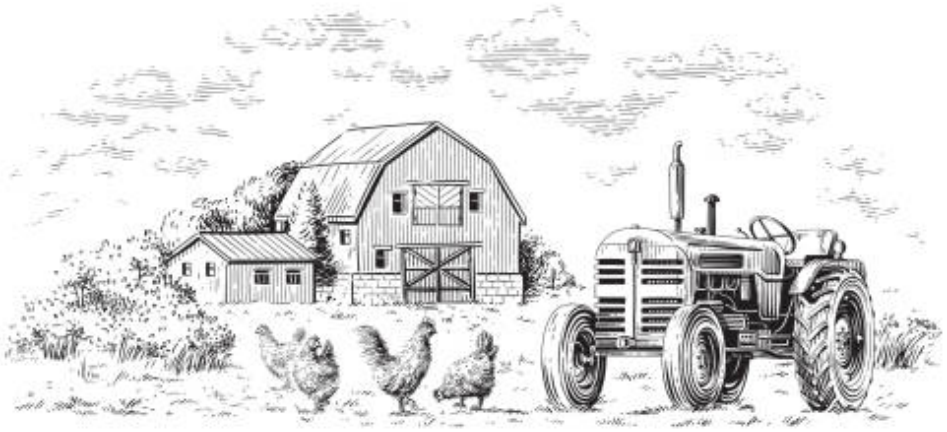
Yasmine Hedhiri & Elisa Robeau

Système expert : Cyber cultivateur



Sommaire

Introduction.....3
Base de faits : situation de départ.....4
Base de règles.....5
Fonctions de services.....6
Moteur d'inférence.....7
Conclusion.....9

Introduction

Hay Day est un jeu de simulation agricole développé par Supercell. Le but du jeu est de gérer et de développer une ferme virtuelle prospère. Les joueurs commencent avec une petite ferme et sont chargés de cultiver des cultures, d'élever des animaux, de produire des biens et d'interagir avec d'autres joueurs.

Les joueurs plantent et récoltent différentes cultures, telles que le blé, le maïs, les carottes, etc. La gestion efficace des cultures est essentielle pour maximiser les rendements. Les joueurs peuvent également s'occuper d'animaux tels que les poulets, les vaches et les moutons. L'élevage d'animaux permet de produire des ressources comme des œufs, du lait et de la laine.
En utilisant les ressources récoltées sur la ferme, les joueurs fabriquent divers produits tels que du pain, du fromage, des gâteaux, etc. Ces produits peuvent être vendus ou utilisés pour remplir des commandes. Les joueurs peuvent vendre leurs produits sur le marché du jeu. Ces ventes sont cruciales pour gagner de l'argent virtuel et acquérir des ressources.
Avec les gains, les joueurs peuvent agrandir et améliorer leur ferme en débloquant de nouveaux bâtiments, en améliorant les installations existantes, et en débloquant de nouvelles fonctionnalités.

Le système expert qu'on propose de développer repose sur des règles définies pour modéliser les décisions agricoles. Les faits reflètent l'état actuel de la ferme. Le moteur d'inférence analyse ces règles en fonction des faits, générant des recommandations personnalisées pour le joueur. Par exemple, s'il pleut beaucoup, le système peut conseiller de planter des cultures résistantes à l'eau.

Base de faits : situation de départ

Au début de la partie, le joueur se trouve doté soit de deux unités de blé, symbolisant les prémices de son exploitation agricole, soit d'une réserve d'argent lui offrant plus de flexibilité, ici dix unités. Ces deux situations initiales confèrent au joueur les ressources minimales indispensables pour commencer la partie. Le blé est surtout pour la symbolique : dans *Hay Day*, on commence l'aventure avec du blé.

Ainsi, la base de faits peut être représentée de la manière suivante :

Permet à l'utilisateur d'entrer son nom :

```
(defun choix_nom_joueur ()
 (progn
  (format t "Veuillez entrer votre nom : ")
  (setq nom (read))
  ))
```

Permet à l'utilisateur de choisir le type de départ souhaité :

```
(defun choix_depart ()
 (progn
  (format t "Veuillez entrer votre choix de depart (ble ou argent) : ")
  (setq depart (read))
  ))
```

Si le joueur décide de commencer avec de l'argent, il peut planter soit du blé, soit du maïs, soit des carottes, soit des cannes à sucre.

