Flinesterflure Projet

---

0 : Si Le joueur n°2 possède un pion au pattern 1 (actif) ^ (¬ gagnant)
Le joueur n°2 n'a pas sélectionné de pions

Alors Le joueur n°2 sélectionne ce pion
—> Appel Récursif en supprimant la règle 0

---

1 : Si Le joueur n°2 possède un pion au pattern 2 (actif) ^ (¬ gagnant)
Le joueur n°2 n'a pas sélectionné de pions

Alors Le joueur n°2 sélectionne ce pion
—> Appel Récursif en supprimant la règle 1

---

2 : Si Le joueur n°2 possède un pion au pattern 3 (actif) ^ (¬ gagnant)
Le joueur n°2 n'a pas sélectionné de pions

Alors Le joueur n°2 sélectionne ce pion
—> Appel Récursif en supprimant la règle 2

---

3 : Si Le joueur n°2 possède un pion au pattern 4 (actif) ^ (¬ gagnant)
Le joueur n°2 n'a pas sélectionné de pions

Alors Le joueur n°2 sélectionne ce pion
—> Appel Récursif en supprimant la règle 3

---

4 : Si Le joueur n°2 possède un pion au pattern 5 (actif) ^ (¬ gagnant)
Le joueur n°2 n'a pas sélectionné de pions

Alors Le joueur n°2 sélectionne ce pion
—> Appel Récursif en supprimant la règle 4

---

5 : Si Le joueur n°2 possède un pion au pattern 6 (actif) ^ (¬ gagnant)
Le joueur n°2 n'a pas sélectionné de pions

Alors Le joueur n°2 sélectionne ce pion
—> Appel Récursif en supprimant la règle 5

---

6 : Si Le joueur n°2 a sélectionné un pion (actif) ^ (¬ hors-plateau) ^ (¬ gagnant)
Le pion sélectionné est sur la case « sortie » et possède au moins 1 mouvement

Alors Le pion sélectionné est (¬ actif) ^ (hors-plateau) ^ (gagnant)
Le joueur n°2 n'a pas sélectionné de pions
La case de destination est « null »
—> **Pas d'Appel Récursif !**

---

7 : Si Le joueur n°2 a sélectionné un pion (actif) ^ (hors-plateau) ^ (¬ gagnant)
Le pion sélectionné possède au moins 1 mouvement
La case de destination est « null »

Alors La case de destination est la case « entrée ; 3 »
—> Appel Récursif en supprimant la règle 7

---

8 : Si Le joueur n°2 a sélectionné un pion (actif) ^ (¬ hors-plateau) ^ (¬ gagnant)
Le pion sélectionné peut se déplacer vers le Nord
La case de destination est « null »

Alors La case de destination est la case « Nord ; 0 »
—> Appel Récursif **SANS** supprimer la règle 8

---

9 : Si Le joueur n°2 a sélectionné un pion (actif) ^ (¬ hors-plateau) ^ (¬ gagnant)
Le pion sélectionné peut se déplacer vers l'Ouest
La case de destination est « null »

Alors La case de destination est la case « Ouest ; 3 »
—> Appel Récursif **SANS** supprimer la règle 9

---

10 : Si Le joueur n°2 a sélectionné un pion (actif) ^ (¬ hors-plateau) ^ (¬ gagnant)
Le pion sélectionné peut se déplacer vers le Sud
La case de destination est « null »

Alors La case de destination est la case « Sud ; 2 »
—> Appel Récursif **SANS** supprimer la règle 10

---

11 : Si Le joueur n°2 a sélectionné un pion (actif) ^ (¬ hors-plateau) ^ (¬ gagnant)
Le pion sélectionné peut se déplacer vers l'Est
La case de destination est « null »

Alors La case de destination est la case « Est ; 1 »
—> Appel Récursif **SANS** supprimer la règle 11

---

12 : Si Le joueur n°2 a sélectionné un pion (actif) ^ (¬ hors-plateau) ^ (¬ gagnant)
La distance entre la case sélectionnée et le monstre, est strictement supérieure à 5
La case du pion sélectionnée ne contient pas de tokens
La case du pion sélectionné n'est pas dans l'axe du monstre
La case de destination est « null »

Alors Le pion sélectionné est (¬ actif) ^ (¬ hors-plateau) ^ (¬ gagnant)
La case de destination est « null »
Le joueur n°2 n'a pas sélectionné de pions
—> **Pas d'Appel Récursif !**

13 : Si Le joueur n°2 a sélectionné un pion (actif) ^ (¬ hors-plateau) ^ (¬ gagnant)
La case de destination n'est pas « null »
La distance entre le pion sélectionné et la case « sortie » est égale à X
Le nombre de mouvements du pion sélectionné est supérieur ou égale à X + 1
Le pion sélectionné n'est pas dans l'axe du monstre
Le monstre n'est pas sur la case « sortie »

Alors Le pion se déplace en fonction de la direction de la case de destination
La case de destination est « null »
—> Appel Récursif **SANS** supprimer la règle 13

