Université Libre de Bruxelles

INFO-F202

Langages de programmation 2

Projet : Candy Crush

Nieto Navarrete Matias
QUERINJEAN Arnaud
B-INFO

Janvier 2022

Contents

1 Introduction

Pour notre projet d'année du cours INFO-F202 (Langages de programmation 2), nous avons réalisé une implémentation en C++ du jeu Candy Crush Saga (avec quelques différences) avec une interface graphique. Le but du jeu est de faire écraser des bonbons colorés en associant des combinaisons d'au moins trois bonbons, afin de remplir des objectifs.

1.1 Outils et langages utilisés

Pour le projet nous avons utilisé le langage C++ ("C++2a"). Pour ce qui est de l'interface graphique, nous avons utilisé FLTK comme bibliothèque. Nous utilisions Github pour stocker et partager notre code.

1.2 Problématique

Pour la réalisation du projet nous avons dû faire face à la création d'un jeu en 2D pour permettre un visuel à l'utilisateur mais aussi notre capacité à créer un programme volumineux. Nous avons utilisé des classes, commentaires et la documentation pour un code lisible et structuré.

2 Tâches réalisées

Nous avons réalisé 11 tâches :

1. Fonctionnalité de base.
2. Animation des objets en cours de suppression.[1]
3. Murs.
4. Faire glisser le carré.
5. Impossible.[2]
6. Bonbons spéciaux.
7. Score.
8. Meilleur score.
9. Écran d'accueil.
10. Niveaux et sélection de niveau.
11. Objectifs.[3]

[1] Seul la suppression des bonbons sélectionné est animé.

[2] Pour détecter qu'il n'y a plus de mouvement possible l'utilisateur dois presser la touche A.

[3] Vue que nous avons pas fait le glaçage. Nous avons choisis un objectif différent qui est d'atteindre un score et si se score est atteint on passe au niveau suivant.

3 Implémentation

3.1 Classe cell

La classe Cell permet de dessiner un bonbon dans une boxe à partir d'une position x et y et une couleur.

- `Cell(Point center, int w, int h, int fruit)`

  c'est le constructeur, il appelle la classe `cree_bonbon()`.

- `void cree_bombon()`

  Permet de dessiner une image dans une des box du plateau de jeux.

- `void toucher(int fruit,bool s)`

  Permet de créer une petite animation quand l'utilisateur passe sa souris sur un bonbon d'une boxe.

- `void explosion()`

  Crée une animation d'explosion sur le plateau quand trois bonbons où plus sont aligné.

- `void deplace(Point d)`

  Re-dessine le bonbon a l'emplacement qu'il faut.

- `void mouseMove(Point mouseLoc)`

  Permet d'appeler l'animation du toucher d'un bonbon quand la souris passe sur un bonbon.

- `void mouseClick(Point mouseLoc)`

  Permet de vérifier si l'utilisateur a cliqué sur un bonbon.

- `int get_color()`

  Renvoi l'entier qui correspond a la couleur du bonbon.

- `void set_color(int new_fruit`

  Permet de modifier l'entier d'un bonbon par un autre entier. Ce qui permet de changer la couleur d'un bonbon.

- `void set_center(Point new_center)`

  Permet de modifier les coordonnées x et y d'un bonbon.

- `Point get_center()`

  Renvoi les coordonnées x et y du bonbon.

- `bool est_sélectionner()`

  Renvoi vrai si le bonbon est sélectionné où faux s'il est pas.

- `void désélectionner()`

  Permet de désélectionné un bonbon.

3.2 Classe Score

La classe Score permet de définir un score et de le modifier a chaque mouvements.

- `void score()`

  Augmente l'attribue scores de 10.

- `int get_score()`

  Renvoie la valeur de l'attribue scores.

- `int lecture_fichier_score()`

  Renvoie la valeur qui est écrit dans le fichier meilleur score.

- `void écrire_fichier_score()`

  Écrit le meilleur score de la partie dans le fichier meilleur score.

- `void reset_score()`

  Permet de remettre à 0 le meilleur score. En écrivant 0 dans le fichier meilleur score.

- `void moins_score()`

  Diminue l'attribue scores de 10.

- `void score_egale_0()`

  Remet l'attribue score à 0.

3.3 Classe Effacer_bonbon

La classe Effacer_bonbon permet d'effacer les bonbons qui sont aligné et prend en compte les pouvoir des bonbons spéciaux. Et appelle la classe Score qui va permettre de modifier le score après chaque effacement de bonbon.

