Bordeaux INP
ENSC

# Intelligence artificielle

## Projet Prolog 2020 - 10 minutes to kill

---

Encadrant :
Pierre-Alexandre FAVIER


Yann Coussemacker - Nina Franiatte - Léa Le Moal
Promo 2021

# Sommaire

# Introduction

Ce projet a été effectué dans le cadre du module d'Intelligence Artificielle et afin d'exploiter le langage de programmation logique qu'est Prolog. Il nous a alors été demandé de livrer une version informatisée du jeu de société *10 minutes to kill programmée* sous Prolog. Ce langage propose une approche différente des autres langages de programmation  orientés objets auxquels nous sommes habitués (en C ou C# par exemple) puisqu'il répond à un raisonnement logique.

Prolog peut être assimilé à une des formes que peut prendre l'intelligence artificielle, dans le sens où nous allons programmer la machine pour qu'elle soit "apte" à résoudre certains problèmes. En outre, elle doit être capable d'interagir avec un utilisateur extérieur, de réagir et de s'adapter en fonction des coups que propose ce dernier.

L'enjeux principal de ce projet a été de transposer les règles du jeu de société en un système logique par l'intermédiaire de prédicats. Ainsi, l'accent n'a pas été mis sur l'affichage graphique et l'utilisabilité du jeu. Néanmoins il est évidemment possible pour un utilisateur de comprendre l'avancée du jeu malgré l'absence d'éléments graphiques. Ainsi, le présent rapport a pour visées à la fois de de guider le correcteur, et d'apporter des précisions et des justifications concernant nos choix de conception qui nous ont permis d'atteindre notre objectif final.

# Choix de conception

## Exigences fonctionnelles

Pour coder notre jeu, nous avons fait le choix d'une configuration à deux joueurs : un utilisateur extérieur face à l'ordinateur. Les seuls changements de règles dans une partie à deux joueurs concernent le nombre de policiers (trois à la place de deux) et l'alternance des tours : une fois au tour de l'utilisateur et une fois au tour de l'ordinateur.

Le tableau des exigences fonctionnelles ci-dessous résume l'ensemble des faits et actions qui construisent notre jeu :

| Code | Description |
|---|---|
| EF_01 | Seize personnages portant des noms d'animaux ont été déclarés. Ils ont tous des rôles (tueur, cible ou inutile) qui leurs sont attribués. |
| EF_02 | Il y a 3 policiers en dehors du plateau, qui pourront être utilisés au cours de la partie. |
| EF_03 | Les personnages peuvent être déplacés sur un plateau de dimension 4x4 cases sans trou (contenant 16 tuiles au total). |
| EF_04 | Certaines cases choisies préalablement correspondent aux tuiles "sniper" sur lesquelles il faut se trouver pour utiliser l'attaque fusil. |
| EF_05 | Le joueur qui commence est tiré de manière aléatoire. |
| EF_06 | Le tueur et les cibles sont attribuées à chaque joueur. |
| EF_07 | Chaque joueur peut déplacer, contrôler ou tuer un personnage (avec une des trois attaques : couteau, pistolet, fusil) et avec un maximum de deux actions par tour. |
| EF_08 | Si un personnage est tué, il disparaît de la liste des personnages en jeu. |
| EF_09 | Les deux joueurs jouent chacun leur tour. |
| EF_10 | Les deux joueurs ont tous deux un score qui évolue en fonction des coups joués. |
| EF_11 | Le joueur qui a le plus de points est déclaré vainqueur. |

# Exigences techniques

## Éléments de jeu

Pour configurer les éléments de base qui constituent le jeu, certaines déclarations ont été faites via des manipulations dynamiques, permettant à nos programmes de se modifier au cours du jeu en ajoutant ou supprimant des clauses. Ces manipulations dynamiques concernent les tuiles, les différents rôles tels que tueur, cible ou inutile, les personnages morts et vivants, les tours et les points.

