Élèves : Simon CAMPANO & Nicolas MANIE
Groupe : 1A - Db
Encadrant : Mme Gonclaves

Projet d'algorithmique
Semestre 2

Sujet : Immuno Wars
Date : 18 mai 2010

Table des matières

Introduction 1

Sujet Personnel : Immuno Wars 1

Objectifs 1

Règles du jeu 1

Déplacement des pions 2

Type de partie 2

Déroulement de la partie 2

Analyse et Conception 3

Analyse 3

Conception 6

Organisation du développement 9

Jeu d'essais 10

Conclusion 14

Annexes :

- Code source
- Tableau des dépendances

Introduction

Le projet d'algorithmique constitue au deuxième semestre un enjeu d'envergure : c'est la mise en application des notions acquises en première année, il représente un fort investissement. Nous avons opté pour un sujet personnel afin d'avoir plus de liberté sur ce que nous allions faire, de pouvoir nous approprier plus étroitement le projet, mais aussi de diriger le réellement de A à Z.

Nous allons vous présenter notre sujet et la manière dont nous nous y somme pris pour relever les défis posés au cours du développement de ce jeu, en insistant sur les points qui nous ont parus important ou intéressants.

Sujet Personnel : Immuno Wars

Objectifs

- IA
- Interface Graphique

Règles du jeu

- Mode de base : 2 joueurs : Globules Blancs (GB) vs Virus.
- Objectif des GB : Tenir la charge des virus pendant un certain nombre de tours, ou bien les détruire.
- Objectif des virus : Prendre un point de contrôle ou détruire les GB.
- Mode Attaque : le joueur joue les virus. Mode Défense : le joueur joue les GB.

Pions

Globules Blancs (Taille Variable)

Pour tuer un virus, il suffit au GB de passer dessus.

- Niveau 1 (1 case) : Son but est de survivre pour passer au niveau 2.
- Niveau 2 (2 cases)
- Niveau 3 (4 cases) : A la possibilité de se diviser en 4 Globules blancs de niveau 1 en attrapant un Globule Rouge.

Virus (Taille constante : 1 Case)

Pour attaquer un GB, les virus doivent s'accrocher aux GB.

Il existe différents types de virus qui ont des propriétés variées sur le GB attaqué.

- Type 1 : Ralentit les GB.
- Type 2 : Immobilise les GB.
- Type 3 : Fait diminuer le niveau du GB à chaque tour, jusqu'à disparition.

Déplacement des pions

À chaque tour, les pions se déplacent d'un certain nombre de cases, qui détermine leur vitesse. Ils se déplacent selon le parcours de leur choix, dans la limite de leur vitesse.

Type de partie

On dispose de plusieurs variables permettant de faire varier la difficulté du jeu

- Nombre de GB initial
- Nombre de virus arrivant à chaque tour
- Nombre de Globules Rouges arrivant à chaque tour

Déroulement de la partie

- Génération du plateau de jeu choisi
- Tour des GB : ils doivent défendre les points de contrôle, attraper les GR pour se multiplier, tuer les virus.
- Tour des virus : ils arrivent par vagues définies. Ils doivent attaquer les points de contrôle et/ou tuer les GB.
- Fin de la partie

Analyse et Conception

Analyse

Données

- Données du problème : Grille de jeu, position des globules blancs et des différentes cases (entrée des virus, point de contrôle, entrée des globules rouges, cases bloquées) au départ
- Données utilisateur : Action à faire (Jouer, Éditer le niveau, Quitter le jeu), Type de jeu désiré (un contre un ou contre l'IA), coordonnées de départ et d'arrivée des pions à déplacer

Résultats

- Intermédiaires : Tour (nom du joueur qui doit jouer), Plateau mis à jour, État du point de contrôle, Nombre de tours restants avant que les globules gagnent
- Final : Résultat du jeu (Nom du gagnant), puis retour à l'écran d'accueil

Faisabilité / Conception

Structures de données pour la grille et les pions

On va utiliser une double structure de données pour stocker les pions : d'une part un tableau bidimensionnel qui va nous permettre d'accéder facilement aux informations d'une case (par exemple pour l'afficher ou obtenir les cases accessibles à proximité d'un pion), et d'autre part un tableau de listes chaînées de pions, qui révélera son intérêt lorsqu'il faudra parcourir tous les pions d'un certain type (par exemple pour faire évoluer les globules, ou pour l'IA).