- `bool a_coter(int x0, int y0, int x1, int y1)`

  Renvoi vrai si les deux bonbons sont l'un a coté de l'autre.

- `bool a_coter_diagonal(int x0, int y0, int x1, int y1)`

  Renvoi vrai si les deux sont l'un a coté de l'autre en diagonal.

- `void super_bonbon(int x1, int y1, int x0, int y0)`

  Permet de crée le pouvoir d'un bonbon multi-color.

- `void pouvoir_rayer(int sauve)`

  Permet de crée le pouvoir d'un bonbon rayé.

- `void pouvoir_emballer(int x, int y)`

  Permet de crée le pouvoir d'un bonbon emballé.

- `void condition_super_bonbon(int i, int j, int x, int y, int sauve)`

  Condition qui permet d'enregistrer les couleur des bonbons.

- `void efface()`

  Appel les methodes `efface_vertical_horizontal(..)` qui permet d'effacer les alignements.

- `int couleur_special(vector<Cell > position, int couleur_new_bonbon )`

  Renvoi la couleur du bonbon qui dois être crée par un alignement supérieur à trois.

- `void couleur_bonbon_emballer()`

  Permet de crée un bonbon emballée.

- `void efface_vertical_horizontal(vector<Cell > position,int couleur_new_bonbon, st`

  Efface les alignements possible.

3.4 Classe Mouvement

La classe Mouvement permet de déplacer deux bonbon si le déplacement mène à un un alignement et appel la classe Effacer_bonbon pour supprimer les alignements.

- `void print()`

  Permet d'afficher le plateau de jeu sur le terminal. Le plateau contient des nombres entiers qui représentent la couleur des bonbons.

- `void change_2_bombon(int x0, int y0, int x1, int y1)`

  Change la position x et y d'un bonbon avec un autre sur le plateau jeu du terminal.

- `void echange_bonbon(vector<Cell *> &liste)`

  Échange les bonbons qui sont sélectionnés dans le plateau jeu.

- `void deplacement(Cell *c)`

  Déplace si possible les bonbons sélectionnés par l'utilisateur.

- `bool aligner_horizontal(Cell horizontal)`

  Renvoi vrai si des bonbons sont alignés de façons horizontal.

- `bool aligner_vertical(Cell vertical)`

  Renvoi vrai si des bonbons sont alignés de façons vertical.

- `bool aligner(Cell c)`

  Renvoi vrai si la methode `aligner_vertical(..)` ou `aligner_horizontal(..)` renvoi vrai.

- `void animation()`

  Permet d'afficher une image d'exposition après avoir fait un mouvement.

- `void tomber_fruits()`

  Permet de faire tomber les bonbons qui se trouvent au dessus de l'alignement.

- `bool verifie_pas_0()`

  Renvoi vrai si il y a toujours des cases sans bonbons sur le plateau de jeu.

- `bool verif_mur(int x0,int y0,int x1,int y1)`

  Renvoi vrai si un des bonbons sélectionnés est un mur.

- `void efface_alligner_aleatoire()`

  Ver fie s'il y a des alignements et les supprimes.

- `void reset_plateau()` Remplace tout les bonbons du plateau par d'autres bonbons.

- `bool mouvement_verif(int a, int b, int c, int d)`

  Renvoi vrai s'il y a plus de mouvement possible sur le plateau.

3.5 Classe Canvas

La classe Canvas permet d'afficher un plateau de jeu qui correspond au niveau demandé. Et appelle la classe Mouvement qui va permettre de faire le déplacement de bonbon sur le plateau de jeu.

- `Canvas ()`

  C'est le constructeur, il permet de crée d'afficher un plateau de jeu sur la fenêtre.

- `void lire_fichier(const char* fichier_input)`

  Permet de lire le fichier pour créer le plateau du niveau demandé.

- `void cree_plateau()`

  Initialise un plateau de jeux 9x9 en appelant la classe Cell. Le plateau qui est initialisé est vide.

- `bool debut_check_horizontal()`

  Renvoi vrai si il y a des alignement horizontal de bonbon au lancement du niveau un car il est lancé avec des bonbons aléatoires.

- `bool debut_check_vertical()`

  Renvoi vrai si il y a des alignement vertical de bonbon au lancement du niveau un car il est lancé avec des bonbons aléatoires.

- `void mouseMove(Point mouseLoc)`

  Permet d'appeler la méthode mouseMove(..) de la classe Cell sur les bonbons toucher par l'utilisateur.

