PROJET C 1A

2015

Candy Crush Saga

Adélaïde ARMAND

Sarah EXBRAYAT

Elisabeth MOUNIER

I. Présentation du projet

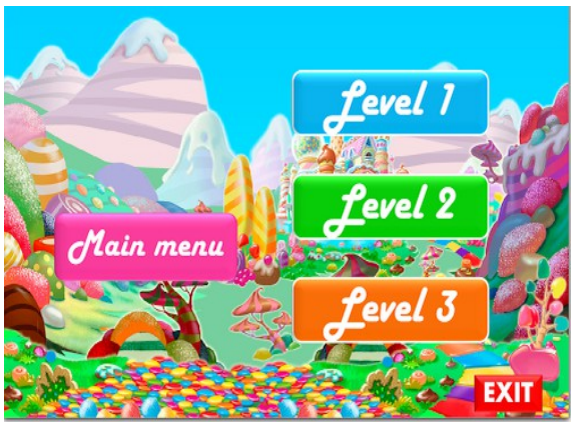
Nous avons choisi de coder le jeu Candy Crush Saga.

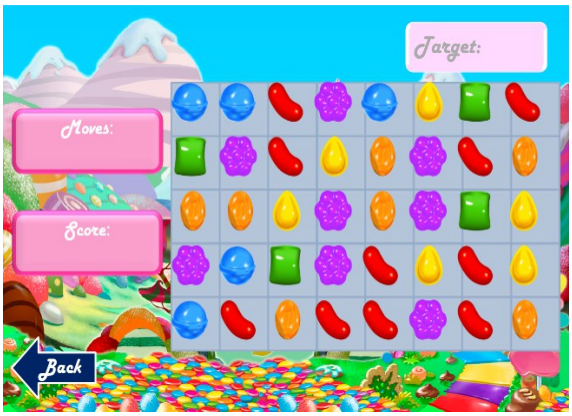
Le but du jeu est d'échanger 2 bonbons de manière à former, au minimum, des lignes ou colonnes de 3 bonbons identiques. Ces bonbons explosent et on marque ainsi des points.

Chaque niveau a ses particularités. Dans notre projet nous avons décidé :

- Niveau 1 : Score de 2000 à atteindre en 15 déplacements

→Nous avions prévu deux niveaux supplémentaires que nous n'avons pas eu le temps de mettre en place.





II. Sources

Nous avons utilisé des images dont nous n'avons pas les droits.

Tutos utilisés pour la SDL :

- http://lazyfoo.net/tutorials/SDL/index.php
- http://alexandre-laurent.developpez.com/tutoriels/sdl-2/creer-premieres-fenetres/

Pour l'algorithme :

- http://openclassrooms.com/courses/apprenez-a-programmer-en-c

III. Remerciements

Nous remercions particulièrement Florian Rebaudo et Théo Ropiteaux, tout deux élèves en deuxième année à Télécom Nancy pour leur aide précieuse et leurs conseils bienveillants.

IV. Mise en place du projet

Mise en place :

Cette partie a été travaillée ensemble.
Il s'agit de la définition des règles du jeu, des choix d'affichage, de la répartition du travail et des objectifs de travail pour chaque partie.
Globalement, ça s'est bien passé à part un problème de répartition du travail au début : Adélaïde et Sarah ont fait le même travail au début (structure des bonbons) mais s'en rendant vite compte il n'y a plus eu de problèmes par la suite.
Voici donc comment s'est réparti notre travail finalement :

Répartition du travail :

Nous avons divisé notre travail en 3 grandes parties, chacune était responsable d'une partie.

Partie I : Mise en place de l'affichage, gestion des menus ; puis gestion de l'affichage de la grille.
Cette partie a été réalisée par Elisabeth MOUNIER
Temps de travail : entre 35 et 40 heures

Partie II : Fonctions du jeux à proprement dites, code permettant de jouer au jeu sur le terminal avec toutes les fonctions que cela implique.
Cette partie a été réalisée par Adélaïde ARMAND
Temps de travail : + de 35 heures

Partie III : Gestion de l'affichage des bonbons et affichage du jeu en lien avec la partie II.
Cette partie a été réalisée par Sarah EXBRAYAT
Temps de travail : entre 35 et 40 heures

V. Description du travail

PARTIE I.

Le but de cette partie était de gérer l'affichage des menus à l'aide de la SDL afin qu'il ne reste plus qu'à confronter la SDL avec l'algorithme réalisé comme expliqué dans la partie suivante.

Mon travail se divise en 4 fichiers sources :

- main.c : où j'ai initialisé la SDL et créé une fenêtre, puis Sarah l'a complété en faisant la boucle de jeu.

- display.c (et display.h) : contient toute la gestion des menus

- events.c (et events.h) : gère tout les évenements comme la gestion du clic de la souris

- grid.c (et grid.h) : permet d'initialiser la grille en affichant les bonbons aléatoirement

Remarques sur le travail réalisé :

Il m'a fallu beaucoup de temps avant de me familiariser avec la SDL. Mais à l'aide de plusieurs tutos j'ai finalement réussi à charger une image, gérer les événements de la souris et enfin afficher des boutons qui me redirigeaient vers le menu suivant.

