Simulation Multi-Agents
Humains vs Zombies

Maxime ESCOURBIAC
Jean-Christophe SEPTIER

ZZ2 F2

Plan

Objectifs

l'objectif de ce mini-projet est de mettre en place un simulateur multi-agents pour mette en application la dernière partie de cours de simulation portant sur la SMA.
Les seules consignes reçues sont:

- La présence d'au moins 2 agents.
- Un monde «ouvert».
- Le choix du thème sera libre

Pour compléter cette partie, on réalisera un tutoriel chacun portant sur l'utilisation d'un outil utilisé dans le génie logiciel et plus particulièrement dans la partie simulation du projet.

I Présentation des modèles choisis

1°) Humain:

Les humains seront représentés dans cette simulation par des agents cognitifs qui auront les propriétés suivantes:

- Vitesse (entier compris entre 1 et 3)
- Force (entier compris entre 1 et 10)
- Perception visuelle primaire et secondaire (détaillés ci- dessous)
- Perception auditive primaire et secondaire (détaillés ci- dessous)

En cas de perception de plusieurs entités les interactions avec les autres entités se feront dans l'ordre suivant:
Zombie, Arme Humain.

Interactions avec les zombies:

Les humains les fuient dès que possible, si un zombie est détecté lors d'une perception alors l'humain fuira dans le sens opposé. En cas de rencontre, il y aura combat.

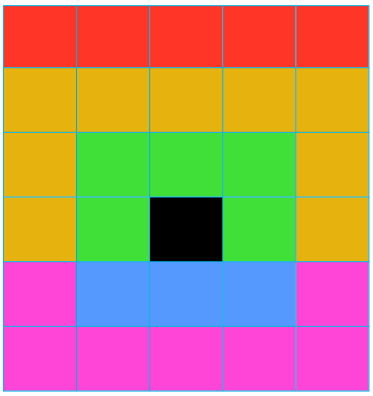
Interactions avec les autres humains:

Les humains seront interactifs que par la perception visuelles alors l'humain ira dans le sens de l'humain détecté. En cas de rencontre, on considérera les humains solidaires, c'est à dire qu'ils partageront les points de force.
Précision concernant la perception auditive, l'humain sera pas faire la différence entre un humain et un zombie donc il fuira dans la direction opposé au bruit.

Interactions avec les armes:

Les armes seront détectables que par la perception visuelle . En cas de rencontre, on ajoutera les points de l'arme à la force de l'humain qui les ramassera.

cône de perception: (exemple de perception vers le nord)



vert : perception visuelle immédiate 100% de perception
orange: perception visuelle primaire 50-75% de perception
rouge: perception visuelle secondaire 20-50% de perception
bleu : perception auditive primaire 40-60% de perception
violet: perception auditive secondaire 10-30% de perception

2°) Zombie:

Les zombies seront représentés dans cette simulation par des agents cognitifs qui auront les propriétés suivantes:

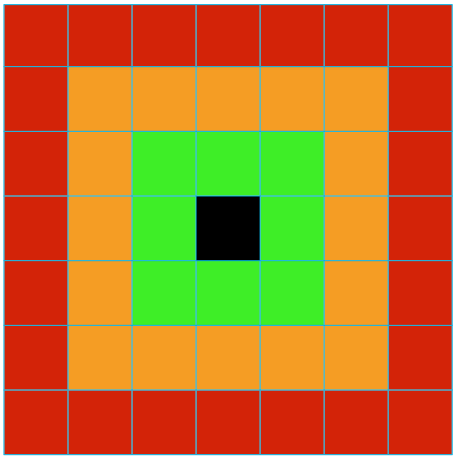
- Vitesse (entier compris entre 1 et 2)
- Force (entier compris entre 8 et 18)
- Perception olfactive primaire et secondaire (détaillés ci- dessous)

Les zombies réagiront qu'à la présence d'humain. donc les interactions avec les autres zombies et les armes sont nulles. On considéra que le zombie ne repérera pas à la vue ou au son mais uniquement à l'odorat. Concernant la vitesse on a fait le choix de la diminuer par rapport à celle des humains. De plus la force du zombie à été volontairement augmenté par rapport à la force humaine pour qu'en combat à mains nues le zombie soit vainqueur dans la plupart des cas. Cependant l'issue du combat aura une part d'aléatoire ce qui apportera plus de profondeur quand aux résultats de la simulation.