- `void mouseClick(Point mouseLoc)`

  Permet d'appeler la méthode mouseClick(..) de la classe Cell sur les bonbons toucher par l'utilisateur. Et d'aussi appeler la méthode deplacement(..) de la classe Mouvement et vérifie le score en appelant la classe Score.

- `void keyPressed(int keyCode)`

  Permet d'initialiser le niveau de jeu demander par l'utilisateur, de quitter le programme.

- `bool get_jouer()`

  Renvoi vrai lorsque l'utilisateur tape sur la touche ESPACE pour lancer le jeu.

- `void select_level(int niveau)`

  Permet de lancer le niveau à partir d'un fichier.

3.6 Classe MainWindow

La classe MainWindow permet de crée une fenêtre de jeu. Et appelle la classe Canvas pour créé le plateau de jeu 9x9 avec des bonbon.

- `MainWindow()`

  C'est le constructeur, il permet de crée une fenêtre de jeu .

- `void affiche_case_text_jeu()`

  Permet de crée des rectangle avec le score et les options du jeu sur la fenêtre. (Cette fonction prend les bonnes coordonnées pour l'affichage du jeu sur mac.)

- `void affiche_case_text_jeu_linux()`

  Permet de crée des rectangle avec le score et les options du jeu sur la fenêtre. (Cette fonction prend les bonnes coordonnées pour l'affichage du jeu sur Linux.)

- `void ecran_acceuil()`

  Crée un écran d'accueil du jeu. (Cette fonction prend les bonnes coordonnées pour l'affichage du jeu sur mac.)

- `void ecran_acceuil_linux()`

  Crée un écran d'accueil du jeu. (Cette fonction prend les bonnes coordonnées pour l'affichage du jeu sur Linux.)

- `void draw() override`

  Permet de dessiner tout ce qu'il faut dans la fenêtre.

- `int handle(int event) override`

  Permet de créer des évènement quand l'utilisateur utilise son clavier ou sa souris.

- `static void Timer_CB(void *userdata)`

  Permet de de rafraîchir la fenêtre du jeu.

3.7 Autres classes

les classes Rectangle, Text et Textectangle sont celle repris dans les laboratoires du cours INFOF202. Et les classes Plateau et Candy permette le dessin d'une image.[4]

La classe Common reprend toutes les libraires et les dimensions de la fenêtre du jeu.

4 Score

Le score dans notre programme est assez simple a chaque destruction d'un bonbon, le score s'augmente de dix.
Donc si il y a un alignement de trois bonbons, le score augmente de trente. Si un bonbon spécial est détruit, le score augmente de dix pour chaque bonbon supprimé par le bonbon spécial.

5 Logique du jeu

5.1 Visuel

Une fois l'exécutable de notre programme lancé, nous arrivons sur la page d'accueil.

Pour lancer le jeu, il suffit de touche la touche ESPACE. Si aucun niveau est sélectionné avant de lancer le jeu, le niveau amateur est lancé par défaut. Nous pouvons changer de niveau a tout moment en touchant la touche F ou D.

Une fois que nous déplaçons deux bonbons, notre programme vérifie que le déplacement est possible et qu'il permet l'alignement de trois bonbons minimum. Si celui ci est possible, le mouvement s'affiche et une animation de suppression aura lieux. Les bonbons des ranger supérieur vont tomber pour compléter les trou de la suppression et le score va s'augmenter de dix a chaque bonbon détruit.

5.2 Dans le code

Une fois le programme compilé, nous lançons l'exécutable (Candy_crush est le main). Candy_crush va faire appel à la classe Mainwindow de la bibliothèque FLTK, qui va faire appel à la classe score pour charger le score si le joueur avait déjà joué une partie.

Ensuite, la méthode ecran_accueil va être appelé. Puis va faire appel à la classe Text-Rectangle qui va permettre de creer un rectangle pour faire le fond de l'écran d'accueil. La méthode (ecran_accueil) va ensuite écrire du texte pour afficher les touches du jeu et des boxes pour afficher le score (score qu'on à déjà récupérer au début) avec la méthode draw qui permet de dessiner. Mainwindow appel les méthodes de la classe Canvas qui permet de détecter les touches clavier (méthode keypressed).

Une fois le niveau choisit (touche f pour les niveaux facile ou d pour difficiles), la classe

[4]PLatau et Candy viennes de `https://github.com/BDMR21/candy_crush`

Canvas va appelé la méthode lire_fichier pour lire le fichier en fonction du niveau choisit et va construire le plateau avec les différentes méthodes de la classe et sera affiché à l'écran par Mainwindow. Il détectera aussi les mouvements et clique de la souris. Si un clique est détecté et qu'il se trouve sur les coordonnées d'un bonbon alors il rajoute dans la méthode déplacement de la classe Mouvement, la référence de cellule demandé (la classe Cell permet de dessiner un bonbon dans une boxe).