L'idéal serait donc de ne pas stocker les pions dans le tableau, mais plutôt de faire pointer sa case vers son adresse. De cette façon, on aurait un double accès aux coordonnées, ce qui permet de naviguer facilement entre les deux structures selon les besoins et de les tenir à jour sans encombres.

Déplacement des pions

Le déplacement est complexe, puisque les pions ne se déplacent pas nécessairement en ligne droite, que certains prennent plus de place que d'autres, et qu'un déplacement peut avoir des implications diverses et variées (division d'un globule, suppression d'un pion, atteinte aux caractéristiques d'un globule).

Il faudrait pouvoir afficher facilement les cases accessibles. Celles-ci étant fonction de la vitesse, il serait pénible pour l'utilisateur d'avoir à compter les cases pour savoir où il peut aller.

Dans ces conditions, nous avons décider de générer un tableau appelé «masque» comprenant l'ensemble des cases accessible pour un pion donné. C'est ce masque, centré sur le pion, que l'on pourra «coller» sur le plateau pour indiquer les cases accessibles, qui nous indiquera aussi si la case d'arrivée est valide pour le pion, ou encore qui nous permettra de calculer le chemin emprunté par un pion pour aller d'un point à un autre.

Edition du plateau de départ

Dans l'optique de pouvoir éditer le niveau facilement, nous allons les stocker dans un fichier qui sera éditable directement via le logiciel *(évolutivité)*. Le plateau de jeu de départ devra donc être généré à partir de ce fichier.

Passage en mode graphique & organisation du code

Pour pouvoir passer facilement en mode graphique, toutes les fonctions concernant l'affichage devront être regroupées dans un seul et même fichier, dont les sous-programmes seront ensuite traduits de façon à afficher du graphique plutôt que du texte.

On étendra cette bonne pratique à tout le code, séparant clairement l'algorithme principal, le moteur du jeu, le générateur de plateau, l'IA et les fonctions d'affichage.

Intelligence artificielle (IA)

On va dans un premier temps réaliser une IA basique qui permette de choisir le meilleur coup à jouer pour les virus, afin que l'utilisateur puisse joueur seul. On pourra faire évoluer cet algorithme de façon à faire varier la difficulté du jeu *(évolutivité).*

Évolutivités

- Avoir une IA plus performante *(algorithme min/max par exemple)*
- Ajouter un mode Histoire avec différents organes pour l'infection
- Donner plus de fonctions aux pions, par exemple des globules blancs qui pourraient soigner leurs congénères
- Faire apparaître aléatoirement des boosters sur le plateau qui permettent d'augmenter la vitesse d'un pion dans la limite de la vitesse maximale (ce qui serait utile pour les globules ralentis)
- Avoir un mode de jeu en réseau
- Sauvegarder les statistiques de jeu

Objets Principaux

-Joueur : joueur (int)

-Grille de jeu : tableau_jeu[][] (S_case)

S_case : *-pion : pion (S_pion*)*

-type de case : type (int)

-Tableau des listes de pions : tab_listes[] (S_pion*)

S_pion : *-type de pion : type_pion (int)*

-niveau du pion : niveau (int)

-coordonnées du pion dans la grille : pos (coord)

-pion suivant dans la liste : suiv (S_pion)*

-Masque : masque[][] (S_masque)

