Année universitaire: 2013 / 2014

Projet

L3 INGENIERIE INFORMATIQUE

ROMAIN BEAUMESNIL - ELEONORE GEDEON – SULLIVAN PERRIN

Table des matières

1 Introduction

1.1 Travail Effectué

Comme vous le savez le sujet de ce projet était de faire un « candy crush like », nous avons donc effectué la version de base, à laquelle nous avons ajouté quelques fonctionnalités.

En particulier, l'ajout de bonbons spéciaux :

- ✓ Rayés horizontalement ou verticalement, qui détruisent la ligne ou la colonne où il se trouve lors de leur destruction.
- ✓ Sucrés qui détruisent un rectangle de 3*3 autour d'eux.
- ✓ Bombes qui détruisent tous les bonbons avec lesquelles les bombes sont interverties.

Ainsi que de cases spéciales :

- ✓ Des cases blocs, où les bonbons ne peuvent ni se trouver, ni les franchir/traverser.
- ✓ Des cases vides qui sont identiques aux blocs, à l'exception qu'elles sont franchissables.
- ✓ Des cases gélatines, qui correspondent un objectif supplémentaire pour les niveaux en comportant : détruire toutes les gélatines, ces dernières se détruisent si les bonbons présents sur ces cases sont détruits.

Et enfin, un système de niveau.

1.2 Choix de langage et de librairie graphique

1.2.1 Le langage

Concernant le langage, deux options s'offraient à nous, soit C++, soit Java.

Néanmoins Java a été écarté car nous nous sommes mis d'accord sur le fait que nous n'aimions pas Swing. Le choix du C++ s'est donc avéré être le plus intéressant, de plus nous avions codé en C++ lors du premier semestre, alors que si nous avions choisis Java, une remise à niveau aurait été nécessaire.

1.2.2 La librairie graphique

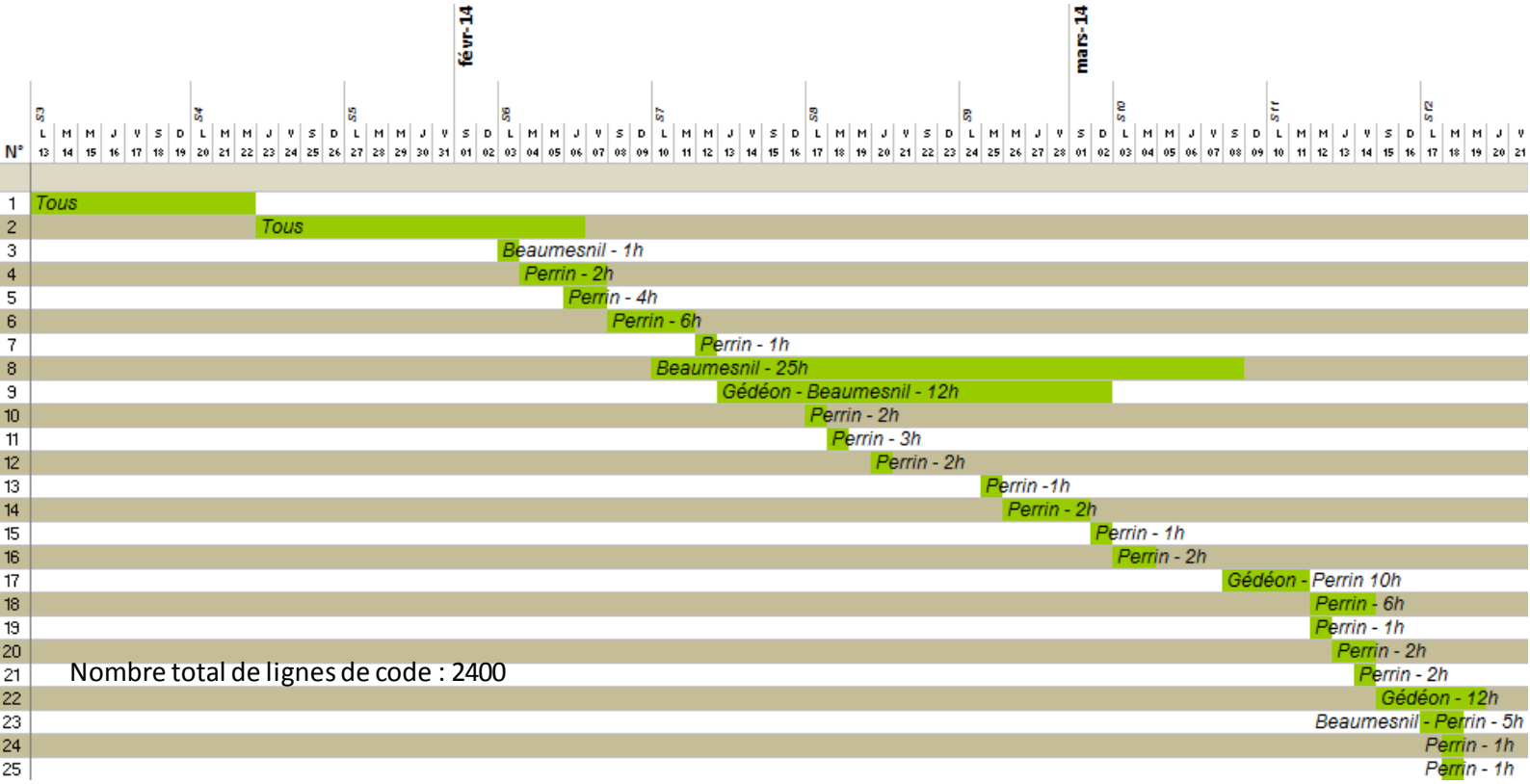
Une fois C++ choisi, un large panel de librairie graphique était disponible, plusieurs points étaient nécessaires : une prise en main rapide et simple, une librairie multiplateformes et si possible pensé en partie ou totalité pour les jeux et les animations.