Il existe d'autres situations de départ, mais pour présenter le système expert, nous allons nous contenter de ces deux-là.

Base de règles

Nous avons simplifié les règles de base du jeu *Hay Day*. Voici dans les grandes lignes la base de règle (que l'on peut retrouver dans le fichier code source).

Planter des ressources :
On peut planter du blé, du maïs, des carottes et de la canne à sucre.

Récolter des produits animaliers :
On peut récolter du lait dans le cas d'une vache ou des œufs dans le cas d'une poule. Pour cela, on doit avoir de la nourriture et l'animal correspondant. Il y a donc deux autres règles pour faire la nourriture d'une vache ou d'une poule.

Produire des ressources transformées :
On peut cuisiner des plats grâce aux ressources récoltées. On peut aussi cuisiner des plats à partir d'autres ressources transformées.
Exemple : le gâteau à la carotte a besoin de 1 sucre, 2 carottes et 1 beurre. Le sucre est lui-même un produit transformé.

Vendre les produits transformés :
On ne peut vendre que les produits qu'on aura fabriqués à partir des récoltes. C'est le seul moyen de gagner de l'argent dans le jeu. Plus le produit est difficile à obtenir, plus il rapporte de l'argent.

Acheter :
On peut acheter des ressources, des animaux et la maison qui est le but final.

Fonctions de services

Fonction pour obtenir le nom du joueur :

```
(defun obtenir-nom-joueur (liste-joueur)
  (cadr (cadr (assoc 'identificateur liste-joueur))))
```

Fonction pour obtenir les ressources :

```
(defun obtenir-ressources (liste-joueur)
  (cdr (assoc 'ressources liste-joueur)))
```

Fonction pour accéder aux préconditions d'une règle :

```
(defun obtenir-preconditions (regle)
  (cdr (assoc 'precond regle)))
```

Fonction pour accéder aux effets d'une règle :

```
(defun obtenir-effets (regle)
  (cdr (assoc 'effets regle)))
```

Fonction pour accéder à l'ensemble d'une règle :

```
(defun obtenir-regle (nom-regle base-de-regles)
  (cdr (assoc nom-regle base-de-regles)))
```

Fonction pour accéder à la quantité d'une ressource dans les préconditions :

```
(defun obtenir-quantite-precondition (ressource preconditions)
  (cadr (assoc ressource preconditions)))
```

Fonction pour accéder à la quantité d'une ressource dans les effets :

```
(defun obtenir-quantite-effet (ressource effets)
  (cadr (assoc ressource effets)))
```

Fonction pour obtenir une production :

```
(defun obtenir-produit (nom liste)
  (assoc nom liste))
```

Moteur d'inférence

Pour créer notre système expert, nous avons décidé d'utiliser un chaînage avant. En effet, il est plus intéressant pour le joueur de choisir les actions qu'il veut effectuer pour sa ferme.
A cause du chaînage avant, il est possible de perdre la partie. Nous avons trouvé deux manières de perdre (il y en a peut-être plus, ce sera à découvrir en jouant).

- **Premier cas : on commence la partie avec dix unités d'argent. On achète 1 maïs et 1 blé qui demande chacun 5 unités d'argent. On décide de planter du blé. On a alors la possibilité de fabriquer de la nourriture de poule qui demande exactement 2 blé et 1 maïs. Si on fabrique cette nourriture, on se retrouve alors avec seulement 1 nourriture de poule dans nos ressources. Puisqu'on n'a pas de poule et qu'on ne peut pas vendre de la nourriture dans le jeu, on est alors bloqué et la partie prend fin.**
- **Deuxième cas : on arrive à obtenir 100 unités d'argent en vendant soit du pain, soit du sucre, soit du pop-corn, soit du jus de carotte. On n'a plus de ressources à part l'argent. On décide d'acheter une vache qui demande 100 unités d'argent. Puisqu'on n'a pas de nourriture de vache, on ne peut rien faire avec l'animal. On a alors perdu.**