Après avoir une liste de 2 éléments dans la méthode déplacement, vérifie si les bonbons sont à coté et s'ils font un alignement (en utilisant les différentes méthodes de la classe Mouvement). Alors, la méthode Effacer_bonbon est appelée de la classe Effacer_bonbon (qui met à zéro les bonbons dans la matrice) et la méthode animation. La classe Score est mise à jour (+10 par bonbon détruit).

Enfin tomber_fruit est appelée et va regarder les bonbons supprimés pour faire descendre les bonbons pour remplir le jeu. (bien évidemment à chaque mouvement le jeu est refresh à 60 images par secondes, donc à chaque fois la méthode draw dessine tous les mouvements que nous décrivons).

6 Conclusion

Ce projet nous a apporté de multiples enseignements, sur l'expérience d'un travail en équipe et d'un programme volumineux.

Tout d'abord, l'utilisation de FLTK et des différentes classes pour faire fonctionner notre programme a été un défi. Nous nous somme initiés au jeu Candy Crush pour comprendre ses fonctionnalités et ses règles. Nous avons aussi acquis de l'expérience en programmation orienté objet et nous avons mis en application notre cours d'analyse et méthodes.

Pour conclure, Nous avons appris à gérer un projet et à faire face aux difficultés ensemble.

7 Bibliographie et références

1. Les Laboratoires du cours INFOF202
2. `https://github.com/BDMR21/candy_crush`
3. `https://www.fltk.org`

7.1 Exemple d'exécution

Figure 1: Écran d'accueil après première exécution.

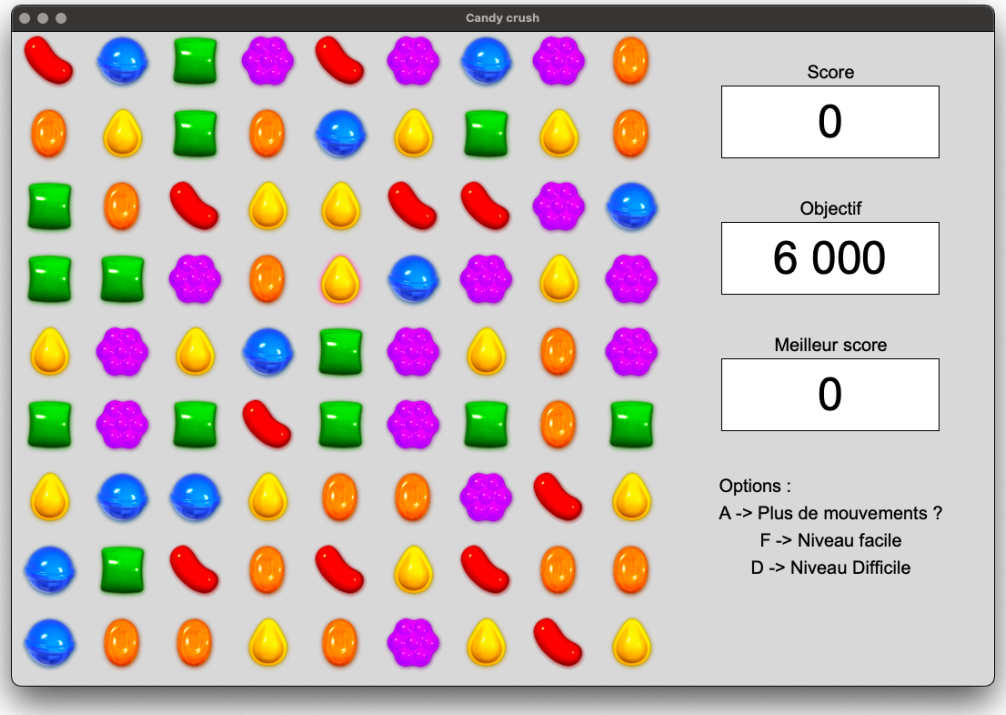

Figure 2: Ecran de jeu après avoir appuyer sur ESPACE.