Nous avons donc choisi, Qt Quick pour toutes les raisons citées ci-dessus.

Qt Quick est particulier sur plusieurs points :

- ✓ Qt Quick fournit un langage à base de xml : le qml. Le qml est un langage déclaratif qui permet de décrire les interfaces graphiques et si nécessaires d'effectuer des scripts grâce à du javascript.
- ✓ Qt possède un système de signaux et de slots, permettant de faire passer des messages/évènements, entres objets Qt.

1.3 Organisation



Nombre total de lignes de code : 2400

Diagramme de Gantt (cf. les informations sur les tâches ci-dessous)

	Début	Durée (jour)	Fin	N°
Choix langage et bibliothèque	13/01/14	10	22/01/14	1
Etude et choix architecture	23/01/14	15	06/02/14	2
Mise en place svn google	03/02/14	1	03/02/14	3
Classes en version basique	04/02/14	4	07/02/14	4
Vue version basique + images	06/02/14	2	07/02/14	5
Liaison QML/C++	08/02/14	4	11/02/14	6
Optimisation graphique	12/02/14	1	12/02/14	7
Fonctions coupPossible et auxiliaires	10/02/14	27	08/03/14	8
Fonctions destruction combos	13/02/14	18	02/03/14	9
Ajout icone	17/02/14	1	17/02/14	10
Ajout/Suppression bonbon/case	18/02/14	1	18/02/14	11
Constructeur niveau depuis fichier	20/02/14	1	20/02/14	12
Fonction remplir	25/02/14	1	25/02/14	13
Vue version finale, sélection bonbon	26/02/14	4	01/03/14	14
Animations de déplacements	02/03/14	1	02/03/14	15
Fonctions estPossible, deroulementJeux	03/03/14	2	04/03/14	16
Fonction tomber	08/03/14	4	11/03/14	17
Version de déploiement de l'application	12/03/14	3	14/03/14	18
Modification detruire combo en marquer combo	12/03/14	1	12/03/14	19
Fonction compterScore + gélatine	13/03/14	2	14/03/14	20
Animations explosions	14/03/14	1	14/03/14	21
Fonctions ajouterBonbonSpeciaux	15/03/14	5	19/03/14	22
Fonction redistribuer	17/03/14	2	18/03/14	23
Fonction estFini, plusAucuneGelatine	18/03/14	1	18/03/14	24
Règles de permutations des bonbons spéciaux	18/03/14	1	18/03/14	25

Après avoir effectués nos choix de langage et de librairie graphique, nous nous sommes séparés les tâches ainsi :

- ✓ Partie graphique : Sullivan PERRIN.
- ✓ Modèle et Logique algorithmique : Romain BEAUMESNIL et Eléonore GEDEON.
- ✓ Contrôleur : tout le monde.

Cependant cette répartition n'est pas figée, comme vous pouvez le voir sur le diagramme de Gantt, ci-dessus.

Concernant l'organisation lors du travail, dès que l'un de nous travaillait sur le projet, il se connectait sur Skype, afin de pouvoir échanger si nécessaire. Le cas échéant on utilisait les sms. Ainsi, on savait qui travaillait sur tel fonction et quel était son avancement.