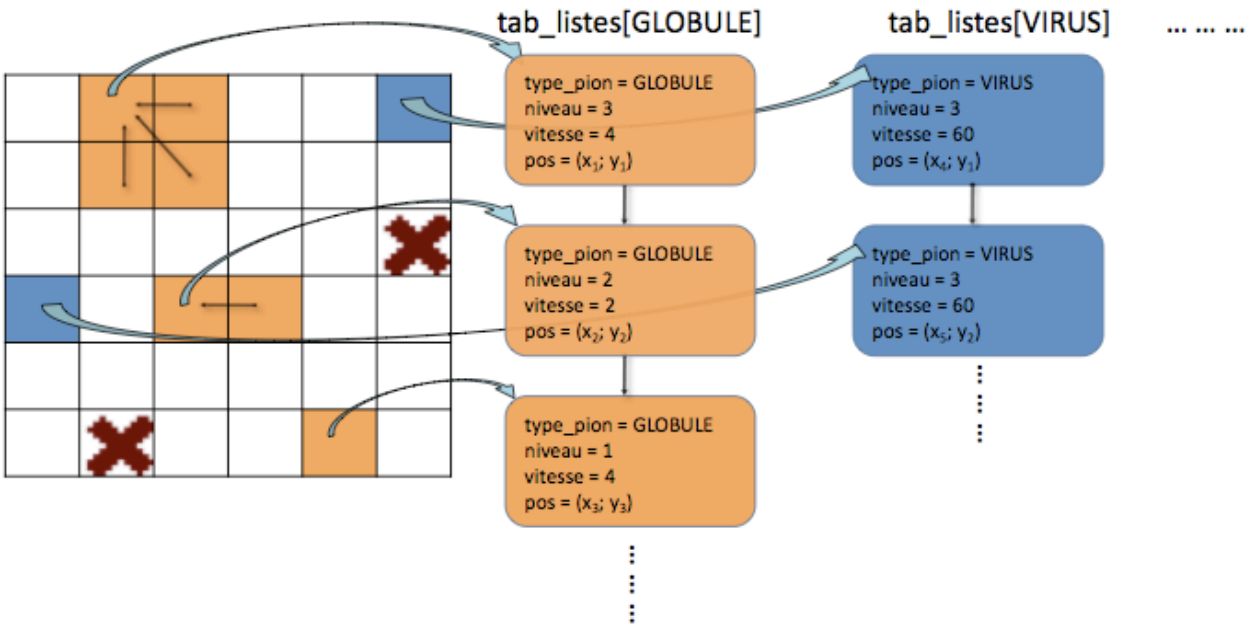
S_masque : *-état de la case : occupe (int)*

-distance à la case de départ (en coups) : poids (int)

-Chemin emprunté : L_chemin (S_chemin*)

S_chemin : *-coordonnées de l'étape du chemin : etape (coord)*

-étape suivante : suiv (S_chemin)*



Conception

Algorithmes Informels

Initialisation des données

Initialiser le plateau de jeu

Initialiser le tableau des listes de pions

Initialiser le masque

Initialiser le joueur

Initialiser le pointeur pour le chemin

Algorithme principal

Générer le plateau de jeu à partir du fichier dédié

Tant que les virus n'ont pas tué tous les globules ou détruit de point de contrôle **et que** les globules n'ont pas résisté le nombre de tours donné **faire**

Afficher le joueur

Afficher le plateau

Si le joueur a des pions **faire**

Tant que le joueur n'a pas saisi un couple de coordonnées valides **faire**

Demander au joueur de saisir l'un de ses pions

Générer le masque pour ce pion

Demander au joueur de saisir les coordonnées de destination du pion

Calculer le chemin

Déplacer le pion

Faire entrer des globules rouges

Faire entrer des virus

Faire évoluer des globules blancs

Changer de joueur

Afficher le gagnant

Algorithme récursif «parcours» de pondération du masque préalablement initialisé à partir de la grille de jeu