Interactions avec les humains:

En cas de perceptions d'un humain, le zombie accourra vers lui, si il y a rencontre alors il y aura un combat, si le zombie en sors vainqueur alors il y aura possibilité de contamination, c'est à dire qu'un nouveaux zombie apparaissent à la place de l'humain.
(probabilité de l'événement = 70%)

cône de perception: (la perception prend pas en compte le sens de déplacement)



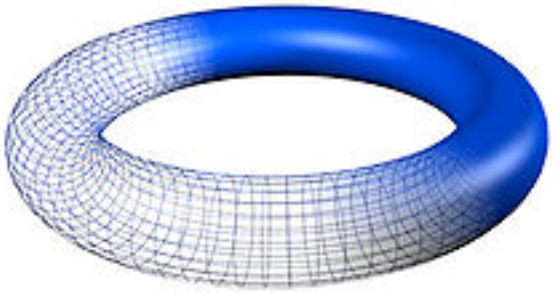
vert : perception olfactive immédiate 100% de perception
orange: perception olfactive primaire 50-75% de perception
rouge: perception olfactive secondaire 20-50% de perception

3°)Arme:

Les armes seront représentées dans cette simulation comme des entités fixes, leur seule propriétés est leur force (entier variant de 1 à 5).
Elles seront utilisables seulement par les humains, et leur action sera d'augmenter la force de l'humain. Une fois ramassé une arme est pas réutilisable même en cas de mort du propriétaire.

4°) Monde:

La consigne sur le monde où évolueront les agents était qu'il soit ouvert, c'est à dire qu'il se comporte comme un tore. C'est à dire que les extrémités Nord-Sud soient reliées ainsi que les extrémités Est-Ouest.



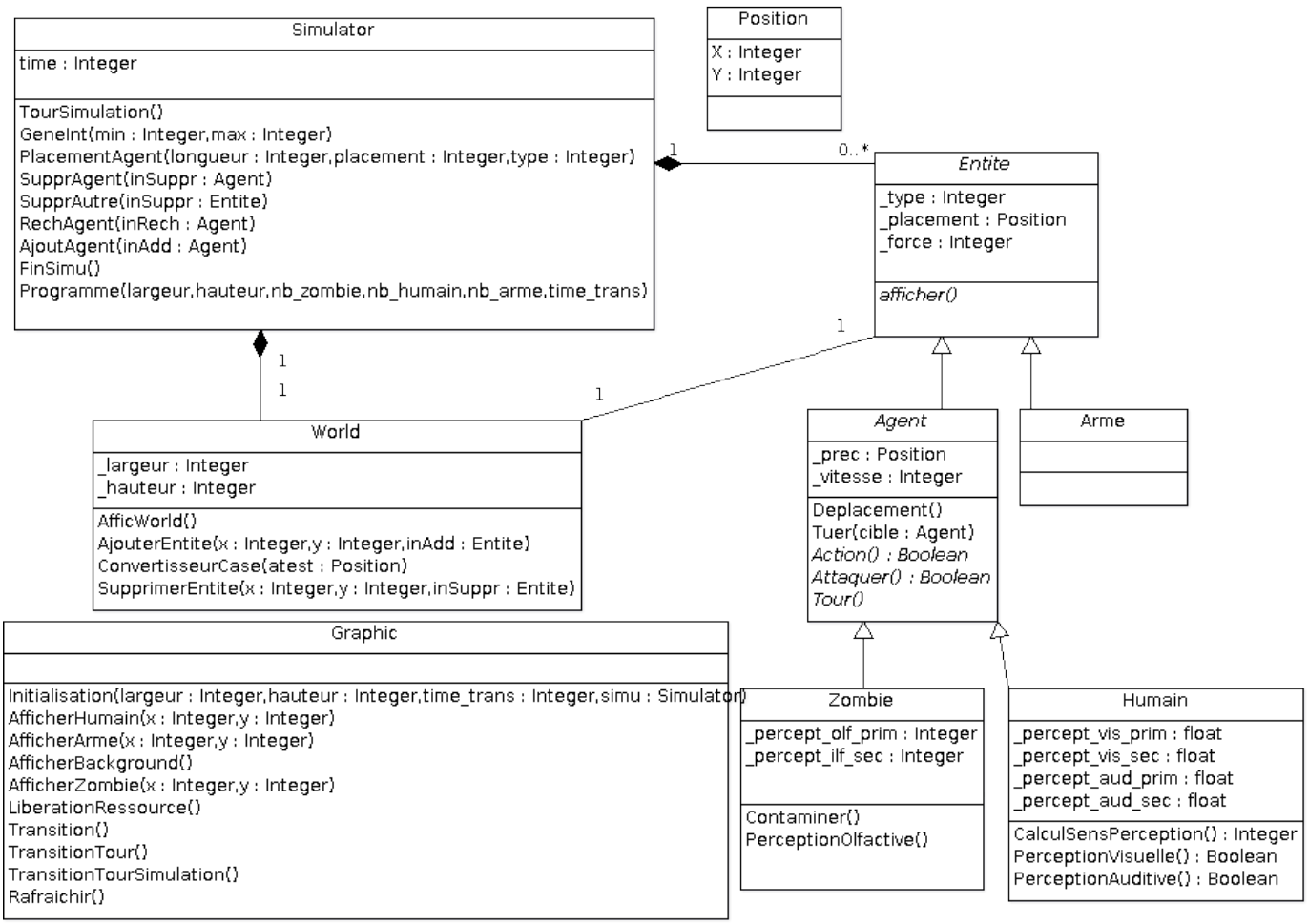
représentation d'un tore
source: wiktionary