Nous avons implémenté plusieurs fonctions pour éviter l'alourdissement de la fonction qui gère le système expert.

reglesValides :
La fonction reglesValides donne les actions que l'utilisateur peut effectuer. Elle parcourt la base de règles et met toutes les règles applicables dans une liste. Une règle est applicable si le joueur possède les préconditions requises.

modifRessources :
La fonction modifRessources modifie les ressources du joueur en fonction de la règle qui est appliquée. Elle va recréer une nouvelle liste de ressources. La fonction parcourt toute la liste des ressources. Pour chaque ressource :

- **si elle n'a pas à être modifiée, elle est mise telle quelle dans la nouvelle liste**
- **si elle est concernée par une précondition, on change la ressource en retirant la quantité utilisée grâce à la fonction modifRessIndivMoins (modification de la ressource individuelle - soustraction)**
- **si elle est concernée par un effet, on lui ajoute la quantité gagnée grâce à la fonction modifRessIndivPlus (modification de la ressource individuelle - addition)**
- **enfin, si elle est concernée à la fois par une précondition et par un effet, on effectue tour à tour modifRessIndivMoins et modifRessIndivPlus**

A la fin de la création de la nouvelle liste, on regarde si l'effet est dans la liste. Si le joueur n'avait pas cette ressource avant l'exécution de la règle, on doit ajouter l'effet à la liste.

depart :

La fonction depart sert à initialiser le profil du joueur. Elle demande au joueur de renseigner son nom et avec quelle ressource il veut commencer. En fonction de la ressource demandée, la fonction va créer le profil du joueur pour le système expert.

La fonction utilise les fonctions choix_nom_joueur et choix_depart qui, respectivement, demandent le nom du joueur et le choix de son départ.

cyberCultivateur :

La fonction cyberCultivateur est notre système expert. Il gère une partie de jeu.

La fonction commence par initialiser le profil du joueur avec la fonction depart. Ensuite, elle effectue une boucle : tant que l'objectif du jeu n'est pas dans les ressources du joueur, elle ne s'arrêtera pas (sauf si le joueur ne peut plus effectuer une seule action). Ici, l'objectif du jeu est de posséder une maison.

La boucle va prendre toutes les règles valides et les afficher grâce à la fonction afficher. Elle demande ensuite au joueur quelle action il veut faire. Puis elle effectue l'action et modifie les ressources du joueur. Pour un visuel plus propre, nous avons créé la fonction supRessource qui supprime toutes les ressources à 0 du joueur, sauf l'argent. Ainsi, on évite l'accumulation des ressources inutiles.

A la fin du jeu, le système annonce si le joueur a gagné ou non.

Conclusion

Cyber cultivateur offre une expérience immersive de gestion agricole où les joueurs sont invités à construire et développer leur propre ferme virtuelle. À travers la cultivation de diverses cultures, l'élevage d'animaux et la production de biens, les joueurs sont confrontés à des défis nécessitant une gestion efficace des ressources.

Notre système expert est basé sur des règles définies pour modéliser les décisions agricoles. En utilisant les faits reflétant l'état en temps réel de la ferme, le moteur d'inférence génère des recommandations personnalisées, permettant aux joueurs de choisir les actions qu'il souhaite effectuer. En effet, il existe des situations dans lesquelles le jeu peut s'arrêter car l'objectif ne sera plus atteignable. Plus qu'un jeu, notre système expert a donc aussi pour objectif de sensibiliser les joueurs à la difficulté de gérer une ferme : il faut constamment tout anticiper pour être un bon cyber cultivateur !

Nous pourrions améliorer notre système expert. Tout d'abord, le but du jeu n'est pas affiché. Le joueur ne peut donc pas savoir ce qu'il doit faire pour gagner s'il n'a pas été informé au préalable. Enfin, seules les règles applicables sont affichées. On ne peut donc pas savoir à quoi nous servira l'action. Mais on aurait surchargé l'interface si on avait mis toutes les règles.