Si la case du masque n'est pas occupée

Donner le poids actuel à la case

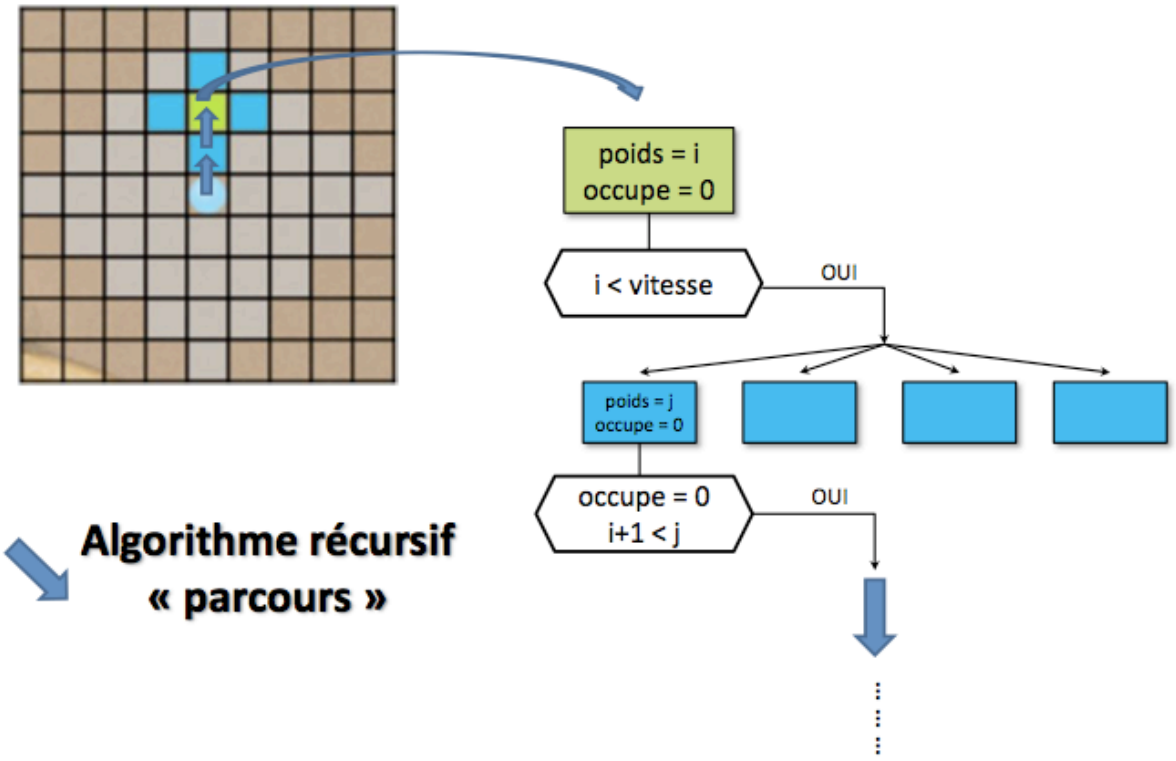
Incrémenter le poids actuel

Si le pion peut continuer à avancer

Tant que les quatre cases avoisinantes n'ont pas été parcourues, pour chacune faire

Si son poids est supérieur au poids actuel ou qu'elle est libre

Appliquer ce même algorithme de parcours à la case



Algorithme récursif de calcul du chemin à partir du masque

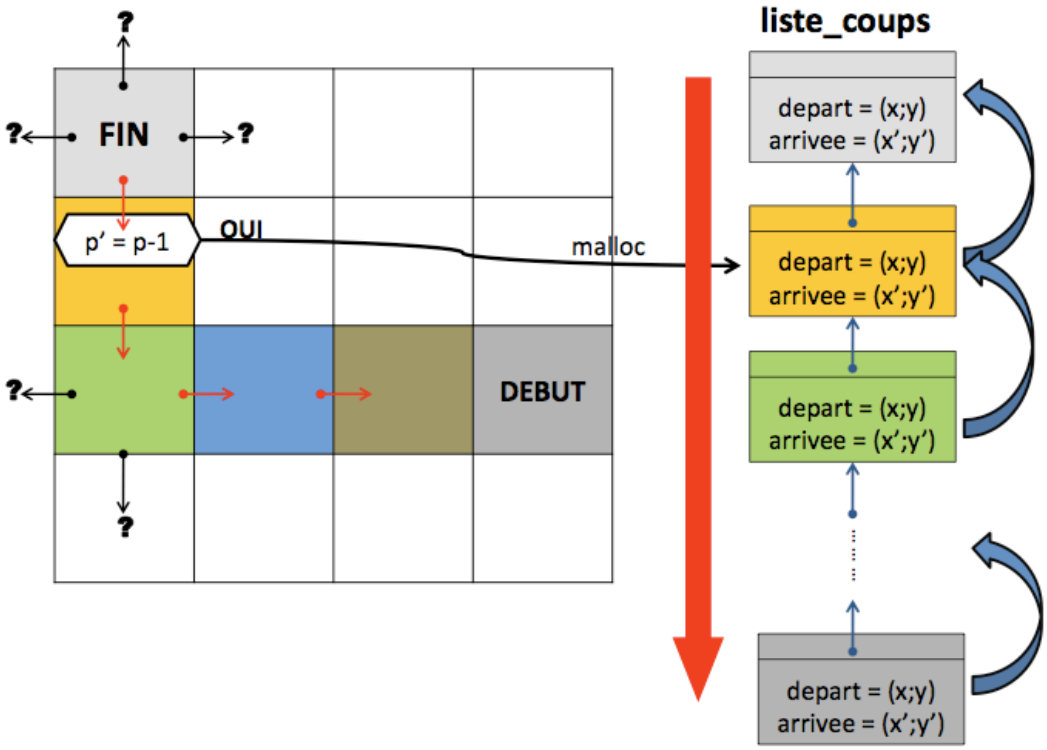
Créer un élément chemin à partir de la case actuelle