Le monde sera donc représenté comme un rectangle dont la taille sera demandé à chaque lancement de la simulation.

La modélisation du monde choisis permettra de stocker plusieurs entités sur la même position.

II Modélisation UML et choix des technologies

Modèle UML du simulateur :



Réalisé avec ArgoUML

La classe principale de ce modèle est le simulateur. Cette classe possède les entités, ainsi que le monde. Cette classe permet de faire tourner la simulation. La méthode TourSimulation est appeler pour permettre les changements de chaque entité pour un tour. Elle permet de placer un agent sur le monde, de supprimer les agents et les armes, ou de les ajouter. C'est elle qui permet donc de déplacer les agents sur le monde. La méthode programme est celle qui est appelé pour lancer la simulation, selon le nombre de zombies et d'humains, le nombre d'arme, le nombre de tour et la taille du monde.

La classe entités est une classe abstraite. Elle représente les entités présente dans le monde. Elle possède un type, ainsi qu'un placement sur le monde et une force.

Cette classe se dérive en deux classes : une classe arme et une classe Agent.

La classe arme représente représente les armes qui peuvent être ramasser par les humains, tandis que les agents sont les éléments qui se déplace et interagisse entre, c'est à dire, pour notre simulation, des zombies et des humains. Cette classe possède une position précédente pour savoir dans quel sens regarde les agents. Elle possède également une vitesse, qui représente le nombre de case que peut se déplacer l'agent pendant un tour.

Ces méthodes permettent le déplacement des agents, les combats entre agents, ainsi que les interactions entre agents.

Cette classe se dérive ensuite en deux autres classes : zombie et humain.

La classe zombie a en paramètre les valeurs de perceptions, ainsi qu'une méthode contaminer (qui transforme un humain en zombie), et une méthode PerceptionOlfactive qui permette de percevoir les agents grâce au valeur de perception.

La classe Humain possède également les valeurs de perceptions visuel et auditive. Elle possède des méthodes qui permette de lancer les différents tests de perception des humains.

La classe monde est représenté par une tableau à 2 dimensions de liste d'entité. Si une case du tableau est nulle, c'est qu'il n'y a rien sur la case. Sinon, la liste montre tout les entités présente à cette endroit. Le méthode permette de gérer l'affichage du monde, et le déplacement de entité.