Pour finir, nous avons utilisez l'IDE QtCreator sur Windows, ainsi tous les tests ont étés effectués sur Windows. De plus, la version de déploiement est disponible uniquement sur Windows.

2 Architecture

2.1 Présentation de l'architecture

De par son système de signaux et de slots, Qt à une architecture dite modèle-vue, néanmoins pour être au plus près du sujet nous avons utilisé une architecture modèle-vue-contrôleur.

2.1.1 Le contrôleur

Le contrôleur fait le lien entre le niveau et l'affichage graphique, il s'occupe de toutelalogique de déroulement du jeu, grâce aux fonctions *chargerNiveau(int n)*, *deroulementJeux()*, *selectionBonbon1(int x,int y)* et *selectionBonbon2(int x,int y).*

Il fournit également des signaux afin de notifier de changements de valeur,et/oud'évènement la vue.

Pour finir, il contient un pointeur vers le niveau en cours, les coordonnées des bonbons sélectionnés, des informations pour le déroulement du jeux (*timer,coefScore,etape*)oupourl'affichage (*resolutionBonbon, tailleBonbon, animationX, animationY*).

2.1.2 Le niveau

Le niveau contient toutes les cases (qui contiennent les bonbons) dans *liste*, ainsi que toutes les fonctions appelées par le contrôleur pour le déroulement du jeu en public (*completer, estPossible, detruire, commuterBonbon, coupPossible, tomber, compterScore, redistribuer, estFini, estBombe, estSpecial, ajouterBonbonSpeciaux, ajouterDeplace, plusAucuneGelatine*) et d'autre en privé (*remplir, combo*).

En plus de cela, niveau contient des données *static*, *viewer* et grille, ces données sont nécessaires pour la création des bonbons et des cases (graphique et modèle en même temps).

2.1.3 Les cases

Une case contient un pointeur sur un bonbon, des booléens pour savoir si la case est une case de début, de fin ou si elle est franchissable, ainsi que le niveau de gélatine.

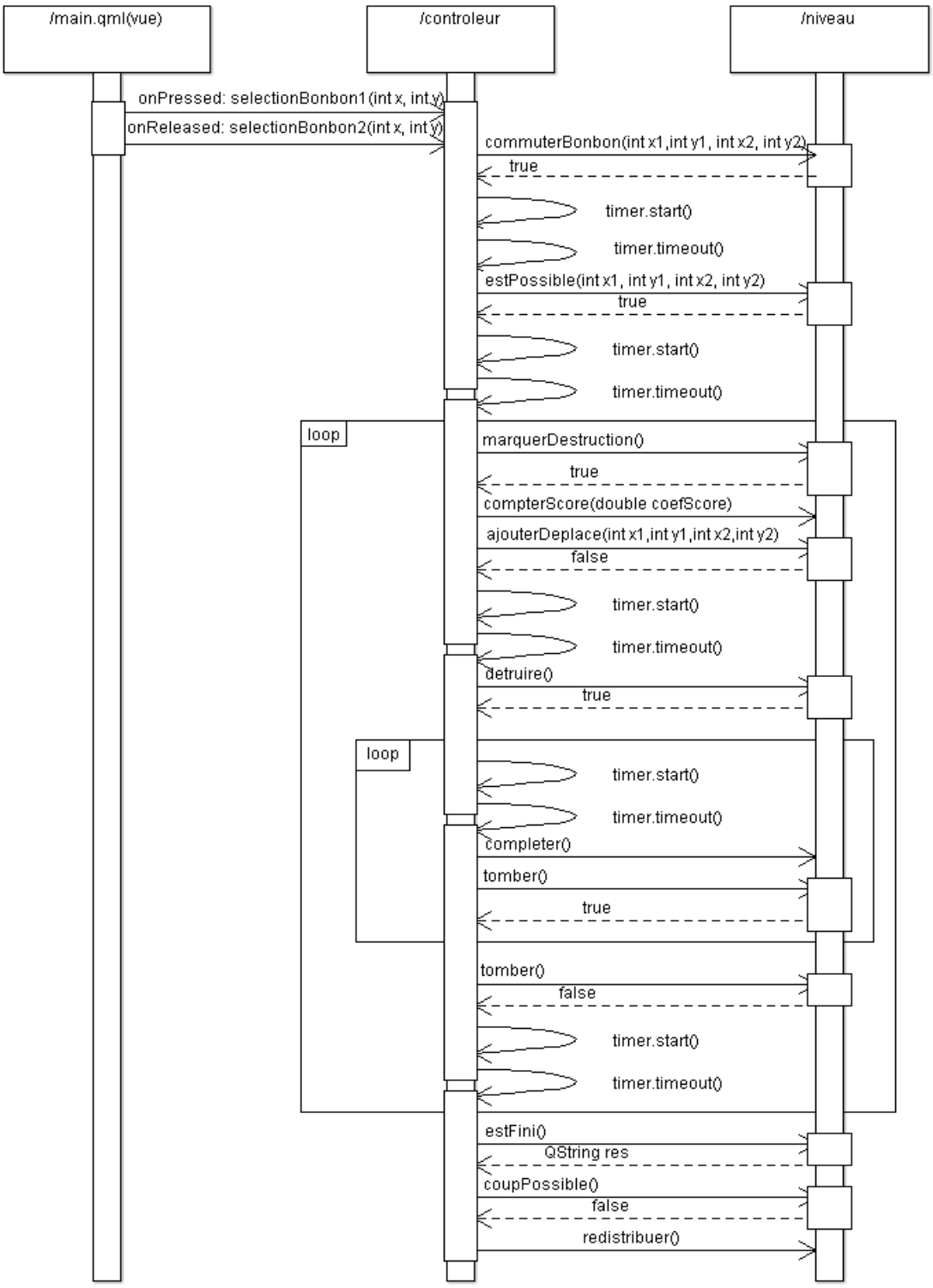
La classe *Case* est hérité de *QQuickItem* ce qui signifie que la vue et le modèle sontdirectement lié. En effet, pour créer une case on créé un composant qml *VueCase* qui sera immédiatement lié à l'instance correspondante de *Case*. C'est pourquoi *Case* a des signaux, cela permet de notifier les changements au composant qml *VueCase*.