14 : Si Le joueur n°2 a sélectionné un pion (actif) ^ (¬ hors-plateau) ^ (¬ gagnant)
La case de destination n'est pas « null »
La case de destination est dans l'axe du monstre
La distance entre la case de destination et le monstre est égale à celle d'un pion qui est (¬ hors-plateau) ^ (¬ gagnant)
La case de destination est devant le monstre

Alors Le pion se déplace en fonction de la direction de la case de destination
La case de destination est « null »
—> Appel Récursif **SANS** supprimer la règle 14

15 : Si Le joueur n°2 a sélectionné un pion (actif) ^ (¬ hors-plateau) ^ (¬ gagnant)
La case de destination n'est pas « null »
La case de destination est dans l'axe du monstre
Le nombre de mouvements du pion sélectionné est strictement supérieur à 1

Alors Le pion se déplace en fonction de la direction de la case de destination
La case de destination est « null »
—> Appel Récursif **SANS** supprimer la règle 15

16 : Si Le joueur n°2 a sélectionné un pion (actif) ^ (¬ hors-plateau) ^ (¬ gagnant)
La case de destination n'est pas « null »
La case de destination n'est pas dans l'axe du monstre
La case de destination n'est pas devant le monstre
La direction de la case de destination n'est pas égale à l'orientation du monstre

Alors Le pion se déplace en fonction de la direction de la case de destination
La case de destination est « null »
—> Appel Récursif **SANS** supprimer la règle 16

---

17 : Si Le joueur n°2 a sélectionné un pion (actif) ^ (¬ hors-plateau) ^ (¬ gagnant)
La case de destination n'est pas « null »
La distance entre la case de destination et le monstre, est strictement supérieure à 5
La case de destination n'est pas dans l'axe du monstre
La case de destination est devant le monstre
La distance entre la case de destination et le monstre est supérieur ou égale à celle de la case du pion sélectionné

Alors Le pion se déplace en fonction de la direction de la case de destination
La case de destination est « null »
—> Appel Récursif **SANS** supprimer la règle 17

---

18 : Si Le joueur n°2 a sélectionné un pion (actif) ^ (¬ hors-plateau) ^ (¬ gagnant)
La case de destination n'est pas « null »
La distance entre la case case de destination et le monstre, est inférieure ou égale à 5
La case de destination n'est pas dans l'axe du monstre
La distance entre la case de destination et le monstre est supérieure ou égale à celle de la case du pion sélectionné

Alors Le pion se déplace en fonction de la direction de la case sélectionnée
La case de destination est « null »
—> Appel Récursif **SANS** supprimer la règle 18

---

19 : Si Le joueur n°2 a sélectionné un pion (actif) ^ (¬ hors-plateau) ^ (¬ gagnant)
La case de destination n'est pas « null »
La distance entre la case de destination et le monstre, est inférieure ou égale à 5
La case de destination n'est pas dans l'axe du monstre
La distance entre la case de destination et le monstre est inférieure à celle de la case du pion sélectionné
Le nombre de mouvements du pion sélectionné est strictement supérieure à 1

Alors Le pion se déplace en fonction de la direction de la case sélectionnée
La case de destination est « null »
—> Appel Récursif **SANS** supprimer la règle 19

Class

Variable ici !

```
méthode(liste de nombres)
{
	for( e : listes )
	{
		if( condition)
		{
			conséquence
			méthode(liste de nombres)
		}

		if( condition)
		{
			conséquence
			méthode(liste de nombres)
		}
	}
}
```

Note : « actif » <=> « le nombre de mouvements du pion sélectionné n'est pas égale à 0 »

// Debut du tour du joueur n°2
Class.method(liste de nombres)
// Fin du tour du joueur n°2
(donc pensez au changement de pattern après la fin)

Code-Couleur :
Sélection des pions
Gestion de directions particulières
Gestion des directions possibles
Passer son tour
Foncer vers la sortie si possible
Prise de sécurité
Gestion du monstre à longue portée
Gestion du monstre à courte portée

IA :

Le projet ici présent est un jeu de société : il regroupe un objectif simple avec un peu de stratégie. Pour concevoir l'intelligence artficielle, une approche symbolique a été donc nécessaire.
Tout d'abord, on résume ce qui constitue le jeu de société : une sortie, une entrée, des pions, un monstre, des bloc de pierre, des tâches de sang, et des murs. On peut alors conclure les constats suivants : un objectif fixe, une gestion des déplacements pour chaque pion, une gestion des pions, une gestion de la vision du monstre et de sa distance.
L'objectif étant fixe et simple, le jeu se résume à des suites de mécaniques qui se répètent : choisir un pion, le déplacer case par case, en prenant en compte ce qu'il l'entoure ainsi que l'objectif, recommencer, etc. De ce fait, les seuls choix se résument aux possibilités de déplacement de chaque pion : aller à nord, à l'ouest, au sud, à l'est, ou rester. Au total, on a donc 5 choix, multipliés par le nombre de pions par joueurs (4) : 20. Mais surtout, il y a donc peu d'euristiques : en plus de seulement 5 choix, ces choix sont quasi-automatiques puisqu'elles ont comme objectif, un seul endroit (la sortie) et donc une direction bien défini (Nord et Ouest).
De ce fait, je sais qu'un parcours en profondeur sera approprié pour ce genre de situation.
L'euristique étant basé sur 5 choix par pion, le reste étant une succession mécanique de choix et d'actions, les priorités seront alors basés sur l'ordre des actions à prendre. Autrement dit, outre la stratégie certes présentes mais faibles, les priorités seront avant tout pragmatiques.