La classe Graphic permet d'afficher le monde. Pour modifier la partie graphique, il suffit de modifier cette classe. Le reste ne se modifie pas.

Elle permet grâce à ces méthodes d'afficher les humains, les armes et les zombies et d'afficher une fond d'écran.

Choix des Technologies:

- Langage : C++ norme Ansi
- Corps métier : STL
- Générateur aléatoire : Mersenne Twister
- Interfaces Graphique: SDL, SDL_image, FMOD
- Documentation : Doxygen
- Outils: NetBeans 6.8 ,Valgrind, Gdb

III Algorithmes et choix logiciels.

1°) Algorithmes de perception:

Pour cette simulation, on a besoin de 3 algorithmes de perceptions:

- Perception Visuelle Humaine
- Perception Auditive Humaine
- Perception Olfactive des Zombies

Ces 3 algorithmes possèdent la même structure c'est à dire plusieurs échelle de perception (immédiate, primaire et secondaire pour les perception visuelle et olfactive).
C'est pour cela que l'algorithme de principe qui va être présenté ci-dessous sera un modèle générique trois perceptions.

Algorithme de perception:

```
calcul du sens de perception; //uniquement pour les perceptions humaine
stockage des cases à analyser pour la perception immédiate ;
//analyse des cases de la perception visuelle immédiate
Tant que (il reste des cases à analyser) //et qu'un zombie a pas été détecté pour les humains
|  tirage d'une case aléatoire
|  Si ( la case est non vide)
|  |  Tant que ( toutes les entités ont pas été géré) //et qu'un zombie a pas été détecté pour les
|  |  |                                                  humains
|  |  |  Selon (Le type d'entité)
|  |  |  |  cas humain:
|  |  |  |     On stocke la case;
|  |  |  |  cas zombie:  //uniquement pour les humains
|  |  |  |     On stocke la case;
|  |  |  |  cas arme:  // uniquement pour les humains
|  |  |  |     On stocke la case;
|  |  |  fin selon;
|  |  fait;
|  fsi;
fait;
Si rien n'a été trouvé /
|  stockage des cases à analyser pour la perception primaire;
|  //analyse des cases de la perception visuelle primaire
|  Tant que (il reste des cases à analyser)  //et qu'un zombie a pas été détecté pour les humains
|  |  tirage d'une case aléatoire
|  |  Si ( la case est non vide)
|  |  |  Tant que ( toutes les entités ont pas été géré) //et qu'un zombie a pas été détecté pour les
|  |  |  |                                                  humains
|  |  |  |  Selon (Le type d'entité)
|  |  |  |  |  cas humain:
```

| | | | | On stocke la case;
| | | | | cas zombie:
| | | | | On stocke la case;
| | | | | cas arme:
| | | | | On stocke la case;
| | | | fin selon;
| | | fait;
| | fsi;
| fait;
| Si rien n'a été trouvé // uniquement pour les perceptions visuelles et olfactives
| | stockage des cases à analyser pour la perception secondaire;
| | //analyse des cases de la perception visuelle secondaire
| | Tant que (il reste des cases à analyser) //et qu'un zombie a pas été détecté pour les humains
| | | tirage d'une case aléatoire
| | | Si (la case est non vide)
| | | | Tant que (toutes les entités ont pas été géré) //et qu'un zombie a pas été détecté pour les humains
| | | | | Selon (Le type d'entité)
| | | | | | cas humain:
| | | | | | On stocke la case;
| | | | | | cas zombie:
| | | | | | On stocke la case;
| | | | | | cas arme:
| | | | | | On stocke la case;
| | | | | fin selon;
| | | | fait;
| | | fsi;
| | fait;
| fsi
fsi
Si (entité trouve) // ou humain trouvé pour les zombies
| on déplace l'agent selon le type et le placement de l'entité trouvé
Sinon
| on déplace l'agent aléatoirement;
fsi;

2°) Gestion du monde

Pour pouvoir gérer plusieurs entité sur la même case, on décidé de prendre une matrice de list (STL) déclaré comme ceci.

```
list<Entite*> ** _map;
```

Donc, avec cette implémentation, le déplacement d'un agent utilisera les méthodes remove et push_back afin de s'enlever et de se rajouter dans une case du monde.