Si la case actuelle n'est pas la case d'arrivée

Chercher une case voisine de poids inférieur

Lui appliquer ce même algorithme

Chaîner l'élément (remontée)



Organisation du développement

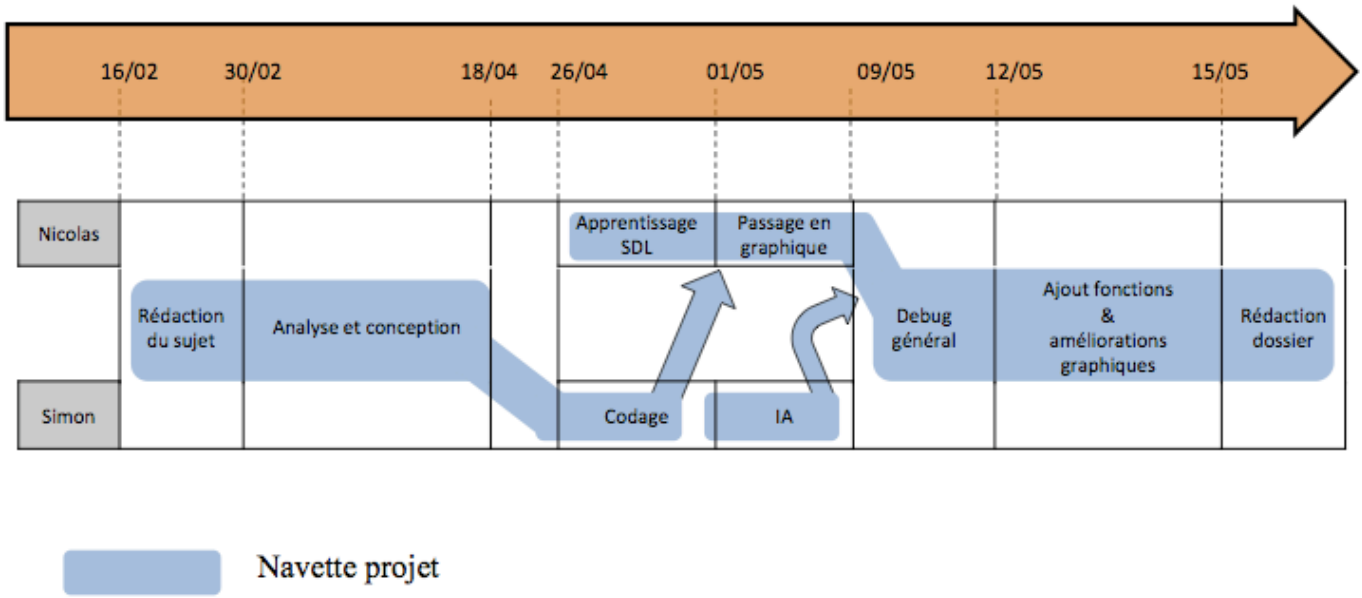
Afin de mener à bien ce projet, nous avons décidé de répartir la charge de travail pour pouvoir développer de façon indépendante les modules que nous nous sommes attribués durant la phase de conception.

Nous avons ainsi fait particulièrement attention à garder les différents modules les plus indépendants possible les uns par rapport aux autres.

Ainsi, comme le montre le tableau des dépendances, les fichiers respectent une certaine logique. Par exemple, `partie` appelle `jeu` qui contient toutes les fonctions du moteur de jeu, qui lui-même appelle des fonctions de deuxième ordre de `tableau_jeu`.

C'est aussi cela qui nous a permis de passer facilement de la console au graphique : toutes les fonctions concernant l'affichage ayant été réunies dans un même fichier, il a suffit de modifier les anciennes fonctions pour le graphique, et d'adapter légèrement le main (devenu `partie` pour le passage en graphique), mais en aucun cas cela n'a affecté les fichiers du coeur du jeu (`jeu`, `listes`, `masque`, `tableau_jeu`).