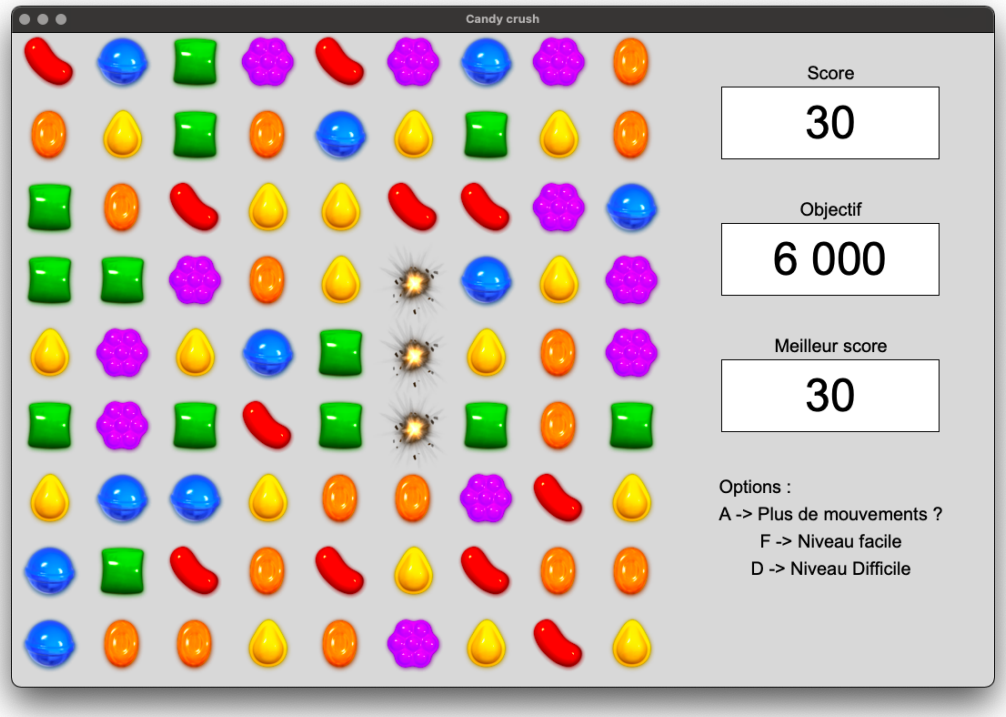

Figure 3: Ecran de jeu pendant le premier mouvement.

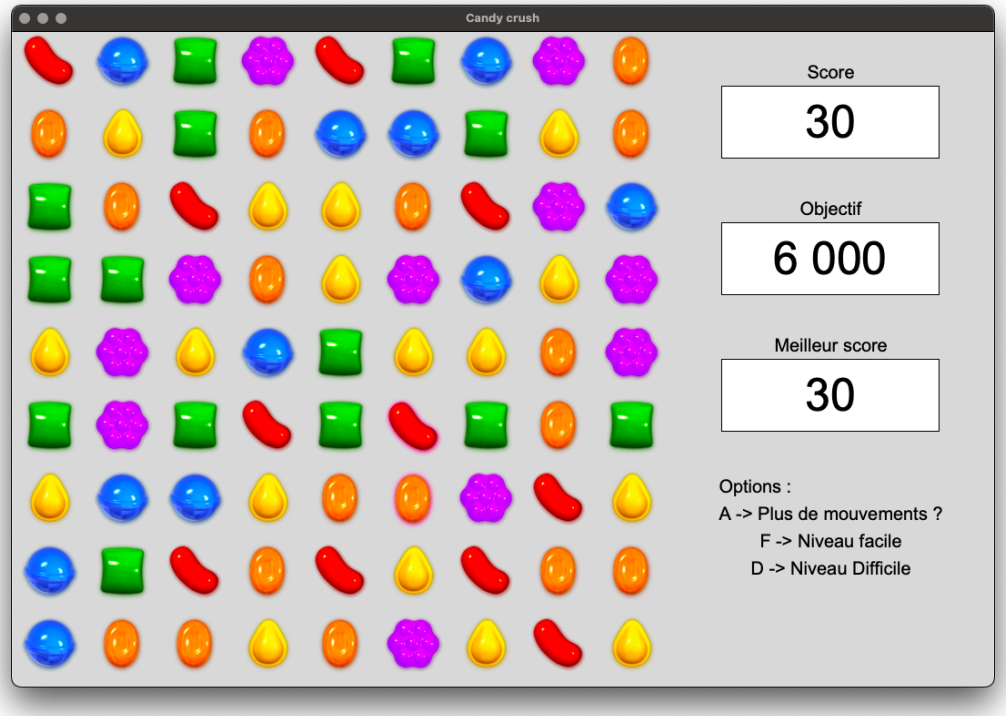

Figure 4: Ecran de jeu après le premier mouvement.

Figure 5: Ecran d'accueil après avoir relancer le jeu.