**Rapport de projet C - Dicewars**

**Introduction**

Ce projet a pour but de réaliser un jeu appelé Dicewars. Pour cela il a fallu suivre une structure de projet d’ores et déjà définie et coder notre programme en langage C. Nous avons utilisé un gitlab pour faciliter notre travail en groupe.

Le but du jeu Dicewars est de contrôler tout les territoire, il se joue à un ou plusieurs joueurs et avec des IA. Chaque territoire possède des dés, si il veut prendre un territoire il mise ses dés contre ceux du territoire attaqué et si il fait un score plus gros, il récupère le territoire sinon il perd tous ses dés sauf un. Chaque joueur joue à son tour, et n’a pas la possibilité d’attaquer si il n’a qu’un seul dé sur son terrain. A la fin de son tour, le joueur reçoit un nombre de dés proportionnel à la taille de son plus gros territoire, si son territoire est coupé en deux par un adversaire, le nombre de dés ne sera que lié à la plus grosse moitié de son territoire. Si le joueur n’a plus de territoire il est éliminé.

Pour réaliser ce projet nous avons dû utiliser la librairie SDL2. Elle permet d’afficher une interface graphique grâce à un renderer qui gère une fenêtre à partir de laquelle on peut afficher des images. Il est également possible de gérer des événements d’input comme un appui sur une touche du clavier ou un clic de la souris.

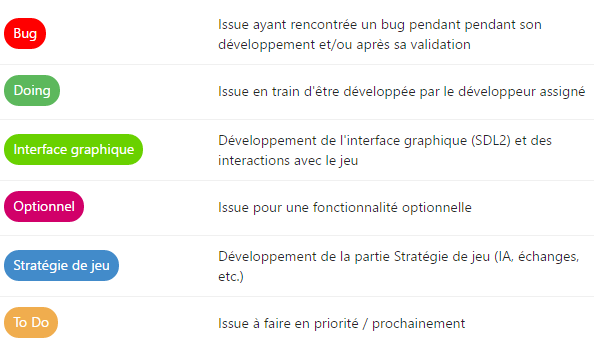
**Arborescence du projet**

Nous avons structuré notre projet de manière à faciliter la lecture, la modification et la compréhension, nous avons donc à la racine plusieurs dossiers et fichiers qui sont les suivants:

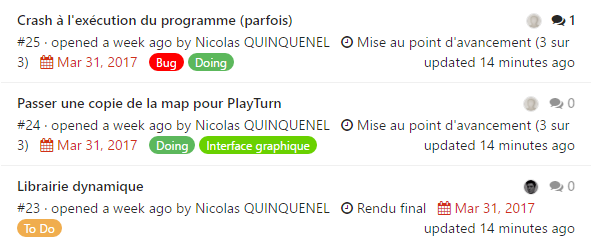
* Le **fichier readme** qui est un fichier explicatif du Projet DiceWars il répond à plusieurs questions essentielles telles que qu’est ce que le projet DiceWars ? Comment y jouer? Comment lancer l’exécutable ? Comment compiler ce projet ? Quelle organisation a été mise en place ?
* Un **dossier Images** qui contient l’ensembles des images utilisées par notre programme principalement les images de nos dés.
* Un **fichier Makefile** qui permet de compiler l’ensemble du projet et de nettoyer les fichiers créés lors des précédentes exécutions.
* Un **fichier exécutable** appelé DiceWars.exe qui permet de lancer le jeu.
* Un **dossier src** qui contient l'entièreté de notre programme celui-ci lui même divisé en plusieurs sous parties.
  + Un **dossier Interface** contenant tous les fichiers .c et .h relatifs à l’interface
    - **fenetre.c** : fichier contenant le “jeu”, c’est à dire toute l’interface
    - **fenetre.h** : fichier contenant notre librairie statique pour la fenetre
    - **free\_memoire.c** : fichier contenant toutes les fonctions pour l’interface
    - **generation.c** : fichier contenant les méthodes nécessaires à la création d’un graphe
    - **generation.h** : fichier contenant notre librairie statique pour la création d’un graphe
    - **jouer.c** : fichier qui contient les fonctions nécessaires à la mise à jour de l’affichage du jeu lorsqu’un joueur sélectionne un territoire et attaque un territoire.
    - **jouer.h** : fichier contenant la librairie statique pour les fonctions de mise à jour de l’affichage.
  + Un **dossier Librairies** qui contient notre librairie statique prédéfinie avec toute la promotion ainsi qu’une librairie statique de méthodes utilisées qui sont propres à notre groupe.
  + Un **dossier Strategie** contenant tous les fichiers .c et .h relatifs à la stratégie
    - **arbitre.c** : fichier qui vérifie que l’IA fait des coups réglementaires.
    - **idéeStratégies.txt** : fichier explique brièvement les marches à suivre selon nous pour avoir une IA la plus efficace possible.
    - **strategie.c** :
* Un **dossier bin** contenant tous les fichiers .o et autres fichiers créés pendant la compilation.