Dans un premier temps, nous avons fait le choix d'établir un plateau de jeu fixe, de dimension 4x4 (qui comporte donc 16 tuiles différentes) et qui pourrait être représenté graphiquement tel que suit. Les cases avec une croix 'X' sont les cases correspondantes aux tuiles "viseur" permettant d'utiliser le fusil.

| | | | |
|---|---|---|---|
| X | | | X |
| | X | X | |
| | X | X | |
| X | | | X |

**Plateau de jeu**

Le choix d'établir un plateau fixe a été fait par soucis de simplification des règles du jeu. Une exigence secondaire pourrait être de créer un plateau avec une répartition des tuiles aléatoire, ainsi le plateau ne serait jamais le même d'une partie à une autre.

Les personnages ont été pensés sous forme de listes. En effet, s'ils sont tous attribués au début de la partie, certains vont être amenés à disparaître au cours du jeu. Ainsi, afin d'éviter de retrouver un personnage en double, nous avons créé une liste de personnages dédiée à ceux encore vivants (nommée *vivants*) et une liste de personnages qui comprend ceux éliminés (nommée *mort*).
L'utilisation de listes s'est révélée très pratique, notamment pour attribuer les cartes de rôle (tueur et cibles) à un joueur de manière aléatoire ; nous avons pu "piocher" dans ses listes avec la fonction *random*.

De plus, au début de la partie, l'ensemble des données sont initialisées. Plus précisément :

- Au départ tous les personnages sont considérés comme morts, et lors de l'initialisation on les déplace dans la liste des personnages vivants en leur attribuant à chacun une tuile correspondant à leur position initiale ;
- Le joueur qui débute est désigné de manière aléatoire à l'aide de la fonction *random* ;
- Les scores des deux joueurs (utilisateur et ordinateur) sont initialisés à 2 ;

La partie peut alors commencer !

Le système de tour permettant de passer d'un joueur à un autre a aussi été codé par méthode dynamique, à l'aide des fonctions *retract* et *assert* qu'offre Prolog.
A chaque tour le joueur choisit l'action qu'il souhaite réaliser (au maximum il en enchaîne deux), les données du jeu se mettent à jour et c'est ensuite au tour de l'adversaire.
La partie se finit lorsqu'un tueur à gage a éliminé toutes ses victimes ou lorsque les deux tueurs à gage sont démasqués.

## Actions du jeu

Pour coder les déplacements, nous avons raisonné en terme de tuiles comportant chacune un ou plusieurs personnages. En effet, nous avons décidé que chaque tuile serait rattachée à la liste de personnages qu'elle contient à l'instant T. Concernant le code, cela signifie que nous avons créé une liste (*listePerso*) propre à chaque tuile, et qu'à chacun des déplacements, nous avons à la fois ajouté le personnage au début de la liste de la tuile sur laquelle il s'est déplacé, et supprimé ce même personnage de sa tuile d'origine.

Concernant les actions qui permettent à un joueur de tuer, nous avons fonctionné en deux temps. D'abord nous avons vérifié que le personnage P1 qui souhaitait tuer le personnage P2, avec l'une des trois attaques disponibles, était bien en mesure de le faire (par exemple on ne peut tuer qu'avec son tueur à gage). Cette vérification est réalisée grâce au prédicat *peutTuer*. Dans un second temps, si les conditions sont vérifiées, le personnage peut effectuer son action grâce au prédicat *Tuer*. Les listes de personnages interviennent également ici : chaque personnage éliminé étant retiré de la liste des vivants pour aller se placer dans celle des morts.

Enfin, dans les actions, nous avons aussi établi le prédicat *donnePoints*, qui permet (comme son nom l'indique…) d'établir un score pour les deux joueurs. Compter les points est certes dans les règles, sinon le jeu perdrait un peu de son intérêt, cependant cela permet aussi d'établir un vainqueur à la fin de la partie (prédicat *vainqueur*).

### Affichage

Grâce  à la commande “plateau(X,Y,L)”, il est possible d’afficher la liste des personnages L sur chaque tuile de coordonnées (X,Y). Pour afficher l’entièreté du plateau il faut faire défiler les résultats proposés par prolog en appuyant sur la touche “;”.