Après avoir donc étudié ces propriétés, je fais l'approche symbolique de l'IA, par la rédaction de règles : chaque règle est constitué de conditions précises suivies des conséquences à engendrer après. Il y au total 20 règles.
Les 5 premières règles gèrent la sélection du pion, dans l'ordre croissant de leur nombre de déplacements. En effet, un pion avec peu de mouvement est plus vulnérable qu'un pion en ayant plus. Une fois, le pion sélectionné, les possibilités de déplacements sont ensuite gérées.
Cette gestion commence d'abords avec les 2 règles suivantes, qui traite les cas particuliers : faire sortir un pion ou le rentrer en jeu immédiatement. Bien sûr, les 5 règles d'après, traitent les 5 choix du pion pour se déplacer (voir si dessus).
Enfin, pour chacun de ces choix, les autres règles estimeront le danger : 1 règle concerne les pions proche de la sortie ; 3 autres concernent les prises de risque du pion lors de ses déplacements, vis-à-vis du monstre (exemple : ne pas se retrouver vue par le monstre, s'il n'est pas possible de lui échapper après) ; 3 dernières concernent la possibilité au pion d'échapper le plus possible du monstre, en cas de danger.

L'ensemble des règles étant structurés, chacune par des conditions puis une action, le schéma de conception est ainsi simple : un algorythme n'a pas été nécessaire ; de plus, la majeur partie des opérateurs sont déjà présents dans le code à l'origine. Le codage de l'IA s'est donc résumé à la création de booléan, ainsi que quelques nouveaux opérateurs exclusifs au fonctionnement de l'IA (ainsi que de nouvelles varibles, toujours en rapport avec les règles établies).

La conception des règles aura pris le plus de temps : il y avait beaucoup de mécaniques prises en compte, leur ordre, mais aussi la notion de récursivités. En effet, la quasi-totalité des règles vont provoquer de la récursivité. Il a été donc nécessaire de savoir, quelles variables vont alors être passé par référence ou non, mais aussi la gestion des règles afin d'éviter de répéter inutilement certaines. Il n'était pas impossible d'améliorer l'IA, en faisant un parcours en largeur et une liste contenant les choix retenus : les pions ne se sauraient pas déplacer par choix de cases, mais par choix de trajets. Toutefois, concevoir les règles et l'IA a pris beaucoup de temps, et cette amélioration bien qu'intéressante, ne pouvait pas être fait dans les temps. Toutefois, l'IA qui a été crée, n'est pas absurde non plus, étant donné que l'objectif était simple mais surtour fixe : une telle IA fonctionne pour le projet, mais n'aurait pas été du tout pratique pour un jeu d'échec par exemple (l'objectif étant difficile à atteindre mais également mobile). De plus, l'handicap du concepteur de l'IA (Aurélien Herbin) a rendu flou les possibilités de piéger ses adversaires : l'IA est alors simple et n'est pas spécialisé pour piéger un joueur, mais lui permet d'en échapper au minimum.

En conclusion, l'IA a été conçu entièrement et les seuls problèmes pouvant être occasionnés ne peuvent qu'être des prises de décision étranges et donc involontaires par le concepteur.

Token :

Il s'agit des pions qui consituent le jeu. On en compte 3 : les blocs de pierres (TokenR), les pions (TokenP) et le monstre (TokenM). Tous partagent des points communs élémentaires : ils ont chacun des coordonnées et peuvent se déplacer. Concernant le déplacement, ses propriétés varient en fonction du token. La propriété la plus importante étant la gestion des murs : si le monstre ignore les murs (ce qui nécessite une gestion des bordures des terrains, exclusive à lui), sa vision en est dépendante ; les pions et les blocs avant de gérer les bordures de terrains, gèrent avant tout les murs (il n'y a donc pas besoin de gestions de bordures, pour les TokenP et les TokenR). De plus, les TokenP et les TokenR se retrouvent détruits au pire, s'ils sont forcé par le TokenM.

Les difficultés rencontrées ont été la gestion de cas très particuliers tels que les flaques d'hémoglobines, voire d'autres qui pourraient avoir été négligé accidentellement étant donné la rareté de ces cas. Les problèmes que peuvent être provoqués les Token sont donc des bogues. Il a été aussi nécessaire de vérifier aussi que les Tokens avaient été correctement instancié afin d'éviter d'autres bogues plus vicieux. Sinon, certains d'entre-eux ont pu être révélé lors de la programmation et des testes : les corriger ont pris alors beaucoup de temps (notamment un bogue qui a pris une semaine).