IMAGE ARBORESCENCE ATOM DU PROJET

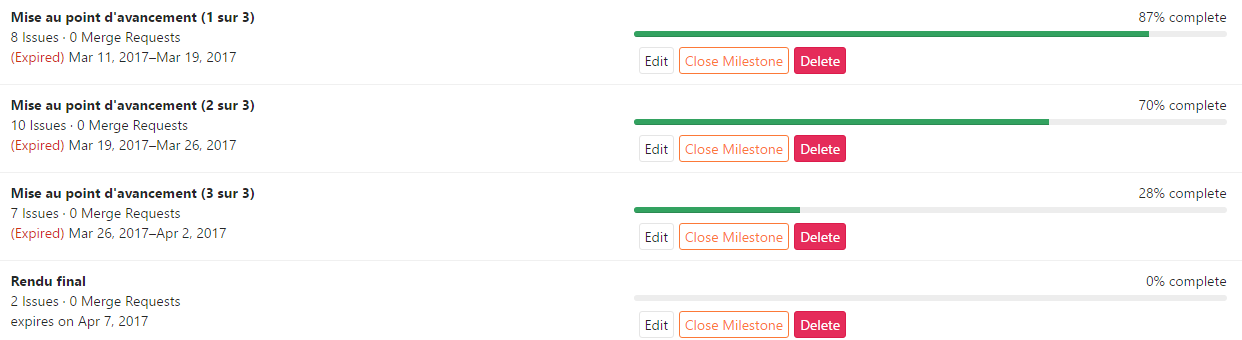
**Utilisation du gitlab**



En plus d’avoir réparti préalablement le travail au sein de notre équipe, nous avons utilisé le gitlab notamment pour différencier les types de problèmes à régler et leur priorité. Nous avons agencé notre planning pour pouvoir réaliser notre travail dans les temps et de manière optimale.



Ci-dessus, un extrait des problèmes rencontrés.



**Méthodes utilisées**

**Explication stratégie IA**

**Stratégie optimale :**

Essayer d'avoir des grappes de 8 territoires comme le nombre maximum de dés par territoire est de 8.

Joindre les territoires alliés:

Mieux vaut avoir 1 grosse grappe que plusieurs petites. Comme il n'y a aucun moyen de choisir quels dés de renforcement ira dans le territoire qu'on veut défendre, avoir + qu'une seule zone affaiblit toute stratégie. Le premier objectif est donc de liés les territoires ensemble, non seulement pour la défense, mais puisque le renforcement de dés à chaque fin de tours est basé sur le nombre de territoires de la plus grosse grappe.

Diviser les territoires ennemis:

Inversement, diviser un territoire ennemi affaiblit considérablement son contrôle. Au début du jeu, il est sage de décider d'une base d'o se développer, et ainsi ignorer les autres territoires. Ou, si des territoires ne sont pas trop espacés, essayer de les liés.

Ne pas passer son tour au début du jeu, ou tout du moins, pas trop longtemps.

Si un joueur a plus de la moitié des territoires, l'attaquer et délaisser les territoires des autres joueurs, quitte à passer son tour pour gagner des dés. Mais attaquer quand même un plus faible si cela permet de rejoindre une autre grappe de territoires alliés pour gagner plus de dés à la fin du tour, ou si nos territoires ont déjà 8 dés.

Quand on est le leader, et donc que tous les autres joueurs nous attaquent, ne pas faire trop de lignes avec peu de dés en attaquant. Sauf si tous (ou presque) nos territoires ont 8 dés.

Ne pas attaquer si on a moins de dès qu'un territoire adverse, voir même quand on en a autant.

