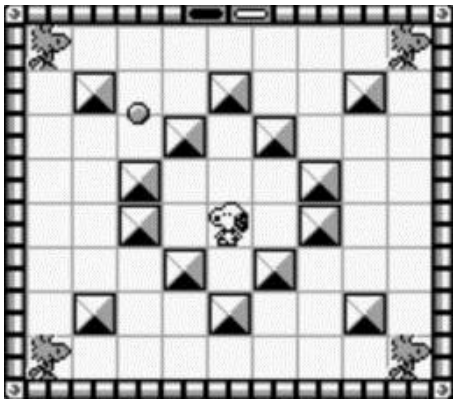
Ecran d'accueil original sur GameBoy

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...

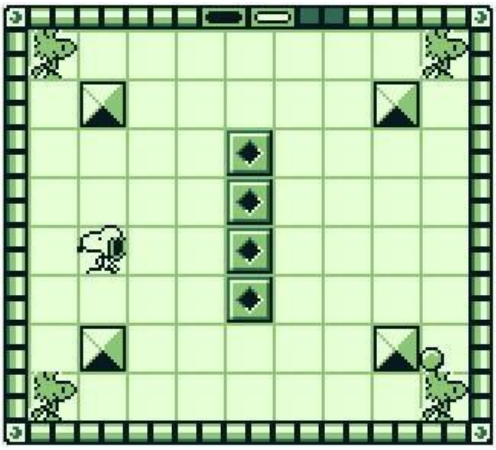
Le jeu original comporte 120 niveaux. Un mot de passe est disponible pour chaque niveau.

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.

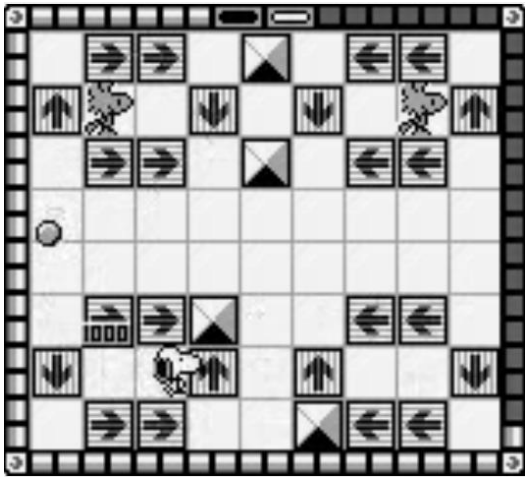
Voici quelques captures d'écran du jeu original :



Les quatre oiseaux a 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 comme malus)



Les blocs "flèches" sont des sortes de tapis roulant
qui suivre 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)
- un "P" : objet 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 sauf qu'il sera **en mode console** ! 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. Jouer (un seul joueur)
2. Charger une partie
3. Mot de passe
4. Scores
5. 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 ennemies (si présents).

Représentation d'un niveau

Chaque niveau sera représenté par une matrice rectangulaire de caractères de 10 lignes par 20 colonnes :

- Snoopy (le personnage) sera représenté par la lettre 'S'
- la balle (ou les balles) sera représentée par la lettre 'B'
- les oiseaux à récupérer seront représentés par la lettre 'O'
- les blocs poussables seront représentés par la lettre 'P'
- les blocs cassables seront représentés par la lettre 'C'
- les blocs piégés (nouveauté !!) seront représentés par la lettre 'T'

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 gérera 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 la touche 'a'. 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 lettres 'O' 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'additionnnent 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 :

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

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 principale 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 connait 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 "p", le jeu se met en pause, c'est à dire :

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

Pour enlever la pause, il suffit d'appuyer à nouveau sur la touche "p".

Planning et organisation du travail

Période de réalisation du mini-projet : 4 semaines de début mars à fin mars 2018.
Équipes : en trinôme ou éventuellement en binôme dans un même groupe de TD.
Evaluation : évaluation en continu (dans le cadre des « notes de suivi ») du travail et de l'organisation de l'équipe, de l'avancement et de la qualité de l'analyse et de la production.

Une soutenance aura lieu en TP la semaine du 2 avril 2018 : 10 minutes de présentation par équipe. Nous vous repréciserons les modalités de cette soutenance.

Un diagramme de classes doit être présenté pour montrer la conception objet, y compris avec héritage, en respect des consignes de l'analyse ci-dessous.

Les grandes étapes à respecter

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 évènements à gérer, déterminer l'organisation et le contenu de l'écran de jeu (maquette), …

Rédiger une Analyse Chronologique Descendante du programme principal.

Définir une organisation modulaire muti-fichiers respectant l'approche **Modèle-Vue-Contrôleur**. 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 un langage objet (codage, tests).

Dans le langage objet Java, implémenter le jeu en respect du CDC et de votre analyse des 2 étapes précédentes : diagramme de de classes, organisation modulaire multi-fichiers selon l'approche Modèle-Vue-Contrôleur, si possible 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.

Versionning de votre projet : GIT

Utilisez l'outil de versionning qui vous a été enseigné : voir page campus du cours « Versioning » http://campus.ece.fr/course/view.php?id=1893. .

Vous utiliserez GIT pour gérer ce mini-projet. Une formation vous a été dispensée pour apprendre à utiliser ces outils.

Vous pourrez ainsi être en mesure de montrer "qui a fait quoi" dans le projet.

Grâce au versionning, vous n'aurez plus de problèmes 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...
- ...

Et donc plus d'excuses le jour de la soutenance :)

Travail demandé, contraintes et consignes

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 console et bien commenté !

Cette partie sera notée sur 15 points.

Les extras seront notés sur 5 points. Vous pouvez par exemple faire :

- L'utilisation correcte du versionning
- gérer les objets "horloge" et "invincibilité" du jeu original
- ajouter de nouveaux objets
- ajouter des bonus / malus
- une IA permettant de jouer à la place du joueur (et donc de gagner tout le temps)
- et toutes les idées que vous aurez... y compris en mode graphique. Laisser parler votre imagination :)

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.

Le rendu se fera exclusivement via Campus avant la date limite (la deadline sera écrite sur la page Campus). Tout retard sera pénalisé par **2 points** par jour de retard.