Et pour rendre le monde torique, on a implémenté une méthode qui prend une référence sur une instance de Position, qui vérifie si les cordonnées sont situé dans le monde et si cela n'est pas le cas corrige les cordonnées.

```
void ConvertisseurCase(Position & atest);
```

3°) Gestion d'un tour de simulation :

l'ordre dans lequel on choisit l'agent à simuler est fait aléatoirement, de façon à pas fausser la simulation

algorithme d'un tour de simulation:

On transforme la liste d'agent en tableau.
Tant que (il reste des agents à traiter)
| On tire un agent aléatoirement sans remise
| Si l'agent est encore actif //on teste si un agent précèdent à pas tuer l'agent à actionner
| | Tour de l'agent
| fsi
fait

4°) Gestion des combats :

L'issue du combat n'est pas déterministe, c'est à dire qu'il y a une part d'aléatoire, pour déterminer le vainqueur.

Algorithme du combat:

tirage aléatoire du multiplicateur de force humain entre 8 et 12
tirage aléatoire du multiplicateur de force du zombie entre 8 et 12
Si (force du zombie * multiplicateur) > (force de l'humain * multiplicateur)
| //zombie gagnant
| Tirage d'un nombre entre 0 et 100
| Si ce nombre est inférieur à 70
| | L'humain est contaminé et un nouveaux zombie est crée
| fsi
Sinon
| //humain gagnant.
fsi

IV Résultats

Avant de réaliser une série de test, voici un aperçu du simulateur.

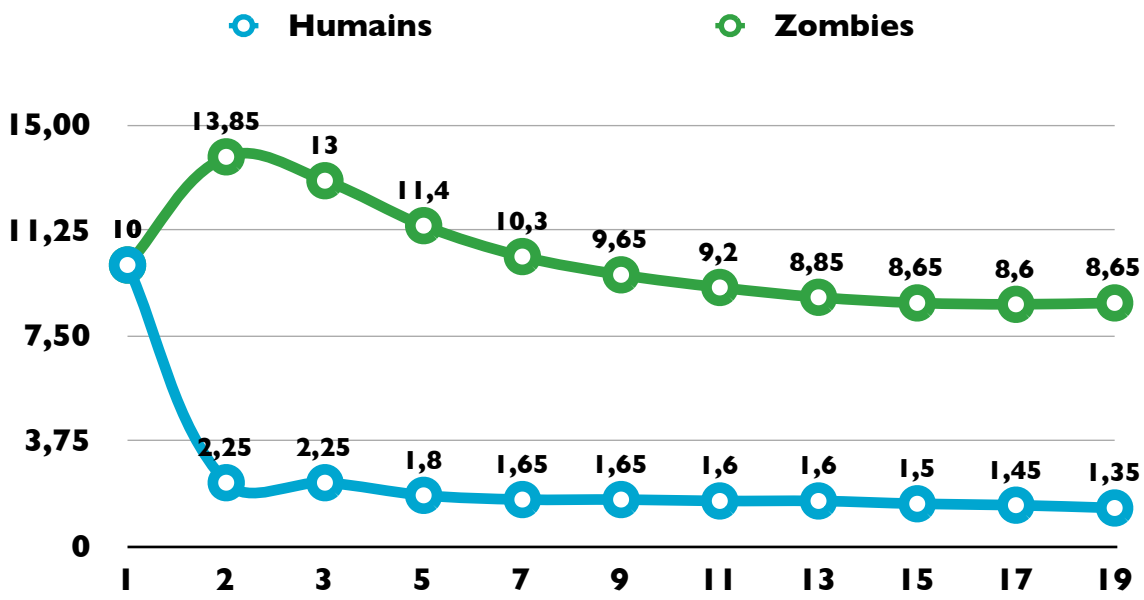
screenshot du simulateur en pleine action

Nos tests se passeront en 3 parties et subiront le même protocole, c'est à dire 20 simulations sans réinitialisation du générateur et les résultats obtenues seront la moyenne du nombre de zombie et du nombre d'humain par tour de simulation.