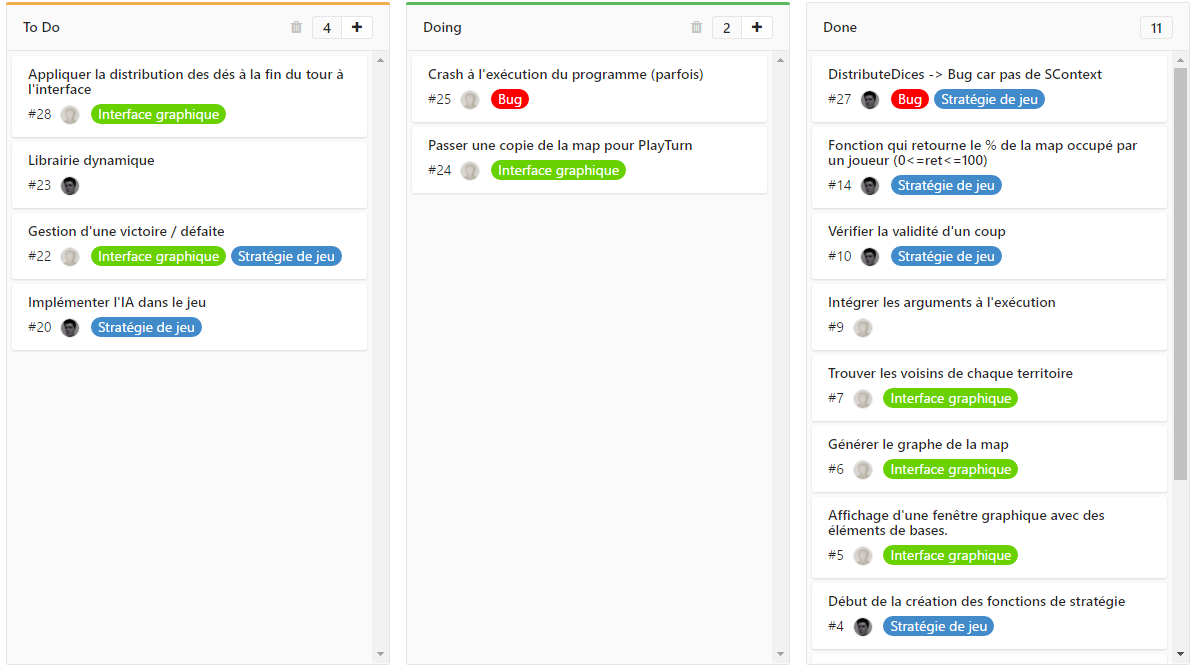
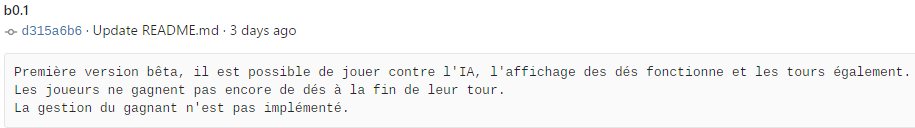
Essayer de ne pas bloquer des cellules avec beaucoup de dés dans un coins

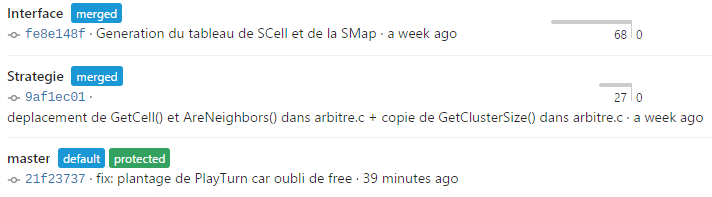
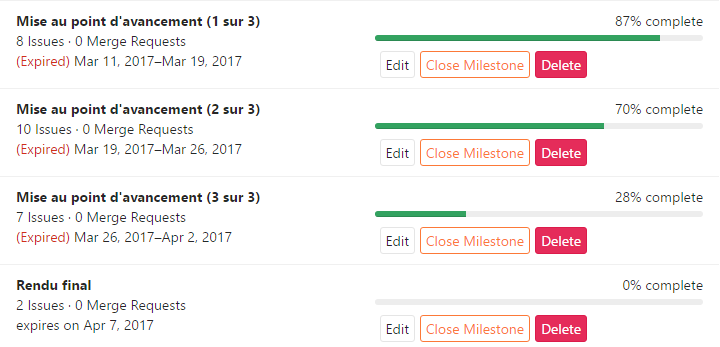
**Stratégie mise en place :**

Nos stratégies sont restées relativement simples, on s’est d’abord penché sur le fait d’attaquer n’importe quel territoire possible, puis par la suite nous avons décidé d’attaquer uniquement les territoires qui nous sont égaux ou inférieurs en nombre de dés. Après la création de cette méthode fonctionnelle nous avons ensuite pensé à deux stratégies différentes une attaquant avec la différence de préférence égale à un pour essayer de ne pas gaspiller de de grandes piles de dés sur des piles de un dé par exemple et de l’autre côté une stratégie qui attaque le territoire ayant le plus d’écart entre les piles de dés dans le but de favoriser nos chances de gagner. A cela on a ajouté une fonctionnalité ou les IA attaquent en priorité si un territoire est contrôlé par un des joueurs. Pour ne pas laisser une IA prendre trop d’avance sur les autres.

**Problèmes rencontrés**

Lors de notre projet, nous avons rencontré un certain nombre de problèmes notamment de fuites de mémoire, cela nous a pris du temps de savoir à quel endroit, la libération de mémoire avait été oublié, après des recherches, nous nous sommes rendus compte





Conclusion

Noter projet a été implémenté en respectant les différentes spécificités imposées, notre approche de faire des IA relativement simples et d’avoir une organisation bien structurée nous a permis de réaliser l’ensemble du projet et de ne pas faire de ne pas faire l’impasse sur différentes parties du projet.