Projet Color Flood : Sprint 2

Équipe Gin Gin Money
Membres : Eléna Bouchaud, Hanna Ben Jedidia,
Hélène Tran et Nicolas Chevillard
Référente : Eléna Bouchaud

Introduction :

L'objectif de ce deuxième sprint est de réaliser le jeu de base de *Color Flood* sans solveur. Le joueur pourra choisir la taille de la grille et le nombre de coups autorisé.

A chaque boucle de jeu, la grille et le nombre de coups restants sont affichés. Le joueur doit alors sélectionner une couleur, ce qui fera afficher la grille modifiée.

A la fin de la partie, le programme affiche si le joueur a gagné, perdu ou si la partie n'est pas encore terminée (au moins deux couleurs encore présentes sur la grille).

Ce rapport présente les solutions apportées par l'équipe Gin Gin Money ainsi que son organisation pour la réalisation du sprint 2.

1- Solution proposée :

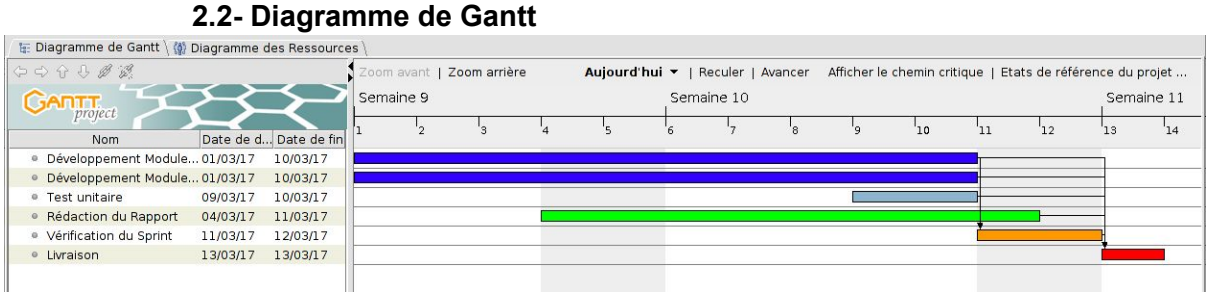
Pour créer cette application, nous nous sommes reposé sur les fonctions de base créées lors du premier sprint. Nous avons d'abord créer un module centré sur la réalisation du jeu en lui-même, avec des affichages en console. Dans ce module, nous avons utilisé abondamment les modules développés lors du sprint 1. Nous avons ainsi découvert certains problèmes dans certains de nos modules réalisés durant le sprint 1. Nous avons réglé ces problèmes et en parallèle, nous avons créé un module consacré au développement de la SDL.

2- Organisation :

2.1- Organisation de l'équipe

Nicolas et Hanna ont développé le module colorFlood.c. Nicolas a également réalisé le correctifs du sprint 1. Hélène et Elèna ont développé le module colorFloodSDL.c permettant d'afficher la grille avec la librairie SDL.

2.2- Diagramme de Gantt



2.3- Plan de charge de travail

	Hélène	Nicolas	Hanna	Eléna			
03/03/17		2					
04/03/17	3	3	3	3			
06/03/17		3					
09/03/17		4					
10/03/17	4	4	4	4			
11/03/17	5	5	5	5			
Somme	**12**	**21**	**12**	**12**		**Somme totale**	**57**

3- Algorithmie :

3.1- Module colorFlood.c

Ce module est constitué d'un main et de certaines fonctions permettant la communication avec le joueur.

Tout d'abord, dans le main, nous demandons au joueur de rentrer la taille de la grille. Pour celà, nous appelons la fonction *saisirTaille*. Cette fonction lit le caractère entré par l'utilisateur et le convertit en entier. Cette fonction est appelé dans une boucle while qui vérifie avant que la taille est bien comprise entre 3 et 25 à l'aide de la fonction *TestTaille*. Nous demandons ensuite au joueur de donner le nombre de coups maximum autorisé et pour celà, nous rappelons la fonction *saisirTaille*.

Ensuite, nous créons une grille remplie de manière aléatoire. Nous créons ensuite l'ensemble des composantes connexes (cases voisines ayant la même couleur) ainsi que la liste des composantes voisines de chacune d'entre elles. Nous recherchons ensuite la composante connexe principale c'est à dire celle qui possède la première case en haut à gauche. Pour réaliser toutes ces actions, nous utilisons les fonctions créer dans le sprint 1. Une fois les composantes connexes bien définies, nous affichons la grille à l'aide de la fonction *afficheGrille*. Cette fonction gère les appelles des fonctions *afficheInterLigneDessus* et *afficheLigneDessus* permettant ainsi d'afficher successivement la délimitation de lignes et les cases "colorées".

Le jeu est ensuite lancé. Nous demandons au joueur de choisir une couleur à l'aide de la fonction *saisirCouleur*. Cette fonction fait appelle à la fonction *getche* qui lit le caractère tapé par le joueur puis vérifie que le caractère correspond bien à l'initiale d'une des couleurs existantes. Une fois que le joueur donne une couleur existante, un caractère représentant la couleur demndé est renvoyé par *saisirCouleur.* Nous convertissons alors ce caractère en

une couleur puis nous décrémentons le nombre de coup , changeons la couleur de la composante connexe principale et affichons de nouveau la grille. Ces actions sont répétées tant que la grille n'est pas coloriée avec une seule couleur ou que le nombres de coups n'a pas atteint 0. Le test permettant de vérifier que la grille possède une unique couleur est réalisée par la fonction *testVictoire* créer lors du sprint 1.

Pour finir, nous vérifions si le nombre de coups et égale à 0. Si c'est le cas, alors le joueur a perdu : il n'a pas réussi à colorer la grille d'une seule couleur avec le nombre de coups qu'il s'est donné. Dans le cas contraire, le joueur a gagné.

3.2- SDL:

Tout d'abord, nous avons repris le support donné dans le cours d'IPI2 dans Scoledge, et nous en avons gardé les fonctions principales comme *drawTexture*, *drawRectangle*, *drawPixel*, *fillScreen.* Nous avons crée la fonction *initFenetre* qui crée l'écran où sera affiché le jeu. Nous avons également mis dans une fonction *quitter* les fonctions TTF_Quit et SDL_Quit qui permet de libérer l'espace mémoire associé à l'écran ainsi qu'à l'écriture. La fonction *afficheGrille2D* crée le tableau de cases de couleurs aléatoires (et cela grâce à l'import de Grille.h du Sprint 1) et qui affiche à l'écran le nombre de coups restants.

- planning (Gantt)
- plan de charge (tableau excel avec nos heures de travail)
- correctifs de ce qu'on a fait dans le sprint 1