Cette modularité a permis d'organiser un développement parallèle et relativement indépendant, ce qui s'est révélé particulièrement pratique lors des phases de débug et d'ajouts de fonctionnalités.



Jeu d'essais

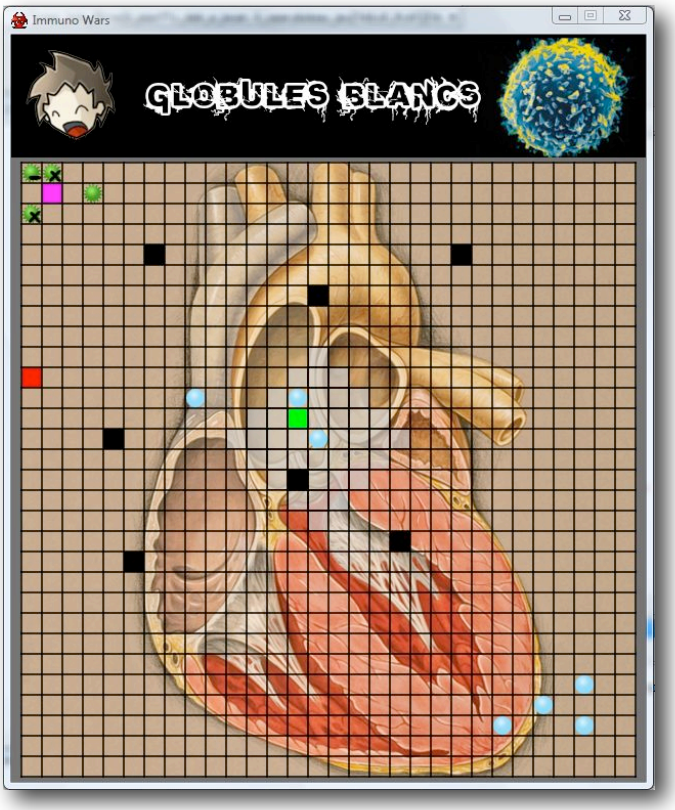
Écran d'accueil

Ici trois options sont proposées : on peut entamer une partie, éditer le plateau de jeu ou bien quitter le jeu.

On remarque aussi que la musique d'introduction est désactivable.

Nouvelle Partie

Si l'on décide de jouer une nouvelle partie, alors on a le choix entre jouer en mode 1 contre 1 ou bien contre l'IA



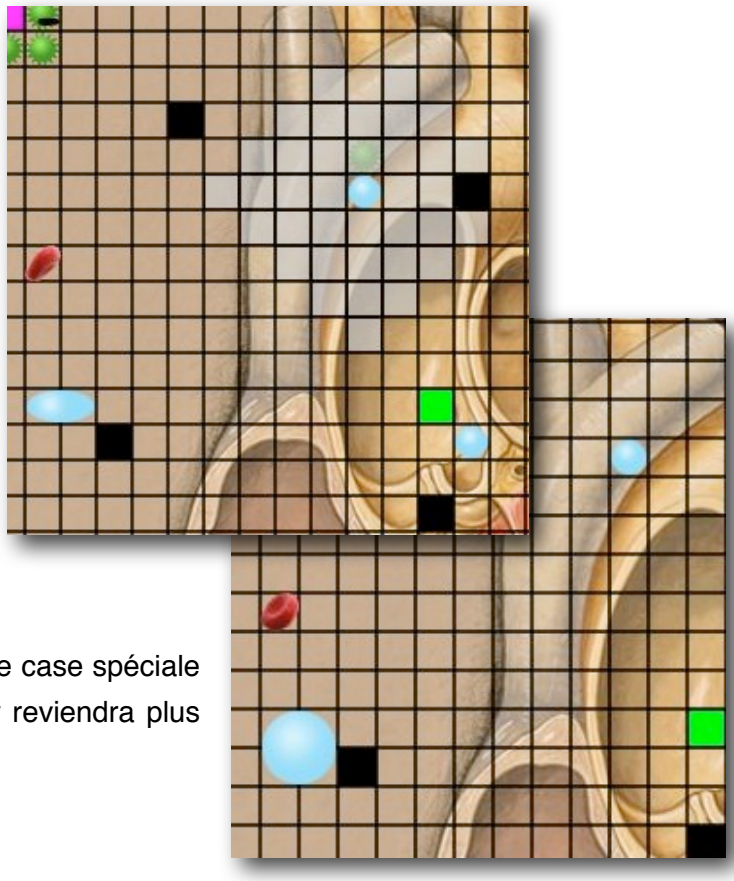
Sélection d'un pion

Attardons-nous un peu sur la sélection du pion, le masque se met alors en surbrillance, et indique ainsi l'ensemble des cases accessibles pour le pion en question.