La première simulera un mode sans-défense des humains, c'est à dire il y aura pas d'arme dans cette simulation.

Caractéristiques de la simulation:

Nombre d'humain initial : 10
Nombre de zombie initial : 10
Nombre d'arme initiale : 0
Taille de zone de simulation : 10x10

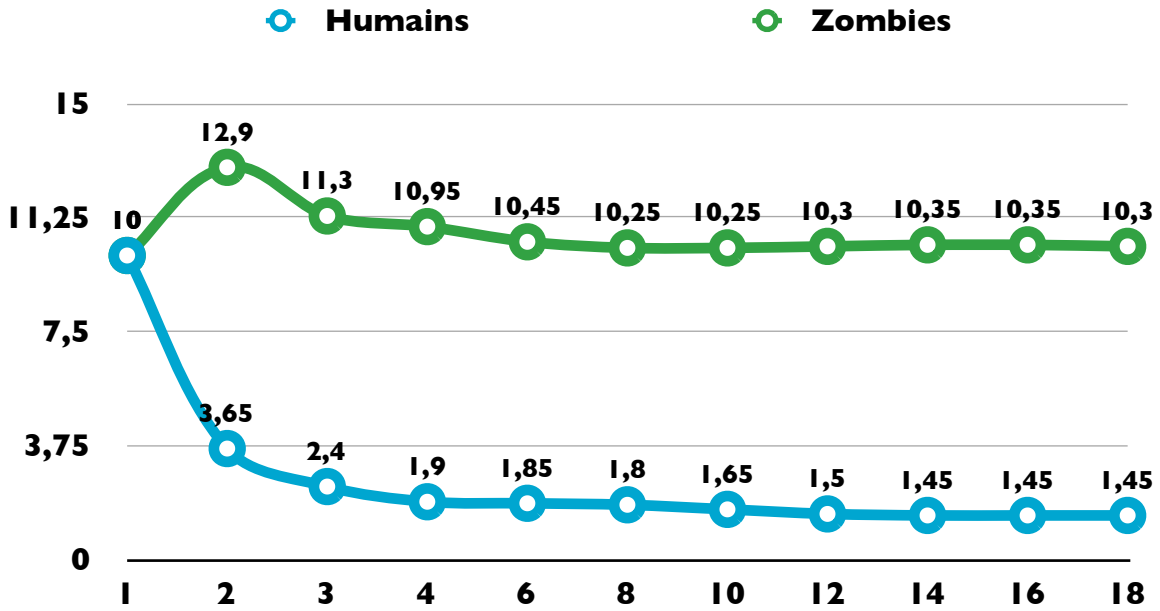


Conclusion de cette simulation : Les zombies ont été modélisés plus fort que les humains à l'état naturel, on le vérifie avec ces chiffres, mais ce qui est intéressant à observer c'est la persistance d'un spécimen humain en moyenne, qui est lié avec la baisse de la population des zombie. On peut expliquer ce phénomène par la présence d'un surhomme qui a une probabilité de vaincre des zombies plus facilement et survivre. Un exemple de sélection naturelle?

La deuxième simulera un monde urbain, c'est à dire une grande promiscuité entre les agents .

Caractéristiques de la simulation:

Nombre d'humain initial : 10
Nombre de zombie initial : 10
Nombre d'arme initiale : 5
Taille de zone de simulation : 8x8