De son côté *VueCase* a en plus des données partagées avec *Case*, deux entiers pour stocker sa ligne et sa colonne, afin de la placer dans la grille.

2.1.4 Les bonbons

Un bonbon contient deux valeurs d'énumération, un pour son type, et l'autre pour sa couleur. De la même manière que *Case*, *Bonbon* est hérité de *QQuickItem* et a un composant qml *VueBonbon*.

2.2 Scénario du déroulement du jeu/commutation de deux bonbons



Lors de la commutation de deux bonbons, un timer est déclenché, représenté par *timer.start()* et *timer.timeout()*, UML n'ayant pas de représentation de temps. Le timer permet la fluidité des différentes animations en chaque fonction appelée.

Puis, la fonction *estPossible* est appelée (correspondant à l'étape 1 de *deroulementJeu()* dans le contrôleur), vérifie si la commutation est possible et détruit ce qu'il faut si un des bonbons commuté est une bombe ou si les deux bonbons sont spéciaux.

Ensuite, *marquerDestruction* (étape 2) marque les bonbons à détruire, *compterScore* compte le score en fonction des bonbons marqué à détruire et *ajouterDeplacer* gère s'il y a formation de bonbons spéciaux sur ceux qui ont été déplacé.

Après, *detruire* est appelé (étape 3), puis *completer* et *tomber* remplissent les cases tant que cela est possible (étape 4).

Dès que *tomber* renvoie faux, on recommence à partir de l'étape 2 tant qu'il reste des combos à détruire.

Enfin, (étape -1) *estFini* vérifie si le jeu est fini, et *coupPossible* vérifie si il reste des coups possible et appelle *redistribuer* s'il n'en reste pas.

3 Conclusion

Le projet a été bénéfique suivant différent niveau pour chaque personne puisque par exemple Romain BEAUMESNIL avait déjà utilisé un SVN et avais déjà créé des interfaces graphiques, bien que ce n'était pas en Qt, cependant pour les autres ce fut une découverte. Cela nous également permit d'apprendre à travailler à plus de 2 personnes (comme c'était souvent le cas dans les projets précédents).

4 Annexe

4.1 Sources des images

Toutes les images utilisées dans le projet sont répertoriées dans le tableau suivant :

Nom image	Licence	Lien
Tous les bonbons + explosion	Domaine public	http://opengameart.org/content/candy-pack-1
Fond paysage	Domaine Public	http://opengameart.org/content/country-field
Bouton rejouer	GPL/Domaine Public	http://openiconlibrary.sourceforge.net/gallery2/?./Icons/actions/view-refresh-5.png
Bouton accueil	GPL	http://openiconlibrary.sourceforge.net/gallery2/?./Icons/actions/go-home-4.png

4.2 Adresse du serveur svn google

Si nécessaire, vous pouvez consulter notre SVN de travail à l'adresse suivante :
https://code.google.com/p/candy-crush-like/source/browse/