Cependant, une fois le menu 2 obtenu je me suis rendue compte que mon code allait se répéter de nombreuses fois car nous avons plusieurs menus (menu principal, menu des niveaux, menu niveau 1, etc). Florian Rebaudo (2A apprentis) m'a conseillé sur la démarche à suivre : c'est-à-dire en faisait une structure Button, une structure Layout, ainsi qu'une énumération des menus. Grâce à ces structures, il est alors plus simple de créer des fonctions (comme add_button_to_menu) qui nous permettent d'indiquer dans quel layout nous nous trouvons, les coordonnées où nous voulons le boutons, le menu dans lequel nous nous trouvons, ainsi que l'image à charger.

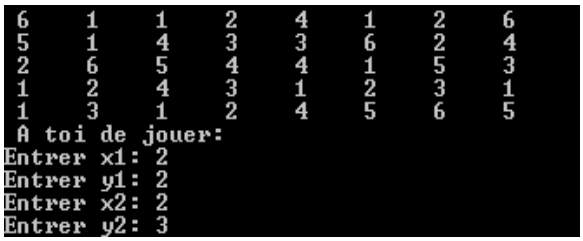
Une fois ce problème résolu, la suite était plus simple, à part quelques erreurs, en particulier seg fault (en fait mes images n'étaient pas de vraies images BMP).

Puis pour un problème d'esthétique, j'ai installé la librairie png, pour utiliser la fonction IMG_LOAD, ce qui enlève le fond blanc des boutons.

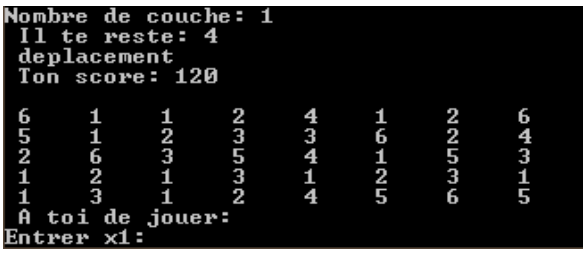
Ensuite, en ce qui concerne l'initialisation de la grille aléatoirement, au début nous comptions prendre l'image avec tout les bonbons et les découper grâce à la SDL. Mais je n'y arrivais pas, donc j'ai choisi une autre méthode : après avoir découper les bonbons « à la main » sur GIMP, j'ai fait un tableau de surface (dans le main.c) permettant de charger les images de chaque bonbon. Puis après avoir initialisé la grille, j'ai procédé à l'affichage avec draw_candy.

PARTIE II.

Le but de cette partie était de coder le jeu de manière à pouvoir y jouer sur le terminal.
Mon travail final se divise en 3 fichiers sources :
-game.c qui contient toutes les fonctions qui permettent de jouer.
-game.h qui comporte la signature de toutes les fonctions
-level.c qui permet de jouer au premier niveau (initialisation du score à atteindre, du nombre de coups par partie autorisée).



Voici une capture d'écran de la version du jeu avec laquelle on peut jouer sur le terminal, ici je vais échanger un 5 et un 4



Voici la grille affichée (Grid) après un coup, j'ai bien formé une ligne de 3 bonbons identiques qui ont « explosé ».

Remarques sur le travail réalisé :

J'ai pu coder de manière plutôt fluide après une longue étape de définition du problème sur feuille. C'est à ce moment que j'ai décidé d'utiliser une fonction temporaire tempt_grid qui me permet de faire des modifications au fur et à mesure de l'analyse de la grille sans perdre de données. J'ai décidé de diviser mon travail en un maximum de fonction pour pouvoir les tester au fur et à mesure.

C'est cette étape qui a été la plus longue avec les tests de chaque fonction.

Problèmes rencontrés lors de l'implémentation des fonctions :
- gestion des indices i et j, se familiariser avec les coordonnées de mes tableaux à 2 dimensions

Ensuite est venu la longue étape des tests : j'ai testé chaque fonction une par une en essayant de couvrir le plus de cas possibles, car si tout semblait logique en théorie, il m'a été difficile d'utiliser les bons indices (boucles for etc.) et donc les tests me permettait de savoir assez vite si je parcourais correctement ma grille etc.

J'ai essayé de commenter un maximum mon code pour qu'il soit clair.

PARTIE III.

Le but de cette partie était de gérer l'affichage des bonbons dans la grille de jeu et de gérer le jeu à partir de l'algorithme de la partie II.

Avant tout, j'ai créé les structures candy et grid dans le fichier structure.c et j'avais commencé à écrire un bout de l'algorithme avant que l'on choisisse de reprendre le travail d'Adélaïde.
Ensuite, j'ai commencé à travailler pour l'affichage. Les fichiers sources sont :

- main.c : Elisabeth a initialisé la SDL et créé une fenêtre, et je l'ai complété en faisant la boucle de jeu.

- projet.c que nous avons décidé d'abandonner.

Remarques sur le travail réalisé :

J'ai réussi assez facilement à définir les structure candy et grid ainsi que le début de l'algorithme. Mais j'ai ensuite mis du temps à me familiariser avec la SDL pour l'affichage. Elisabeth, qui était plus avancé que moi sur le sujet m'a aidé pour l'affichage des bonbons.

Ensuite, j'ai travaillé à coupler la partie algorithmique avec la partie SDL dans main.c.
Théo Ropiteaux m'a conseillé sur la manière de procéder.
Problème rencontré : avec la fonction PollEvent, le jeu était trop sensible, j'ai donc utilisé WaitEvent pour récupérer un seul clic de souris.

Enfin, j'ai utilisé l'algorithme fait par Adélaïde en faisant une translation puis une mise à l'échelle dans la grille de jeu.
Problème rencontré : au début, on rafraîchissait l'image beaucoup trop souvent : à chaque tour de boucle, et le jeu freeze. Désormais on la rafraîchit à chaque changement.