Nous avons choisis d’afficher quelques instructions et informations sous forme de texte après certaines requêtes. par exemple il était intéressant de savoir automatiquement qui commençait à jouer sans avoir à le demander explicitement à prolog. Nous avons de même affiché le résultat de chaque action, c’est-à-dire le statut du personnage tué ou contrôlé et si un déplacement avait bien eu lieu. Ces affichages était obligatoires car les instructions utilisées pour les actions renvoient toujours “false” si le joueur actuel doit encore faire une action. Or cet affichage de “false” pouvait laisser soupçonner que l’action n’avait pu se faire alors que si.

Enfin, nous avons choisis de réaliser des prédicats tels que “cibles()”, “tueur()”, “points()” qui ne prennent aucun paramètre et affichent respectivement les cibles, le tueur et les points du joueur en cours. Ces prédicats permettent une navigation plus rapide et intuitive.

# Notice d’utilisation

Tout ce qui est écrit en gras entre guillemets est à recopier dans la console tel quel, si le contenu d’une quote est en italique il faut changer le texte en italique comme indiqué.

## 1. Lancer la partie

Consulter le fichier source (“Coussemacker_Franiatte_LeMoal_Projet.pl”)
Pour lancer la partie il faut écrire dans la console prolog “debutPartie().”
Une personne physique incarnera “***utilisateur***” et une autre personne physique incarnera “***ordi***”.

La personne qui commence à jouer est tirée au hasard, Puis son nom est affiché.

## 2. Faire un tour

### a) Consulter les informations

Vous pouvez consulter rapidement plusieurs de vos données personnelles lors de votre tour :

- Votre tueur : “**tueur()**”
- Vos cibles : “**cibles()**”

- Vos points : "**points()**".

Vous pouvez aussi afficher les données communes à votre adversaire tel que :

- Le plateau : "**plateau(*CoordoneeX*,*CoordoneeY*,*ListePersonnages*)**". Après avoir entré cette commande vous devrez appuyer sur la touche ";" pour faire apparaître le contenu des autres tuiles.
- La liste des personnages vivants : "**vivants(ListePersoVivants)**".
- La liste des personnages morts et arrêtés : "**morts(ListePersoMorts)**".

## b) Agir

Lors d'un tour, vous pouvez agir 2 fois au total. Parmis ces deux actions vous pouvez ***contrôler un personnage*** et vous ***déplacer*** autant de fois que vous le souhaitez mais vous ne pourrez ***tuer*** qu'une seule fois par tour.

### *Deplacer :*

Il n'y a pas de contraintes pour déplacer un personnage. Il suffit de rentrer "**deplacer(***Personnage***,** *X*,*Y***).**" dans la console prologue avec *Personnage* qui correspond au personnage que vous souhaitez déplacer (n'importe lequel sur le plateau). *X*, *Y* correspondent aux lignes et colonnes de la tuile sur laquelle vous voulez déplacer le personnage. *X* et *Y* doivent être compris entre 1 et 4 inclus.

### *Contrôler :*

Il n'y a pas de contraintes pour contrôler un personnage. Il suffit de rentrer "**controle(***Personnage***).**" dans la console prologue avec *Personnage* qui correspond au personnage que vous souhaitez contrôler (n'importe lequel sur le plateau). Le but de cette fonctionnalité est d'arrêter le tueur à gage adverse. C'est le seul cas où cette action vous dira que vous n'avez pas arrêté un innocent.

### *Tuer :*

Dans ce jeu il existe 3 manières de tuer : avec le couteau, le pistolet ou le fusil. Chacune de ces armes est soumise à des restrictions d'utilisation bien précises et différentes pour chaque. C'est pourquoi le jeu pourra vous proposer à chaque fois quels sont les personnages que vous pouvez tuer avec votre tueur à gage. Pour consulter toutes ces possibilités de meurtres il faudra taper dans la console prolog "**peutTuer(Perso,X,Y)**". Cette fonction vous affichera alors un personnage **Perso** qui se trouve en tuile de coordonnées (**X,Y**) que vous pouvez tuer. En appuyant sur la touche ";" après avoir entré la commande ci-dessus vous verrez apparaître un par un tous les personnages que vous pouvez tuer avec votre tueur à gage.
Enfin, une fois que vous avez choisi le personnage que vous voulez assassiner, il vous suffit d'entrer la commande "**tue(***Personnage***).**" avec *Personnage* correspondant au nom du personnage que vous voulez tuer.