Lorsqu'on sélectionne un point en-dehors du masque, alors le pion est déselectionné et la masque disparaît.

Lors du déplacement du pion, on remarque que celui-ci suit un chemin cohérent (il contourne les cases non accessibles)

Attaque par un globule et évolution de globule



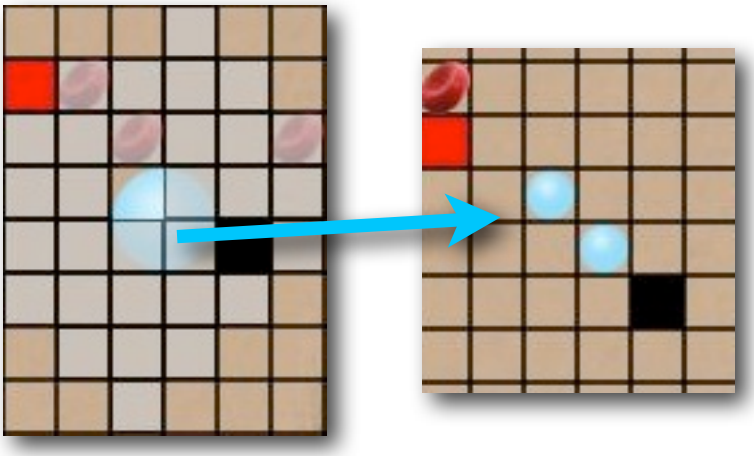
Sur ces captures d'écran, on peut voir au niveau du masque que lorsqu'un globule blanc de niveau 1 va sur un virus, alors celui-ci disparaît. Cela est valable pour tous les niveaux de globules.

En bas à gauche on peut observer une évolution de globule de niveau 2 en globule de niveau 3

Les cases noires sont des cases bloquées, et elles sont inaccessibles à tous les pions.

La case verte est le point de contrôle, c'est la seule case spéciale sur laquelle des pions peuvent aller : les virus. On y reviendra plus loin.

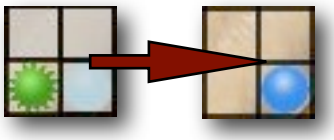
Division d'un globule blanc



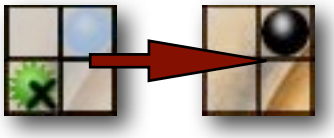
La division est un point important du jeu : c'est le seul moyen pour les globules d'augmenter en nombre. Elle ne se fait que lorsqu'un GB de niveau 3 attrape un globule rouge.

Il se divise alors en deux GB de niveau 1.

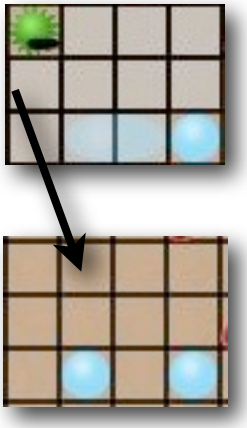
Les attaques de virus



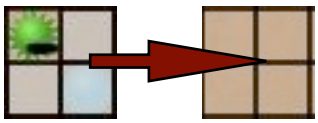
Les virus, en fonction de leur type, ont des actions variées sur les globules. Ici par exemple un virus de **type normal** attaque un globule. La vitesse de ce dernier se voit amoindrie, ce qui est signifié par l'obscurcissement du bleu clair au bleu foncé.



On observe ensuite l'action d'un virus **paralyseur** (marqué d'une croix) : le globule ne peut plus bouger, on voit cela à sa couleur noire.



Enfin, le virus le plus puissant est celui qui **amoindrit le niveau des globules**, il les fait régresser, ce qui comme on peut le constater sur ces jeux d'essais se révèle létal pour les GB de niveau 1.



L'une des façons pour les virus de gagner la partie est de tuer de cette façon tous les GB avant le nombre de tours impartis.

L'autre façon est d'atteindre un certain nombre de fois la case de contrôle (verte), toujours avant le nombre de tour impartis. Plus on se rapproche de sa destruction, plus Toto (la mascotte en haut à gauche de l'écran) semble malade.