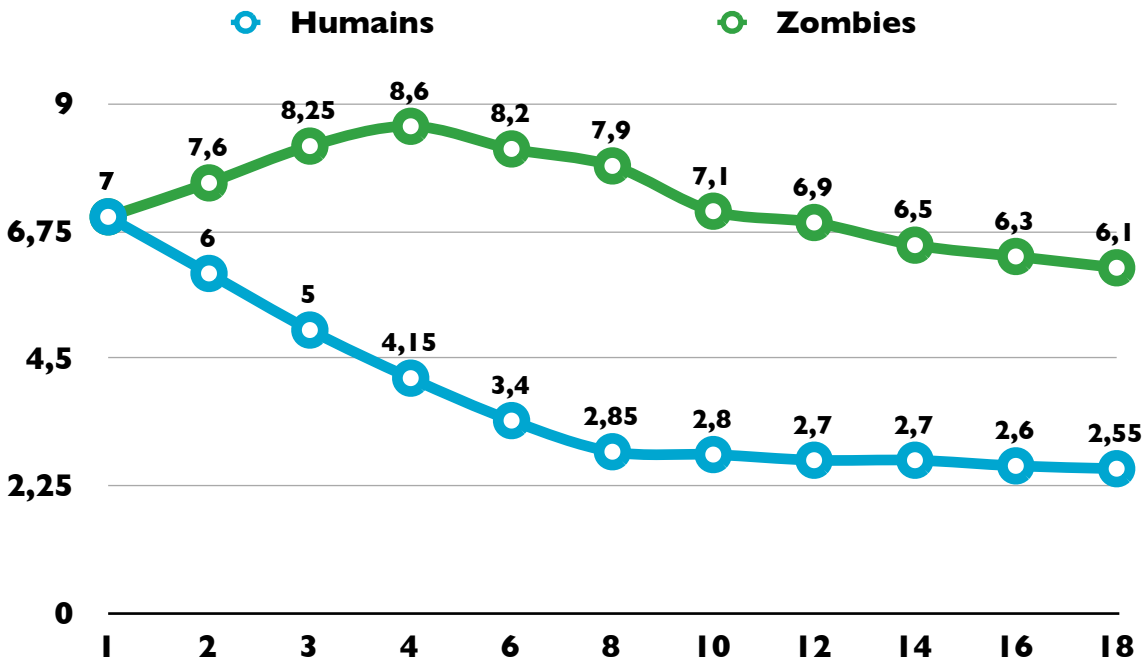


Conclusion de cette simulation : Comme la simulation précédente, les zombies sortent vainqueur de la confrontation, on peut expliquer cela par le fait que les confrontations sont plus fréquente que sur un monde plus grand or les zombies sont plus fort donc on peut expliquer la chute brutale de la population humaine lors des premiers tours de simulation. La présence d'arme facilite la présence du phénomène de surhomme vue dans la simulation précédente car même si la fréquence des combat est supérieure, il est plus fort pour les affronter.

La troisième simulera un monde rural, c'est à dire une faible concentration d'agent pour une zone donnée .

Caractéristiques de la simulation:

Nombre d'humain initial : 7
Nombre de zombie initial : 7
Nombre d'arme initiale : 3
Taille de zone de simulation : 15x15



Conclusion de cette simulation : C'est encore les zombies qui sortent vainqueur de la simulation mais on remarque que le taux de survie des humains est supérieur aux précédents. les faibles confrontations ainsi que le nombre d'arme initial par rapport à la population humaine font en sorte qu'ils survivent mieux.

V Conclusions

Pour faire le bilan de ce qui a été effectué on peut dire que la plus grande partie de ce mini-projet a été la modélisation du problème, du choix du thème à la conception des classes aux retours de problème lors du développement.

La deuxième partie, le code à proprement parler, la difficulté a été de transposer le modèle en code C++, notamment pour la gestion du stockage des agents.

La troisième et dernière partie a été la partie de test et d'évaluation du modèle, a permis d'avoir des observations et de nouvelles remarques pour faire évoluer le modèle ainsi que la façon de percevoir des agents

On pourra conclure en fonction des résultats obtenues lors de ces simulations, qu'en cas de prolifération de mort-vivant, on conseillera à nos proches, qu'il vaudra mieux pour eux de fuir les espaces urbains et aller à la campagne.

Cependant, on pourra proposer des améliorations au modèle choisi:

-La gestion d'un groupe de survivant, ou de zombie
-La réutilisation des armes des humains décédés
-Une amélioration des algorithmes de perceptions (en terme de conception logicielle)
-Dès une détection d'un humain par un zombie, celui-ci se focalise sur la victime en la poursuivant.