A la fin de chaque action vous recevrez un message rendant compte de l'action que vous venez de réaliser.

Une fois que vous avez réalisé vos deux actions du tour, le programme vous notifie que votre tour est terminé : vous devez laisser place à l'autre joueur.

## 3. Fin de la partie

La partie se termine dès lors que les deux tueurs à gage ont étés démasqués et/ou tués ou dès qu'un des deux joueurs a éliminé toutes ses cibles.
Vous pourrez consulter les scores finaux en entrant la commande suivante : "**points(Joueur, Points).**" Le score d'un des joueurs sera affiché, il faudra appuyer sur la touche ";" pour afficher le score de l'autre joueur.

## 4. Partie type

```
% c:/Users/Yann/Documents/Cours/IA/projet_prolog/projet.pl compiled 0.00 sec, 90 clauses
?- debutPartie().
utilisateur qui commence.
true.

?- tueur().
tueur : canard
true.

?- cibles().
Cibles : [souris,chien,oie]
true.

?- plateau(X,Y,LP).
X = Y, Y = 4,
LP = [crocodile] ;
X = 4,
Y = 3,
LP = [chien] ;
X = 4,
Y = 2,
LP = [souris] ;
X = 4,
Y = 1,
LP = [morse] ;
X = 3,
Y = 4,
LP = [canard] ;
X = Y, Y = 3,
LP = [lion] ;
X = 3,
Y = 2,
LP = [lapin] ;
X = 3,
Y = 1,
LP = [chat] ;
X = 2,
Y = 4,
LP = [elephant] ;
X = 2,
Y = 3,
LP = [oie] ;
X = Y, Y = 2,
LP = [loup] ;
X = 2,
Y = 1,
LP = [poisson] ;
X = 1,
Y = 4,
LP = [girafe] ;
X = 1,
Y = 3,
LP = [serpent] ;
X = 1,
Y = 2,
LP = [loutre] ;
X = Y, Y = 1,
LP = [renard].
```

```
?- peutTuer(P,_,_).
P = lion ;
P = crocodile ;
P = chien ;
P = oie ;
P = elephant ;
false.

?- tuer(oie).
Correct to: "tue(oie)"? yes
Bravo ! Vous avez tuÃ© une de vos cibles. Vous gagnez un point.
false.

?- points().
points : 3
true.

?- tue(chien).
false.

?- controle(lion).
Dommage ! Vous avez contrÃ´lÃ© un innocent. Vous ne gagnez ni ne perdez de points.

Votre tour est terminÃ©.
true.

?- tueur().
tueur : renard
true.

?- cibles().
Cibles : [loup,poisson,elephant]
true.

?- peutTuer(P,_,_).
P = loup ;
P = loutre ;
P = poisson ;
P = morse ;
P = chat ;
P = poisson ;
P = girafe ;
false.

?- deplacer(canard,1,1).
Personnage dÃ©placÃ©.
false.

?- tue(canard).
Bravo ! Vous avez tuÃ© le tueur Ã  gage adverse. Vous gagnez trois point et votre adversair
e perd ses deux points d'avance.

Votre tour est terminÃ©.
true.

?- controle(renard).
Bravo ! Vous avez arrÃªtÃ© le tueur Ã  gage adverse. Vous gagnez un point et votre adversai
re perd ses deux points d'avance.
ordi Ã  gagnÃ©
true.

?- points(Joueur,Points).
Joueur = utilisateur,
Points = 2 ;
Joueur = ordi,
Points = 3.
```

# Conclusion

Ce projet a permis de nous familiariser avec le raisonnement logique de Prolog en l'appliquant sur un système complet qu'est le jeu de rôle 10 minutes to kill.

Néanmoins, afin de rendre le jeu plus agréable et lui apporter une meilleure utilisabilité, il serait indispensable d'apporter une dimension visuelle afin d'offrir à l'utilisateur des retours immédiats et explicites sur l'évolution du jeu. Ainsi, des apports graphiques sont une perspective d'amélioration de ce jeu. Il serait aussi envisageable de programmer une IA permettant à l'utilisateur de jouer seul face à une machine. De plus, il serait notamment préférable d'automatiser certaines commandes afin de rendre le programme plus efficient.