IA

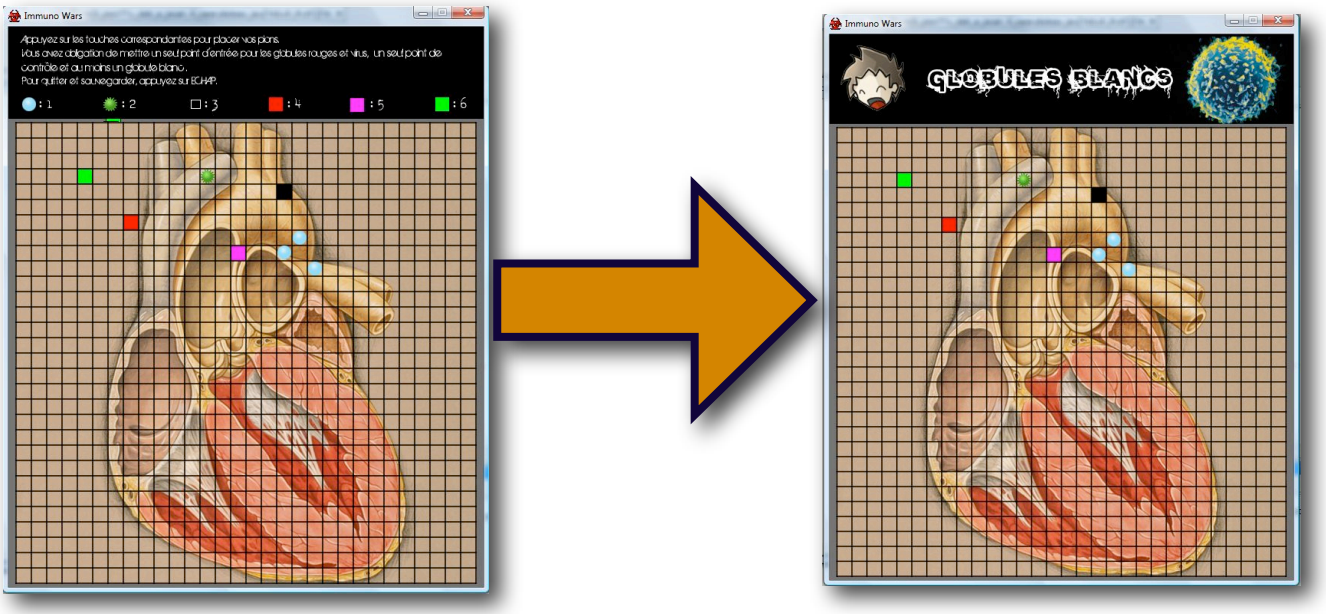
L'IA est compliquée à capturer en image, on laisse aux correcteurs le soin de vérifier qu'elle est fonctionnelle.

Victoire

Lorsque l'une des deux équipes gagne, cela est affiché, puis on est redirigé vers l'écran d'accueil.

Éditeur

On vérifie que l'on peut bien placer les différents types de pions via l'éditeur, et qu'une fois les changements sauvés, ils se répercutent sur le plateau de la partie suivante.



Conclusion

Ces trois mois d'investissement dans un projet à mener du début à la fin nous auront beaucoup appris, dans des domaines variés.

Tout d'abord, le travail en équipe a été un réel bonus, stimulant et efficace grâce à une organisation qui nous a permis d'avoir une vue d'ensemble sur le projet, mais aussi de nous spécialiser un peu chacun dans une branche du projet. Notre méthode de gestion du temps nous a été particulièrement profitable, codant chacun de notre côté (en ayant au préalable parfaitement défini les structures de données et le fonctionnement du programme), et profitant des TPs pour parler des problèmes généraux et fusionner nos travaux.

L'utilisation au maximum de fichiers organisés, noms de fonctions et de constantes aussi clairs que possibles nous a rendu très aisées compréhension des codes réciproques et relecture sans surcharger le code de commentaires.

Nous avons aussi appris à nos dépens qu'il est parfois complexe d'anticiper les besoins d'un programme, et qu'il est dur de revenir sur des erreurs profondément ancrées dans le code, notre longue période de conception a donc pris tout son sens lorsque nous avons dû implémenter de nouveaux modules ou évolutivités.